Содержание

- Введение 1.1
- Глоссарий 1.2
- Безопасность 1.3
- Сборка 1.4
  - Клевер 4.2 Pro 1.4.1
  - Клевер 4.2 1.4.2
  - Клевер 4.2 WorldSkills 1.4.3
  - Клевер 4.1 1.4.4
  - Клевер 4 1.4.5
  - Клевер 3 1.4.6
  - Клевер 2 1.4.7
- Настройка 1.5
  - Калибровка датчиков 1.5.1
  - Настройка пульта 1.5.2
    - Работа с FS-A8S 1.5.2.1
  - Полетные режимы 1.5.3
  - Настройка питания 1.5.4
  - Настройка Failsafe 1.5.5
- Ручной полет 1.6
  - Упражнения 1.6.1
- Работа с Raspberry Pi 1.7
  - Образ для RPi 1.7.1
  - Подключение по Wi-Fi 1.7.2
  - Подключение к Pixracer 1.7.3
  - QGroundControl по Wi-Fi 1.7.4
  - SSH-доступ 1.7.5
  - Командная строка 1.7.6
  - Автоматическая проверка 1.7.7
  - Просмотр видеострима с камер 1.7.8
- Программирование 1.8
  - Настройка камеры 1.8.1
  - Визуальные маркеры (ArUco) 1.8.2
    - Распознавание маркеров 1.8.2.1
    - Навигация по карте маркеров 1.8.2.2
  - Навигация по Optical Flow 1.8.3

- Автономный полет (OFFBOARD) 1.8.4
- Системы координат 1.8.5
- Примеры кода 1.8.6
- Лазерный дальномер 1.8.7
- Светодиодная лента 1.8.8
- Работа с GPIO 1.8.9
- Ультразвуковой дальномер 1.8.10
- Компьютерное зрение 1.8.11
- Визуализация с помощью rviz и rqt 1.8.12
- Автозапуск ПО 1.8.13
- Использование JavaScript 1.8.14
- Блочное программирование 1.8.15
- Симулятор 1.8.16
  - Сборка на собственной машине 1.8.16.1
  - Установка виртуальной машины 1.8.16.2
  - Использование симулятора 1.8.16.3
  - Установка на компьютеры с M1 1.8.16.4
- ROS 1.8.17
- MAVROS 1.8.18

Дополнительные материалы 1.9

- COEX Pix 1.9.1
- COEX PDB 1.9.2
- COEX GPS 1.9.3
- Использование SSH-ключей 1.9.4
- Радио-телеметрия 1.9.5
- Камера Hawk Eye 1.9.6
- Гид по автономному полету 1.9.7
- Имя хоста 1.9.8
- Симулятор 1.9.9
- Настройка PID 1.9.10
- Навигация по настенным маркерам 1.9.11
- CAD-модели Клевера 1.9.12
- Docker-контейнер с симулятором 1.9.13
- Установка ROS Melodic 1.9.14
- Калибровка камеры 1.9.15
- Подключение к VPN ZeroTier 1.9.16
- Подключение к VPN Hamachi 1.9.17
- Управление мультикоптером при помощи 4G связи 1.9.18

- Пакеты Клевера на Jetson Nano 1.9.19
- Пилотирование со смартфона 1.9.20
- Настройка сети RPi 1.9.21
- Интерфейс UART 1.9.22
- Параметры PX4 1.9.23
- Работа с логами PX4 1.9.24
- Прошивка PX4 1.9.25
- Протокол MAVLink 1.9.26
- Использование мультиметра 1.9.27
- Неисправности радиоаппаратуры 1.9.28
- Прошивка ESC контроллеров 1.9.29
- Взаимодействие с Arduino 1.9.30
- Подключение GPS 1.9.31
- Работа с ИК датчиками 1.9.32
- Установка FPV 1.9.33
- Установка FPV (Клевер 3) 1.9.34
- Магнитный захват 1.9.35
- Механический захват 1.9.36
- Груз для магнитного захвата 1.9.37
- Сборка шаровой защиты 1.9.38
- Управление в режиме тренера 1.9.39
- Техника лужения 1.9.40
- Подключение PX4FLOW 1.9.41
- Типы силовых разъемов 1.9.42
- Модуль ESP8266 1.9.43
- Сборка и модификация образа Клевера 1.9.44
- Подключение регулятора 4 в 1 1.9.45
- Светодиодная лента (legacy) 1.9.46
- Вклад в Клевер 1.9.47
- Репозиторий пакетов COEX 1.9.48
- Тестирование Клевера 1.9.49
- Переход на версию 0.20 1.9.50
- Переход на версию 0.22 1.9.51
- COEX DuoCam 1.9.52
  - Виртуальная MAVLink-камера 1.9.52.1
  - Настройка DuoCam 1.9.52.2
- Мероприятия 1.10
  - CopterHack-2023 1.10.1

- CopterHack-2022 1.10.2
- CopterHack-2021 1.10.3
- CopterHack-2019 1.10.4
- Олимпиада НТИ 2019 1.10.5
- Робокросс-2019 1.10.6
- CopterHack-2018 1.10.7
- CopterHack-2017 1.10.8
- Конкурс видео 1.10.9
- Образовательные конкурсы 1.10.10

Проекты на базе Клевера 1.11

- Система радионавигации 1.11.1
- Advanced Clover Simulator 1.11.2
- Copter For Space 1.11.3
- CopterCat CM4 1.11.4
- Система мониторинга воздуха 1.11.5
- Контроль соблюдения ПДД на выделенной полосе с дроном 1.11.6
- Система автоматической посадки (AMLS) 1.11.7
- Разработка системы для управления БПЛА с помощью шлема виртуальной реальности 1.11.8
- Шоу коптеров 1.11.9
- Innopolis Open 2020 (L22_ÆRO) 1.11.10
- Олимпиада НТИ 2020 (P4DF2) 1.11.11
- Генератор ArUco карт 1.11.12
- Модель аэротакси в городе 1.11.13
- Шаровая защита коптера 1.11.14
- Система засечки для дронов 1.11.15
- Дрон для 3D-сканирования человека 1.11.16
- Распознавание лиц 1.11.17
- Управление дроном силой мысли 1.11.18
- Подсчет количества объектов с камеры 1.11.19
- Пульт на Андроид 1.11.20
- Блочный конструктор полета 1.11.21
- Калибровка камеры (legacy) 1.11.22
- Управление дроном для оценки позы человека 1.11.23
- Распознавание видов агрокультур 1.11.24
- AdvancedClover 1.11.25
- Дрон для высаживания семян 1.11.26
- Граффити коптер D-drone 1.11.27
- Программируемый летающий автомобиль 1.11.28

Дрон-Агроном 1.11.29
Easy To Fly 1.11.30
Хардатон Квиддич 1.11.31
Октокоптер со специфичным расположением пропеллеров 1.11.32

Учебник

Теория и видеоуроки 2.1
Учебно-методическое пособие 2.2
Контрольные материалы 2.3

Клевер



«Клевер» — это учебный конструктор программируемого квадрокоптера, состоящего из популярных открытых компонентов, а также набор необходимой документации и библиотек для работы с ним.

Набор включает в себя полетный контроллер COEX Pix с полетным стеком PX4, Raspberry Pi 4 в качестве управляющего бортового компьютера, модуль камеры для реализации полетов с использованием компьютерного зрения, а также набор различных датчиков и другой периферии.

Платформа Клевера также включает в себя преднастроенный образ для Raspberry Pi в полным набором необходимого ПО для работы со всей периферией и программирования автономных полетов. Исходный код платформы Клевера и данной документации открыт и доступен на GitHub.

Если вы детально изучили документацию, но так и не нашли ответа на свой вопрос, напишите в чат техподдержки и наши специалисты вам с радостью ответят: @COEXHelpdesk.

Также у нас есть чат для программистов, которые разрабатывают под PX4, автономную навигацию в помещениях и рои дронов: @DroneCode.

Чат по разработке самой платформы Клевера и образа для RPi: @devclover.

Вы можете скачать PDF-версию этой документации. The English version of this documentation is available.

Глоссарий

БПЛА

Беспилотный летательный аппарат. Примеры: квадрокоптер, гексакоптер, самолет, летающее крыло, конвертоплан (VTOL), вертолет.

Квадрокоптер

Беспилотный летательный аппарат с 4-мя винтами и электронной системой стабилизации.

Мультикоптер

Беспилотный летательный аппарат с электронной системой стабилизации и числом винтов, равным 3 (трикоптер), 4 (квадрокоптер), 6 (гексакоптер), 8 (октокоптер) или более.

Полетный контроллер / автопилот

1. Специализированная плата, спроектированная для управления мультикоптером, самолетом или другим аппаратом. Примеры: Pixhawk, ArduPilot, Naze32, CC3D.

2. Программное обеспечение для платы управления мультикоптером. Примеры: PX4, APM, CleanFlight, BetaFlight.

Прошивка

Программное обеспечение, управляющее работой какого-либо устройства, например, полетного контроллера или регулятора мотора (ESC).

Мотор

Электродвигатель, который вращает винты мультикоптера. Обычно используются бесколлекторные электродвигатели. Такие двигатели подключаются к ESC.

ESC / регулятор двигателя / "регуль"

Electronic Speed Controller. Специализированная плата, которая управляет скоростью вращения бесколлекторного электродвигателя. Управляется полетным контроллером при помощи широтно-импульсной модуляции (ШИМ).

ESC имеет прошивку, которая определяет особенности его работы.

АКБ / аккумулятор / батарея

Перезаряжаемый источник тока для БПЛА. В квадрокоптерах обычно применяются Li-po (литий-полимерные) аккумуляторы.

Ячейка / "банка" АКБ

Составная часть АКБ, непосредственный источник тока. Обычно АКБ для БПЛА состоят из нескольких (2–6) ячеек, соединенных последовательно. Максимальное напряжение одной Li-po ячейки – 4.2 В; общее напряжение АКБ равно суммарному напряжению ячеек. Количество ячеек обозначается буквой *S*, например: *2S*, *3S*, *4S*.

В Клевере обычно применяются аккумуляторы *3S*.

Пульт / аппаратура радиоуправления / "аппа"

Пульт для управления квадрокоптером, работающий по радиоканалу. Для работы пульта к полетном контроллеру необходимо подключить ресивер.

Клевером, также, можно управлять со смартфона.

Телеметрия

1. Передача данных о состоянии квадрокоптера или другого аппарата на расстояние.

2. Совокупность данных о состоянии аппарата, так таковая (высота, ориентация, глобальные координаты и т. д.).

3. Система для передачи данных о состоянии аппарата или команд для него по воздуху. Примеры: радиомодемы (RFD900, 3DR Radio Modem), Wi-Fi модули (ESP-07). Raspberry Pi на Клевере также может быть использован в качестве модуля для телеметрии: использование QGroundControl через Wi-Fi.

Арминг

Armed – состояние коптера готовности к полету. При поднятии стика газа либо при посылке внешней команды с целевой точкой – коптер полетит. Обычно коптер начинает вращать винтами при переходе в состояние "armed" даже если стик газа находится внизу.

Противоположным состоянием является Disarmed.

PX4

Популярный полетный контроллер с открытым исходным кодом, работающий на платах Pixhawk, Pixracer и других. PX4 рекомендуется для использования на Клевере.

Raspberry Pi

Популярный одноплатный микрокомпьютер, использующийся в конструкторе Клевер.

Образ SD-карты

Полная копия содержимого SD-карты, представленная в виде файла. Такой файл можно загрузить на SD-карту, воспользовавшись специальной утилитой, например Etcher. SD-карта, вставленная в Raspberry Pi является единственным его долговременным хранилищем и полностью определяет, что он будет делать.

Конструктор Клевер включает в себя рекомендованный образ для SD-карты.

APM / ArduPilot

Полетный контроллер с открытым исходным кодом, изначально созданный для платы Arduino. Впоследствии был портирован на Pixhawk, Pixracer и другие платы.

MAVLink

Протокол для взаимодействия дронов, наземных станций и других аппаратов по радиоканалам. Обычно именно этот протокол используется для телеметрии.

ROS

Популярный фреймворк для написания сложных роботехнических приложений.

MAVROS

Библиотека-связующее звено между аппаратом, работающем по протоколу MAVLink, и ROS.

UART

Последовательный асинхронный интерфейс передачи данных, применяемый во многих устройствах. Например, GPS антенны, Wi-Fi роутеры или Pixhawk.

IMU

Inertial measurement unit. Комбинация датчиков (гироскоп, акселерометр, магнитометр), которая помогает БПЛА рассчитывать ориентацию и положение в пространстве.

Estimation

Процесс определения ПО полетного контроллера состояния квадрокоптера: положения в пространстве, скоростей, углов наклона и т. д. Для этого используется смешивание данных с установленных датчиков и различные алгоритмы фильтрации, например фильтр Калмана.

В прошивке PX4 есть два модуля для estimation'a: LPE и ECL EKF (EKF2).

В прошивке APM эту функцию выполняет подсистема EKF2.

Инструкция по безопасности

Пайка



Работы, связанные с пайкой и лужением, должны проводиться в специально оборудованных и предварительно подготовленных помещениях. Обязательно должна присутствовать система вентиляции.

Перед началом работы необходимо:

1. Привести в порядок рабочее место, ничего не должно мешать процессу. Рабочее место должно быть хорошо освещено.
2. Проверить целостность проводки и штепсельной вилки.
3. Паяльник, находящийся в рабочем состоянии, установить в зоне действия местной вытяжной вентиляции, в специальную подставку.
4. Перед началом работы надеть защитный халат, очки и, при необходимости, перчатки.

Во время пайки:

1. Паяльник следует держать только за ручку, так как жало имеет высокую температуру.



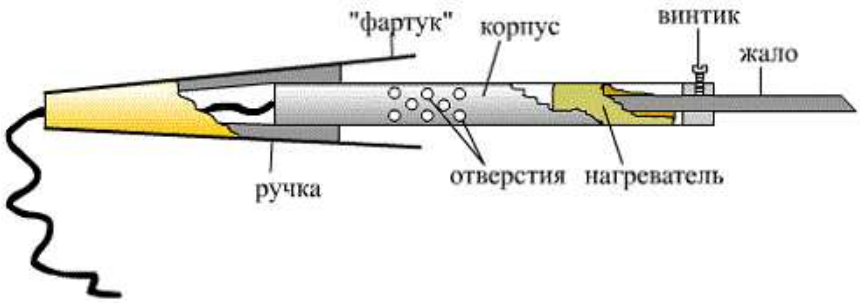
2. Жало паяльника нагревается до очень высокой температуры, поэтому, в случае его прикосновения к электрическому проводу в ходе пайки изоляция будет повреждена в считанные мгновения, с последующим коротким замыканием.
3. Для перемещения изделий применять специальные инструменты (пинцеты, клещи или другие инструменты), обеспечивающие безопасность при пайке.
4. Во избежание ожогов расплавленным припоем при распайке не выдергивать резко с большим усилием паяемые провода.
5. При пайке мелких и подвижных изделий пользоваться специальным держателем.



6. Паяльник переносить за корпус, а не за провод или рабочую часть. При перерывах в работе паяльник отключать от электросети.

> При обнаружении неисправной работы паяльника или возникновении возгорания отключить его от питающей электросети.

Канифоль и припой при плавлении выделяют значительное количество вредных веществ. Настойчиво советуется проветривать помещение после каждой пайки. Через каждые 30 минут нужно делать небольшие перерывы со сквозным проветриванием помещения и не забывать при этом отключать паяльник.



Полеты

Безопасность при подготовке к вылету

- Убедиться, что Li-ion аккумуляторы заряжены.
- Убедиться, что батарейки в аппаратуре управления заряжены.

- Устанавливать пропеллеры только перед вылетом.

Проверить надёжность следующих узлов:

- Затянутость гаек пропеллеров.
- Крепление и целостность защит винтов.
- Надежность крепления проводов, отсутствие болтающихся проводов.

Безопасность перед вылетом

- Располагать зрителей за спиной пилота или за линией, проходящей через оба плеча пилота за спиной пилота.
- Не допускать выхода зрителей в полусферу перед лицом пилота.
- Знать и помнить время полёта, на которое рассчитан данный коптер и его аккумулятор.
- ДО подключения Li-ion аккумулятора включить аппаратуру управления (пульт), перевести левый стик (газ) в нулевое положение.
- Подключать Li-ion аккумулятор только перед взлётом, отключать сразу после взлёта.
- Стоять на расстоянии не менее 3 м от коптера.
- Взлетать с земли с ровной площадки, на расстоянии не менее 3 метров от препятствий.

Безопасность в полете

- Выполнять все указания преподавателя или лётного инструктора.
- Заранее обозначить зону пилотажа. Летать только в обозначенной зоне и не допускать вылета за её пределы. Не залетать за собственную спину.
- При обучении полётам летать на уровне ниже собственного роста.
- Летать рядом с собой на расстоянии, на котором вам видна ориентация коптера в пространстве. Не улетать далеко от себя. В случае сомнений в ориентации коптера немедленно выполнить посадку на месте. Не пытаться взлететь. Подойти ближе к коптеру и выполнить взлёт.
- При управлении все движения стиками выполнять аккуратно и плавно. Не допускать резких движений. При необходимости изменить направление полёта двигать стиками следует энергично, но не резко.
- Летать следует осторожно и выполнять только те элементы, в которых нет сомнений. Запрещается выполнять фигуры пилотажа, в успехе которых возникают сомнения и фигуры, связанные с риском.
- Соблюдать скоростной режим. Скорость полёта коптера держать в пределах скорости идущего человека.
- Вернуть коптер к месту посадки к рассчитанному времени, не допускать полной разрядки аккумулятора в полёте.
- Посадку выполнять только на ровную открытую площадку вдали от препятствий.

Аварийная посадка

В случае удара об землю или жесткой посадки выполнить следующие действия:

1. Прекратить полёт. Посадить коптер на землю. Левый стик (газ) в минимум.
2. Disarm (левый стик влево-вниз на 3 секунды).
3. Отключить Li-ion аккумулятор на коптере.
4. Выключить пульт.
5. Осмотреть коптер и при необходимости отремонтировать.

Запланированная посадка

После запланированной посадки выполнить следующие действия:

1. Disarm (Левый стик влево-вниз на 3 секунды)
2. Отключить Li-ion аккумулятор на коптере.
3. Выключить пульт.

Сборка Клевера

В этом разделе находится статьи, описывающие сборку каждой из версии Клевера.

Версия	Изображение
Клевер 4.2 (WorldSkills, Pro)	
Клевер 4.1	
Клевер 4	
Клевер 3	
Клевер 2	

Ссылки на CAD-модели деталей Клевера доступны в статье "CAD-модели".

Определение версии

Для определения версии Клевера 4 посмотрите на серийный номер.



В данном случае число *420* обозначает версию Клевера **4.2**.

Сборка Клевера 4.2 Pro

Размер крепежа

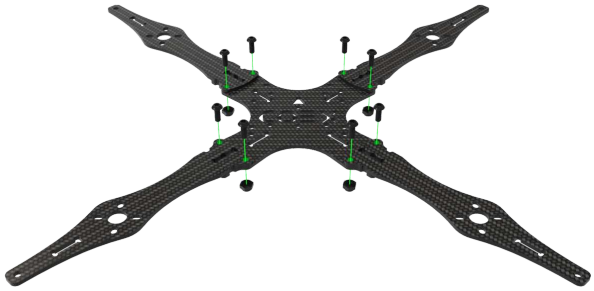
Во время сборки используются винты и стойки различных размеров, использование крепежа не соответствующего размера может повредить коптер.

	Винт M3x10		Стойка алюминиевая 40 мм
	Винт M3x8		Стойка алюминиевая 15 мм
	Винт M3x5		Стойка нейлоновая 40 мм
	Саморез M2x5		Стойка нейлоновая 30 мм
	Гайка M3 (самоконтрящаяся)		Стойка нейлоновая 20 мм
	Гайка M3 (нейлоновая)		Стойка нейлоновая 15 мм
	Стойка демпферная		Стойка нейлоновая 6 мм

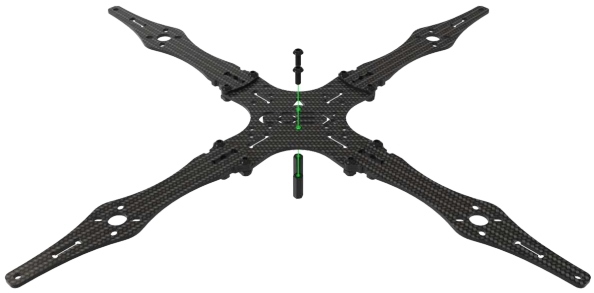
Сборка рамы

1. Совместите 4 луча с центральной декой, зафиксируйте их при помощи винтов M3x8 и гаек с нейлоновой вставкой.

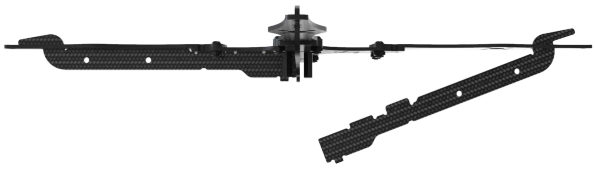




2. На центральные отверстия в главной деке установите 2 стойки 15 мм и закрепите их с помощью винтов M3x8.



3. Установите крючок пластины жесткости в паз на луче.



4. Прижмите пластины жесткости к главной деке.



5. Стяните пластины жесткости с помощью малой карбоновой деки.

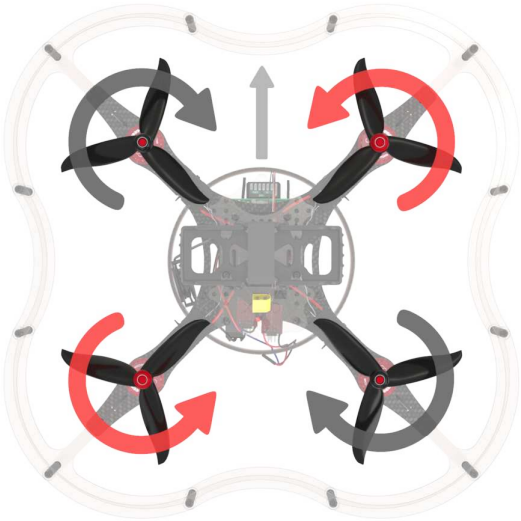


6. Установите 4 нейлоновые стойки 6 мм и закрепите их с помощью винтов M3x5.



Установка моторов

1. При установке моторов обратите внимание на схему вращения моторов. Маркировка вращения на моторах должна совпадать со схемой вращения.



2. Установите мотор на соответствующие отверстия в луче с помощью **винтов M3x5**.

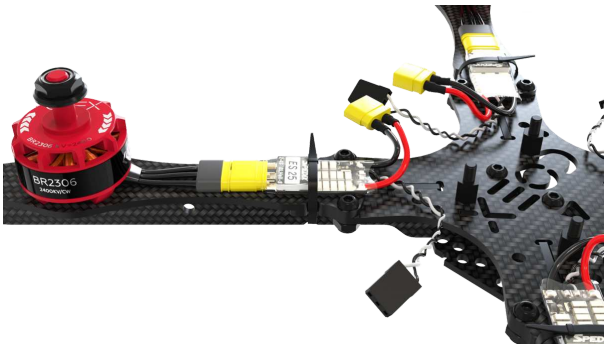


> Убедитесь, что моторы закреплены с помощью винтов M3x5, в противном случае может возникнуть короткое замыкание между обмотками.

Установка ESC и PDB

1. Подсоедините к моторам регуляторы оборотов (ESC) с помощью разъемов MR30 и закрепите их на лучах с помощью хомутов.





2. На заранее закрепленные стойки установите плату распределения питания (PDB) и зафиксируйте ее стойками 6мм. Плата распределения питания должна быть установлена таким образом, чтобы кабель подключения питания был направлен в сторону хвоста коптера.



3. Подключите к плате распределения питания силовые выходы регуляторов оборотов.



Если вы планируете использовать камеру Hawk Eye, припаяйте провода питания на данном этапе, так как после установки полетного контроллера сделать это будет затруднительно

Установка полетного контроллера

Набор "Клевер 4" позволяет установить различные полетные контроллеры, к примеру COEX Pix и Pixracer.

> При установке полетного контроллера обратите внимание на ориентацию платы. Если Вы установите COEX Pix серворазъемами назад (как на изображениях в инструкции) то впоследствии при настройке полетного контроллера в *Autopilot Orientation* необходимо будет указать значение `ROTATION_ROLL_180_YAW_90`, иначе полетный контроллер будет некорректно воспринимать наклоны и повороты коптера. Для полетного контроллера Pixracer это не требуется.

COEX Pix

> Перед установкой демпферных стоек, накрутите 2 слоя нейлоновых гаек, для более прочного крепления или откусите лишнюю резьбу с помощью бокорезов.

Установите демпферные стойки 6мм, на них закрепите COEX Pix с помощью нейлоновых гаек.



Pixracer

1. Установите малую деку на стойки и зафиксируйте ее нейлоновыми гайками.

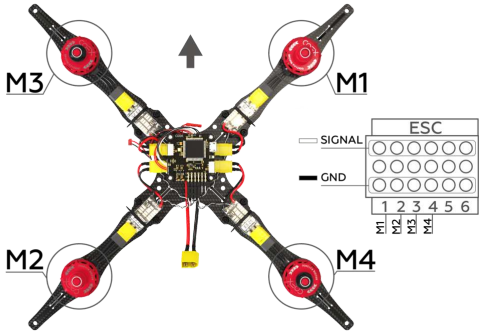


2. Склейте 3–4 слоя двустороннего скотча, приклейте его в центр малой деки и установите сверху Pixracer.



Подключение полетного контроллера

1. Подключите регуляторы оборотов к полетному контроллеру в соответствии со схемой.



2. Подключите кабель питания к плате распределения питания (PDB) и соответствующему разъему на полетном контроллере.



3. Установите алюминиевые стойки 40 мм на винты M3x10.



Установка Raspberry Pi

1. На основную деку установите стойки 20 мм, закрепите их с помощью винтов M3x8.



2. На монтажную деку установите стойки 6мм и стойки 30 мм, закрепите их с помощью винт M3x5 и M3x10 соответственно.



3. Установите собранную монтажную деку на основную и закрепите с помощью винтов M3x8.



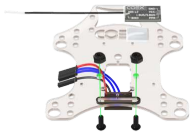
4. Установите плату Raspberry Pi и зафиксируйте с помощью нейлоновых гаек.



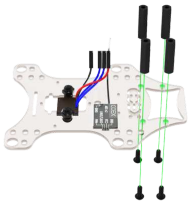
5. На деку захвата установите дальномер с помощью самоконтрящихся гаек и винтов M3x8, и приклейте радиоприемник с помощью двустороннего скотча.

> Обратите внимание, что крепящие гайки расположены обратной стороны от платы дальномера, как на схеме, иначе есть большая вероятность повредить плату.

> Также плата может быть закреплена на саморезы M2 в соседние отверстия.



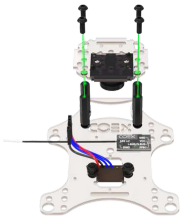
6. Установите 4 стойки 20мм и закрепите их с помощью винтов M3x8.



7. На малую монтажную деку установите камеру и зафиксируйте ее с помощью 2ух саморезов M2x5, в верхнем левом и нижнем правом углах.



8. Установите модуль камеры на деку захвата и закрепите с помощью винтов M3x8.



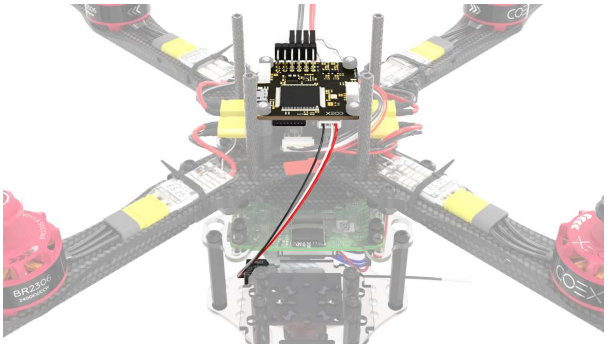
9. Установите собранную деку захвата и зафиксируйте с помощью винтов M3x8.



10. Подключите к Raspberry Pi дальномер и кабель питания.



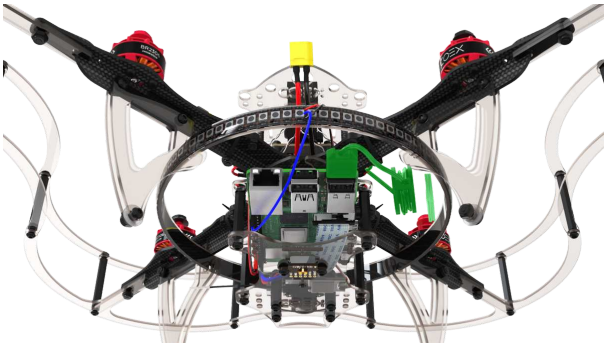
11. Подключите радиоприемник к полетному контроллеру используя разъем *RCIN*.



12. Подключите шлейф камеры к Raspberry Pi.



13. Подключите полетный контроллер к Raspberry Pi с помощью USB-кабеля.

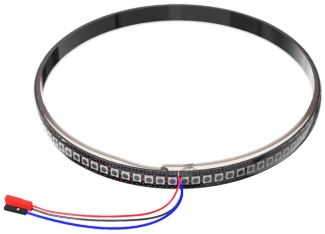


Установка LED ленты и ножек

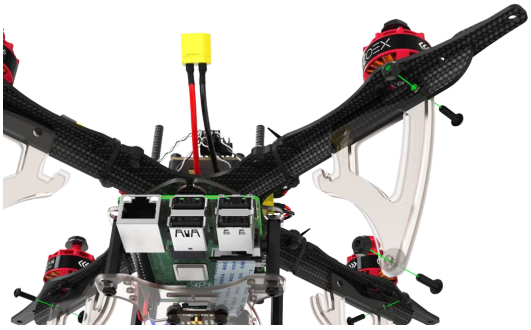
1. Соберите обруч для светодиодной ленты, объединив замок на концах.



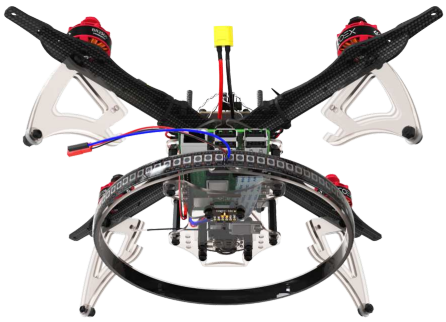
2. Наклейте светодиодную ленту на обруч, для большей крепкости притяните ее с помощью 3-4 хомутов.



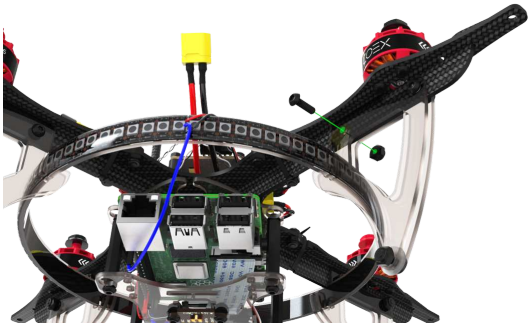
3. Установите ножки на пластину жесткости с помощью самоконтрящихся гаек и винтов M3x10 используя только крайние монтажные отверстия. Снизу, между пластинами ножек установите демпферное силиконовое колечко.



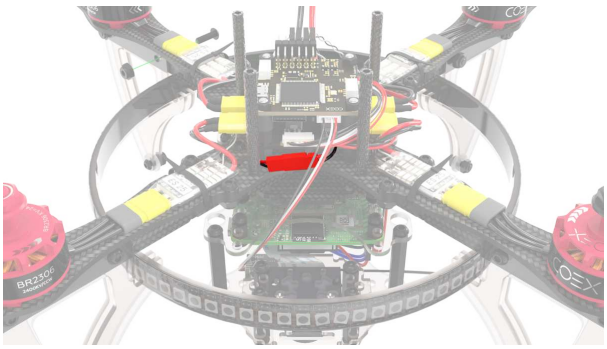
4. Отогните ножки назад и в специальный паз на них установите обруч со светодиодной лентой, таким образом, чтобы кабели подключения выходили с хвостовой стороны коптера.



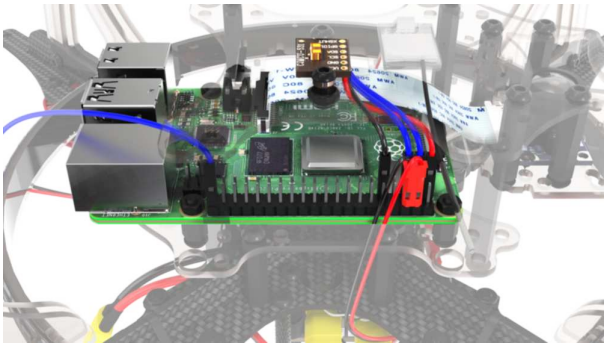
5. Закрепите ножки с помощью самоконтрящихся гаек и винтов M3x10.



6. Подключите питание светодиодной ленты (красный, черный кабели) в короткий разъем JST на плате распределения питания.



7. Подключите сигнальный выход светодиодной ленты (белый кабель) в Raspberry Ri, к пину *GPIO21*.

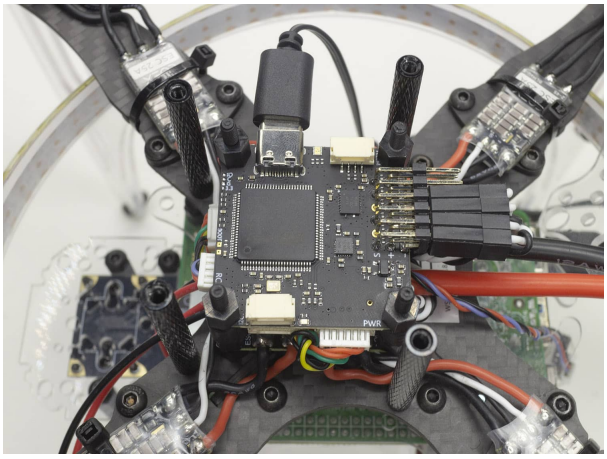


Установка камеры Hawk Eye

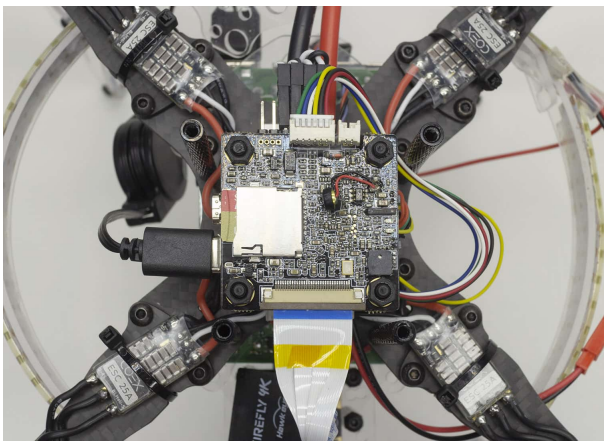
Установите скобу на камеру и зафиксируйте её винтами M2.



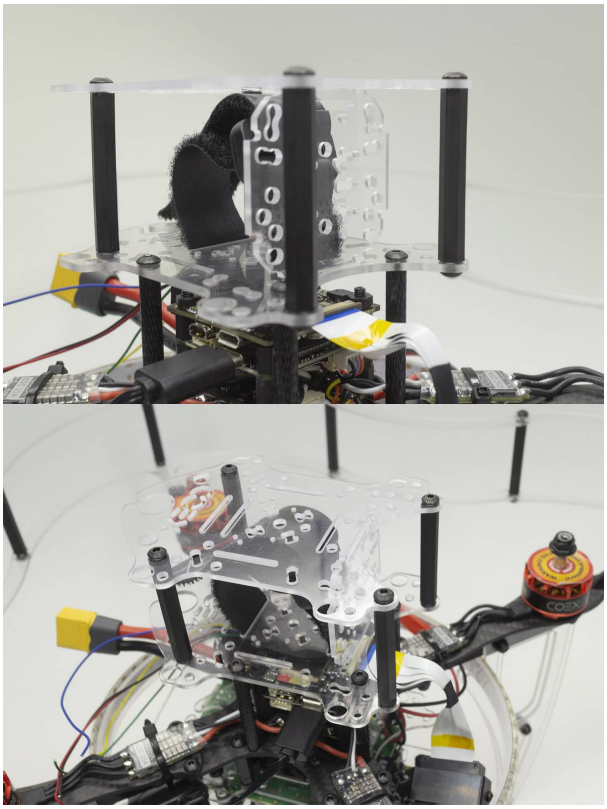
Зафиксируйте полетный контроллер нейлоновыми стойками 6мм.



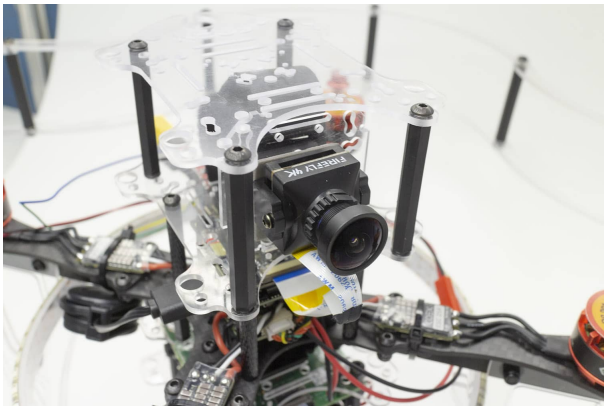
Сверху на стойки установите плату камеры, зафиксируйте её нейлоновыми гайками и подключите шлейф.



Установите верхнюю деку. Установите 4 нейлоновых стойки 40мм, вставьте малую монтажную деку вертикально и зафиксируйте сверху еще одной монтажной декой, как показано на фотографии.



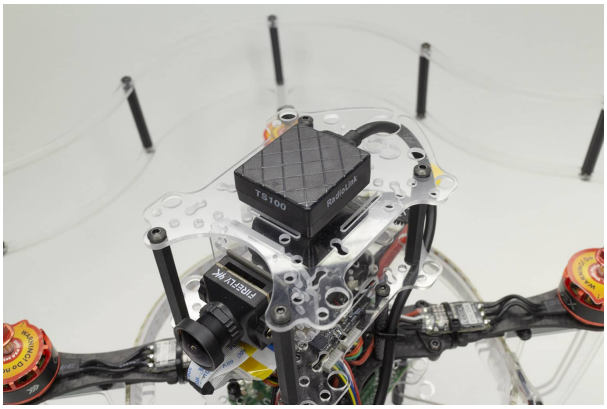
Зафиксируйте камеру на малой монтажной деке при помощи двустороннего скотча.



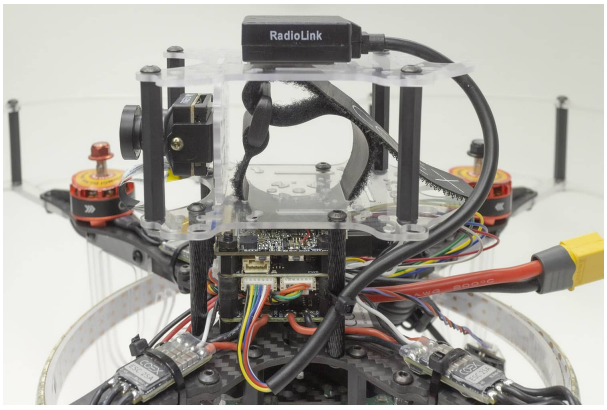
Теперь можно производить настройку камеры.

Установка GPS

Зафиксируйте GPS-приемник на верхней деке при помощи двустороннего скотча. Стрелка на приемнике должна совпадать с передом коптера.



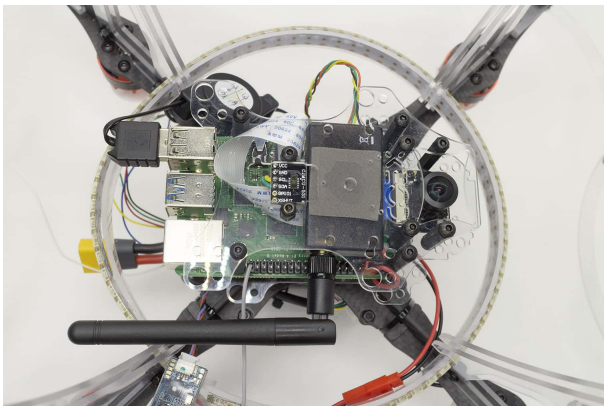
Подключите приемник в разъем GPS полетного контроллера, и зафиксируйте кабель при помощи стяжки.



Настройка и использование GPS описаны в данной статье.

Установка радио-телеметрии

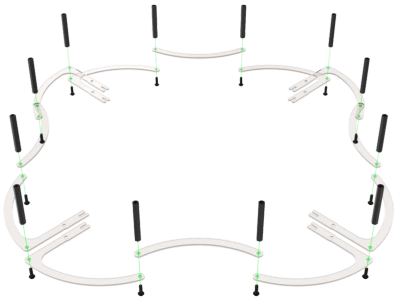
Зафиксируйте модуль телеметрии на нижней деке при помощи двустороннего скотча.



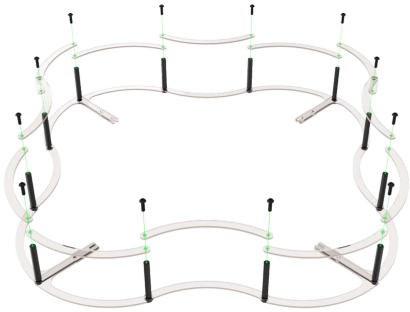
Подключите модуль при помощи кабеля в порт Telem1 полетного контроллера. Теперь можно выполнить настройку.

Установка защиты

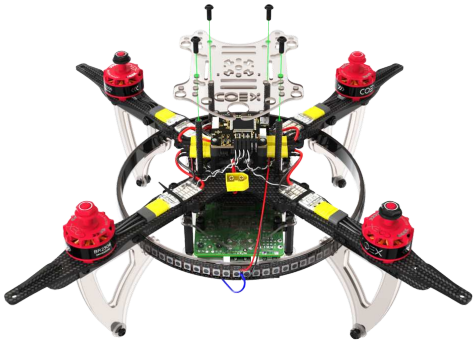
1. Соберите нижний уровень защиты с помощью стоек 40 мм и винтов M3x10.



2. Соберите верхний уровень защиты с помощью винтов M3x10.



3. Установите монтажную деку и закрепите ее винтами M3x8.



4. Установите защиту и закрепите на лучах с помощью самоконтрящихся гаек и винтов M3x10.



Дрон собран, далее произведите настройку.

Сборка Клевера 4.2

Габаритный чертеж – clover-4.2.pdf.

Размер крепежа

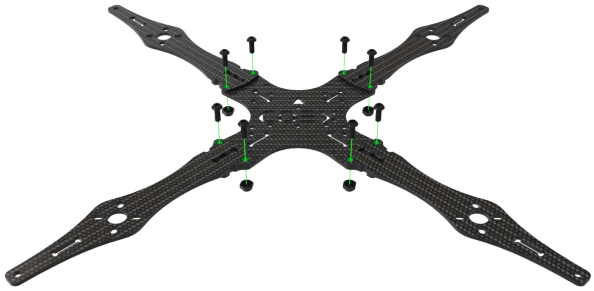
> Во время сборки используются винты и стойки различных размеров, использование крепежа не соответствующего размера может повредить коптер.

	Винт M3x10		Стойка алюминиевая 40 мм
	Винт M3x8		Стойка алюминиевая 15 мм
	Винт M3x5		Стойка нейлоновая 40 мм
	Саморез M2x5		Стойка нейлоновая 30 мм
	Гайка M3 (самоконтрящаяся)		Стойка нейлоновая 20 мм
	Гайка M3 (нейлоновая)		Стойка нейлоновая 15 мм
	Стойка демпферная		Стойка нейлоновая 6 мм

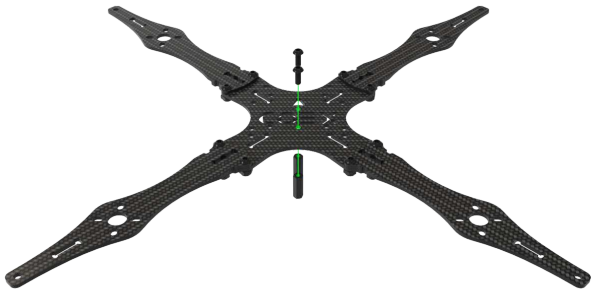
Сборка рамы

1. Совместите 4 луча с центральной декой, зафиксируйте их при помощи винтов M3x8 и гаек с нейлоновой вставкой.

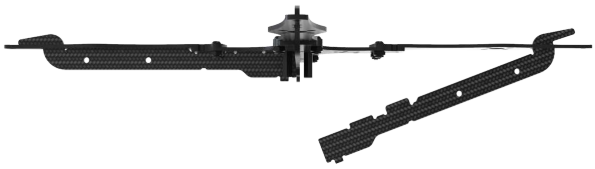




2. На центральные отверстия в главной деке установите 2 алюминиевые стойки 15 мм и закрепите их с помощью винтов M3x8.



3. Установите крючок пластины жесткости в паз на луче.



4. Прижмите пластины жесткости к главной деке.



5. Стяните пластины жесткости с помощью малой карбоновой деки.

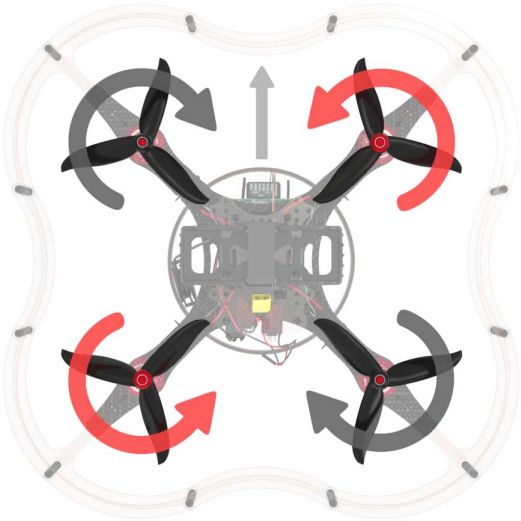


6. Установите 4 нейлоновые стойки 6 мм и закрепите их с помощью винтов M3x5.



Установка моторов

1. При установке моторов обратите внимание на схему вращения моторов. Маркировка вращения на моторах должна совпадать со схемой вращения.



2. Установите мотор на соответствующие отверстия в луче с помощью **винтов M3x5**.

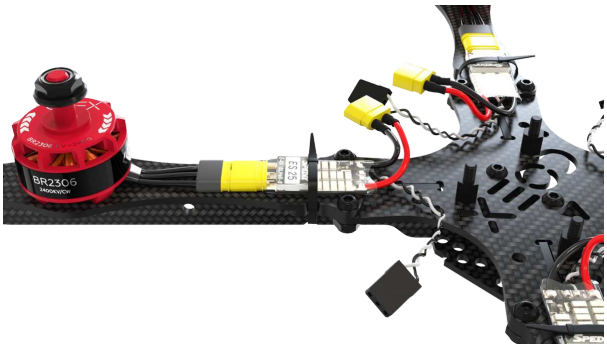


> Убедитесь, что моторы закреплены с помощью винтов M3x5, в противном случае может возникнуть короткое замыкание между обмотками.

Установка ESC и PDB

1. Подсоедините к моторам регуляторы оборотов (ESC) с помощью разъемов MR30 и закрепите их на лучах с помощью хомутов.





2. На заранее закрепленные стойки установите плату распределения питания (PDB) и зафиксируйте ее стойками 6мм. Плата распределения питания должна быть установлена таким образом, чтобы кабель подключения питания был направлен в сторону хвоста коптера.



3. Подключите к плате распределения питания силовые выходы регуляторов оборотов.



Установка полетного контроллера

Набор "Клевер 4" позволяет установить различные полетные контроллеры, к примеру COEX Pix и Pixracer.

> При установке полетного контроллера обратите внимание на ориентацию платы. Если Вы установите COEX Pix серворазъемами назад (как на изображениях в инструкции) то впоследствии при настройке полетного контроллера в *Autopilot Orientation* необходимо будет указать значение `ROTATION_ROLL_180_YAW_90`, иначе полетный контроллер будет некорректно воспринимать наклоны и повороты коптера. Для полетного контроллера Pixracer это не требуется.

COEX Pix

> Перед установкой демпферных стоек, накрутите 2 слоя нейлоновых гаек, для более прочного крепления или откусите лишнюю резьбу с помощью бокорезов.

Установите демпферные стойки 6мм, на них закрепите COEX Pix с помощью нейлоновых гаек.



Pixracer

1. Установите малую деку на стойки и зафиксируйте ее нейлоновыми гайками.

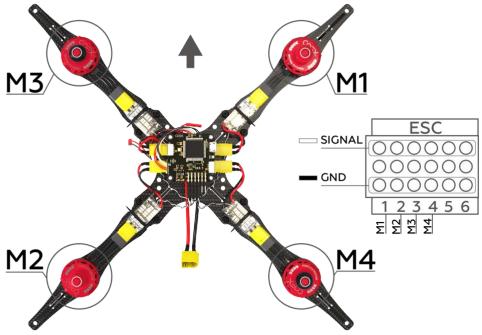


2. Склейте 3–4 слоя двустороннего скотча, приклейте его в центр малой деки и установите сверху Pixracer.



Подключение полетного контроллера

1. Подключите регуляторы оборотов к полетному контроллеру в соответствии со схемой.



2. Подключите кабель питания к плате распределения питания (PDB) и соответствующему разъему на полетном контроллере.



3. Установите алюминиевые стойки 40 мм на винты M3x10.



Установка Raspberry Pi

1. На основную деку установите стойки 20 мм, закрепите их с помощью винтов M3x8.



2. На монтажную деку установите стойки M2.5x6мм и стойки M3x30 мм, закрепите их с помощью винтов M2.5x4 и M3x10 соответственно.



3. Установите собранную монтажную деку на основную и закрепите с помощью винтов M3x8.



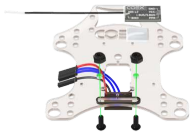
4. Установите плату Raspberry Pi и зафиксируйте с помощью винтов M2.5x4.



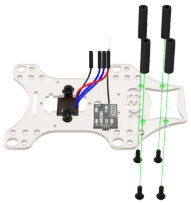
5. На деку захвата установите дальномер с помощью самоконтрящихся гаек и винтов M3x8, и приклейте радиоприемник с помощью двустороннего скотча.

> Обратите внимание, что крепящие гайки расположены обратной стороны от платы дальномера, как на схеме, иначе есть большая вероятность повредить плату.

> Также плата может быть закреплена на саморезы M2 в соседние отверстия.



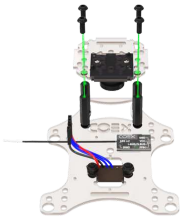
6. Установите 4 стойки 20мм и закрепите их с помощью винтов M3x8.



7. На малую монтажную деку установите камеру и зафиксируйте ее с помощью 2ух саморезов M2x5, в верхнем левом и нижнем правом углах.



8. Установите модуль камеры на деку захвата и закрепите с помощью винтов M3x8.



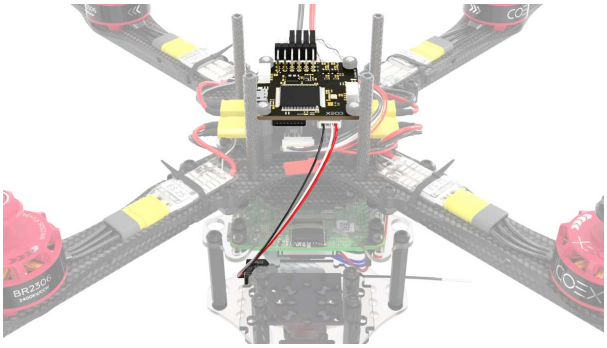
9. Установите собранную деку захвата и зафиксируйте с помощью винтов M3x8.



10. Подключите к Raspberry Pi дальномер и кабель питания.



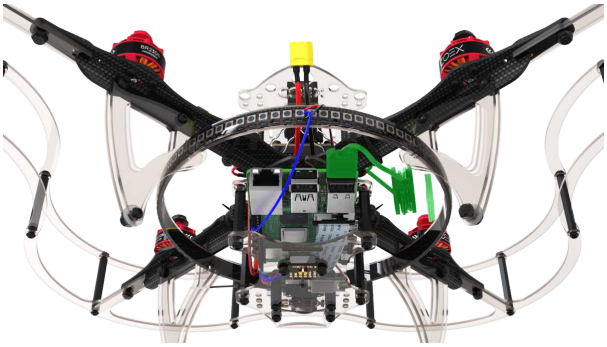
11. Подключите радиоприемник к полетному контроллеру используя разъем *RCIN*.



12. Подключите шлейф камеры к Raspberry Pi.



13. Подключите полетный контроллер к Raspberry Pi с помощью USB-кабеля.

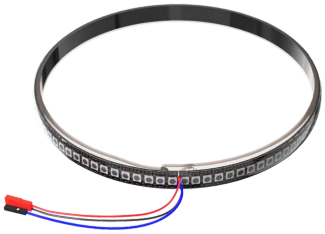


Установка LED ленты и ножек

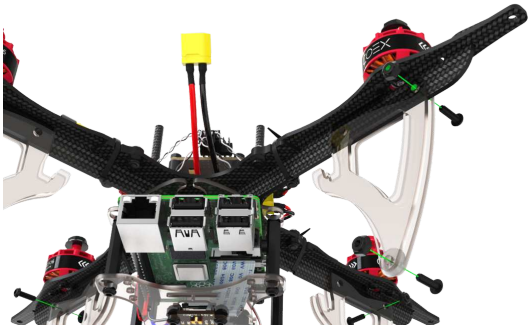
1. Соберите обруч для светодиодной ленты, объединив замок на концах.



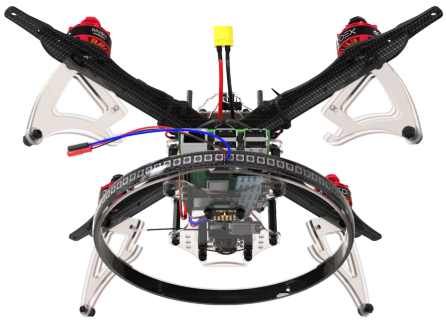
2. Наклейте светодиодную ленту на обруч, для большей крепкости притяните ее с помощью 3-4 хомутов.



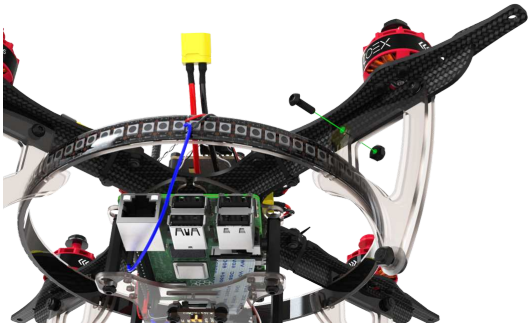
3. Установите ножки на пластину жесткости с помощью самоконтрящихся гаек и винтов M3x10 используя только крайние монтажные отверстия. Снизу, между пластинами ножек установите демпферное силиконовое колечко.



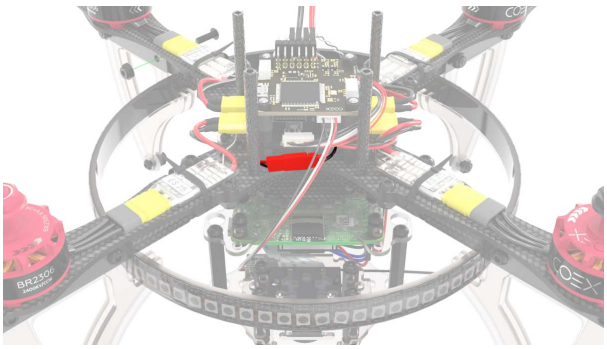
4. Отогните ножки назад и в специальный паз на них установите обруч со светодиодной лентой, таким образом, чтобы кабели подключения выходили с хвостовой стороны коптера.



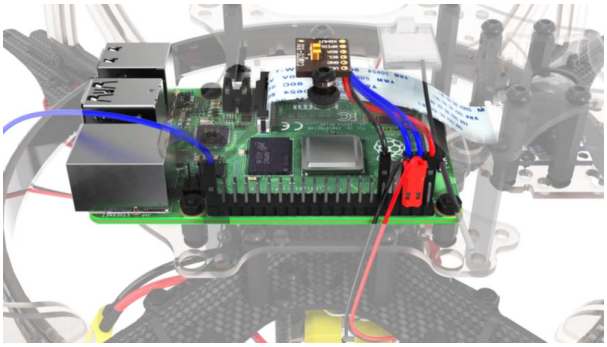
5. Закрепите ножки с помощью самоконтрящихся гаек и винтов M3x10.



6. Подключите питание светодиодной ленты (красный, черный кабели) в короткий разъем JST на плате распределения питания.

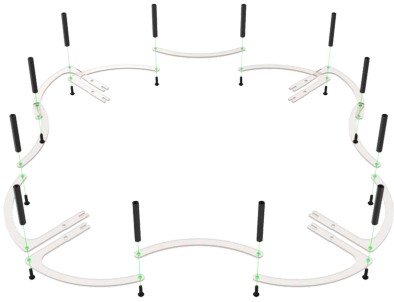


7. Подключите сигнальный выход светодиодной ленты (белый кабель) в Raspberry Ri, к пину *GPIO21*.

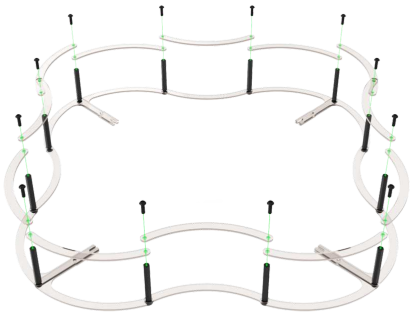


Установка защиты

1. Соберите нижний уровень защиты с помощью стоек 40 мм и винтов M3x10.



2. Соберите верхний уровень защиты с помощью винтов M3x10.



3. Установите монтажную деку и закрепите ее винтами M3x8.



4. Установите защиту и закрепите на лучах с помощью самоконтрящихся гаек и винтов M3x10.



Дрон собран, далее произведите настройку.

Сборка Клевера 4.2 WorldSkills

Габаритный чертеж – clover-4.2-ws.pdf.

Размер крепежа

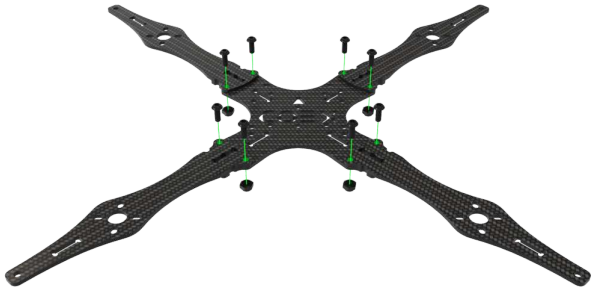
> Во время сборки используются винты и стойки различных размеров, использование крепежа не соответствующего размера может повредить коптер.

	Винт M3x10		Стойка алюминиевая 40 мм
	Винт M3x8		Стойка алюминиевая 15 мм
	Винт M3x5		Стойка нейлоновая 40 мм
	Саморез M2x5		Стойка нейлоновая 30 мм
	Гайка M3 (самоконтрящаяся)		Стойка нейлоновая 20 мм
	Гайка M3 (нейлоновая)		Стойка нейлоновая 15 мм
	Стойка демпферная		Стойка нейлоновая 6 мм

Сборка рамы

1. Совместите 4 луча с центральной декой, зафиксируйте их при помощи винтов M3x8 и самоконтрящихся гаек.





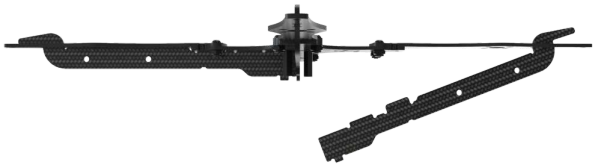
2. На центральные отверстия в главной деке установите 2 стойки 15 мм и закрепите их с помощью винтов M3x8.



3. Установите крючок пластины жесткости в паз на луче.



4. Прижмите пластины жесткости к главной деке.



5. Стяните пластины жесткости с помощью малой карбоновой деки.

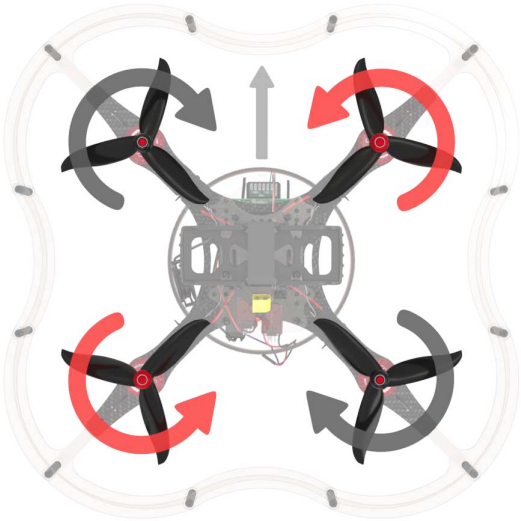


6. Установите 4 нейлоновые стойки 6мм и закрепите их с помощью винтов M3x5.



Установка моторов

1. При установке моторов обратите внимание на схему вращения моторов. Маркировка вращения на моторах должна совпадать со схемой вращения.



2. Установите мотор на соответствующие отверстия в луче с помощью **винтов M3x5**.



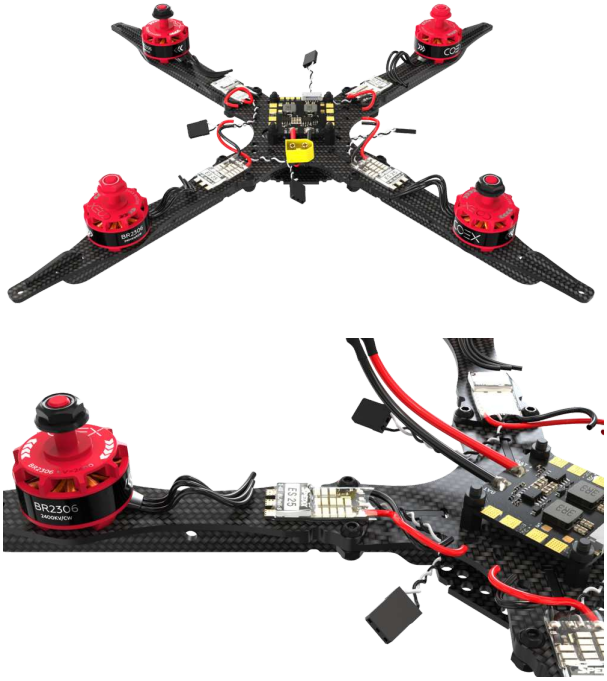
> Убедитесь, что моторы закреплены с помощью винтов M3x5, в противном случае может возникнуть короткое замыкание между обмотками.

Установка ESC и PDB

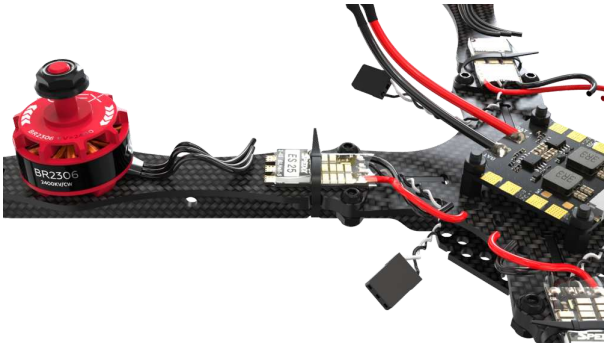
1. На заранее закрепленные стойки установите плату распределения питания (PDB), она должна быть установлена таким образом, чтобы силовой кабель питания был направлен в сторону хвоста коптера.



2. Установите регуляторы оборотов (ESC) в соответствующие места на луче.

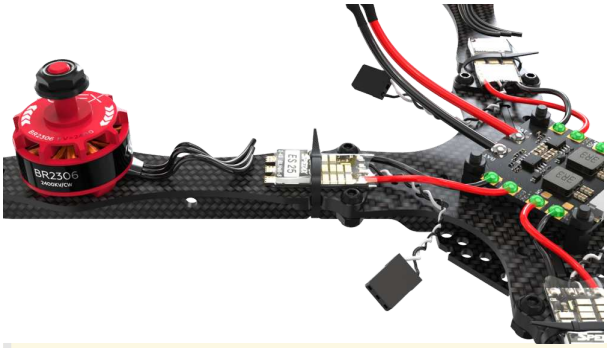


3. Притяните регуляторы оборотов (ESC) хомутами.



4. Отмерьте необходимое количество силового провода регуляторов оборотов(ESC), и обрежьте лишнее.
5. Зачистите и залудите обрезанные провода.
6. Залудите контактные площадки на плате распределения питания.
7. Припаяйте силовые провода регуляторов оборотов к плате распределения питания.



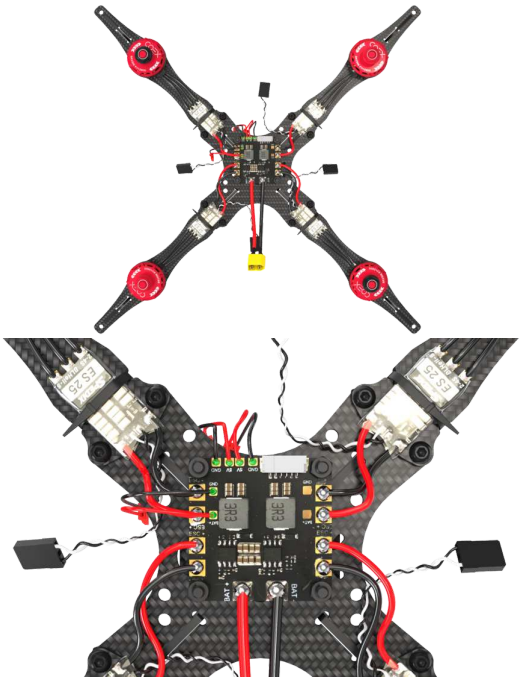


> Будьте внимательны к подписям контактов на плате. Красный провод должен идти к площадке с подписью +, а черный к подписи -.

8. Обрежьте лишний фазный кабель идущий от моторов.
9. Зачистите и залудите фазные кабели.
10. Залудите контактные площадки регуляторов оборотов.
11. Припаяйте фазные кабели к контактным площадкам регуляторов в любом порядке.



12. Припаяйте 3 разъема JST мама к 2ум площадкам *5V* и площадке *bat+*.



Установка полетного контроллера

Набор "Клевер 4" позволяет установить различные полетные контроллеры, к примеру, COEX Pix и Pixracer.

> При установке полетного контроллера обратите внимание на ориентацию платы. Если Вы установите COEX Pix серворазъемами назад (как на изображениях в инструкции) то впоследствии при настройке полетного контроллера в *Autopilot Orientation* необходимо будет указать значение `ROTATION_ROLL_180_YAW_90`, иначе полетный контроллер будет некорректно воспринимать наклоны и повороты коптера. Для полетного контроллера Pixracer это не требуется.

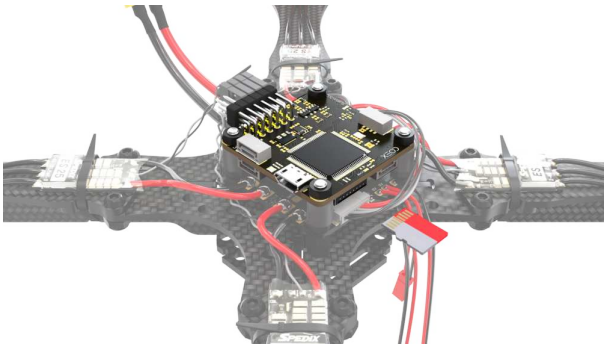
COEX Pix

> Перед установкой демпферных стоек, накрутите 2 слоя нейлоновых гаек, для более прочного крепления или откусите лишнюю резьбу с помощью бокорезов.

1. Закрепите плату распределения питания с помощью нейлоновых гаек, сверху установите демпферные стойки.
2. Установите полетный контроллер и закрепите нейлоновыми гайками.



3. Вставьте флеш-карту для записи логов в полетный контроллер.



Pixracer

1. Закрепите плату распределения питания с помощью нейлоновых стоек 6мм.
2. Установите малую крепежную деку и закрепите ее с помощью нейлоновых гаек.



3. Склейте 3–4 слоя двустороннего скотча, приклейте его в центр малой деки и установите сверху Pixracer.

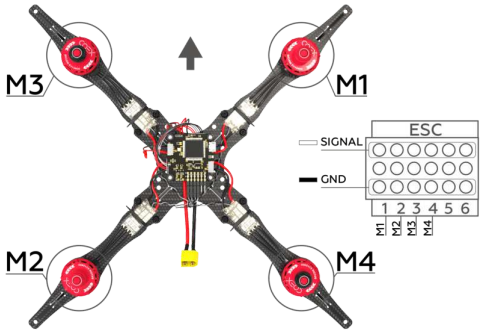


4. Вставьте флеш-карту для записи логов в полетный контроллер.

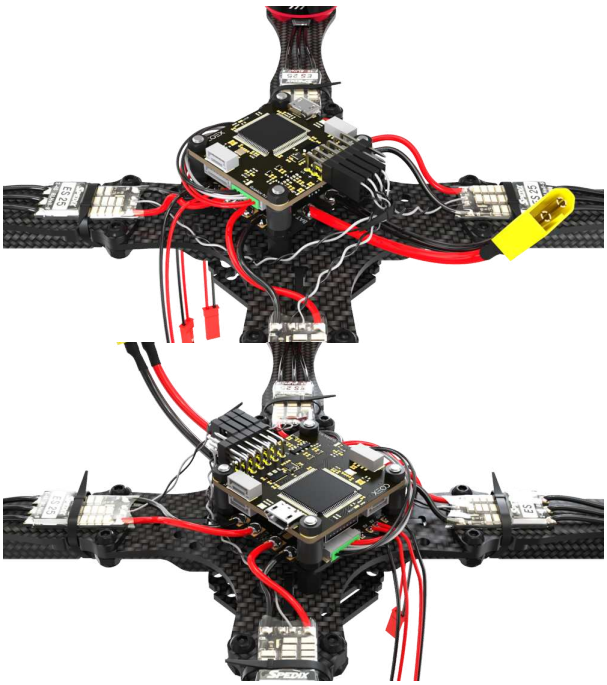


Подключение полетного контроллера

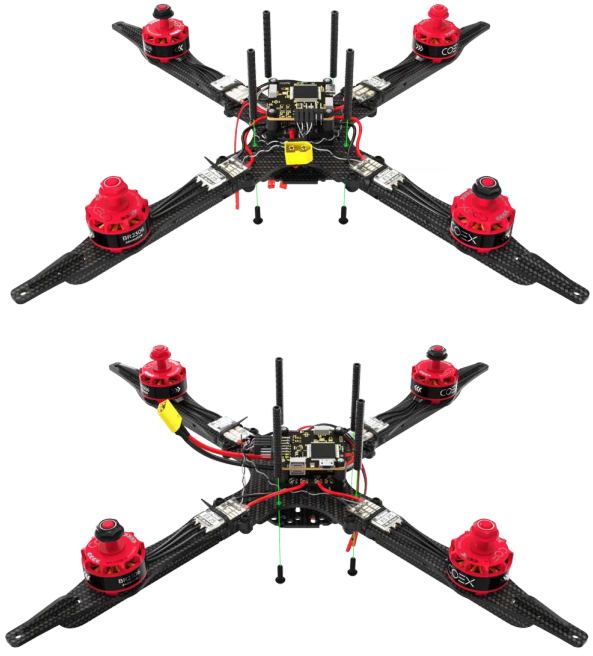
1. Подключите регуляторы оборотов к полетному контроллеру в соответствии со схемой.



2. Подключите кабель питания к плате распределения питания (PDB) и соответствующему разъему на полетном контроллере.

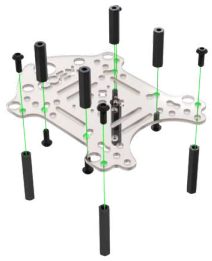


3. Установите алюминиевые стойки 40 мм на винты M3x10.



Установка Raspberry Pi

1. На монтажную деку установите стойки 20 мм и 40 мм, закрепите их с помощью винтов M3x8.

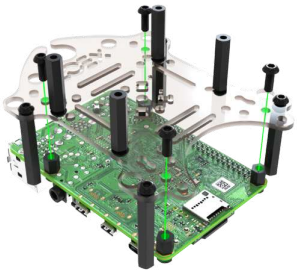


2. Нарежьте резьбу M3 в крепежных отверстиях Raspberry Pi с помощью болта M3x10.
3. Вкрутите в плату Raspberry Pi стойки 6мм, при необходимости закрепите их нейлоновыми гайками.





4. Установите Raspberry Pi на монтажную деку, закрепив ее болтами M3x6.



5. Установите собранный модуль в соответствующие пазы основной деке дрона.



6. Подключите разъем 5V JST в соответствующие пины питания Raspberry Pi.

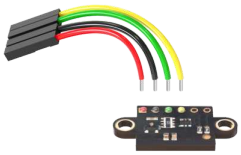


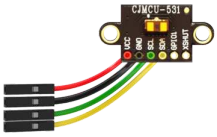
7. Возьмите 4 провода Dupont, обрежьте 5–7 см кабеля и припаяйте к соответствующим пинам дальномера.





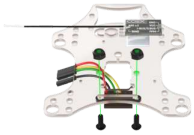
Провод	Пин дальномера
Красный	5v
Черный	GND
Желтый	SDA
Зеленый	SCL



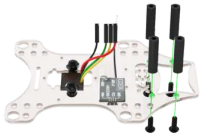


8. Установите дальномер на деку захвата и Приклейте радиоприемник на 3M скотч.

> Устанавливайте дальномер таки образом, чтобы гайки не прислонялись к плате напрямую. При такой установке если большая вероятность повредить элементы платы.



9. Установите 4 нейлоновые стойки 20мм и зафиксируйте их болтами M3x8.



10. На малую монтажную деку установите камеру и зафиксируйте ее двумя короткими саморезами.

> Если закрепить камеру в верхнем правом углу и шляпка самореза будет касаться элемента на камере, камера не будет работать.



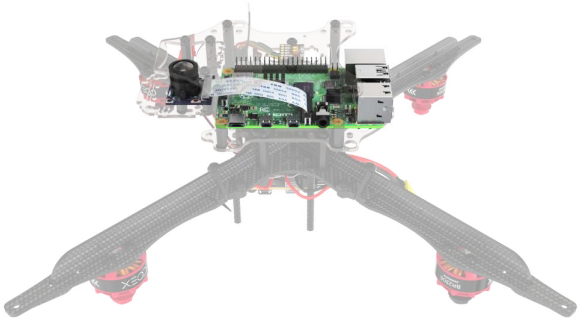
11. Установите малую монтажную деку с камерой на стойки и зафиксируйте с помощью болтов M3x8.



12. Собранный модуль установите над модулем Raspberry Pi и зафиксируйте болтами M3x8.



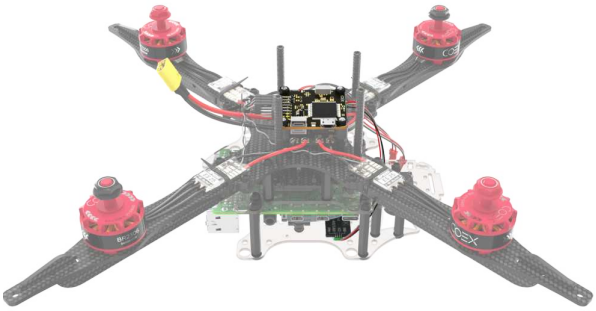
13. Соедините камеру и Raspberry Pi с помощью шлейфа.



14. Подсоедините дальномер к Raspberry Pi в соответствующие пины.



15. Соедините радиоприемник и полетный контроллер с помощью 4-х пинового кабеля.

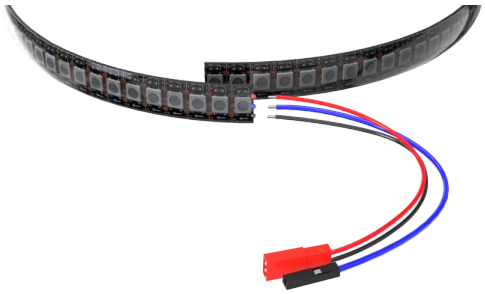


Установка LED ленты и ножек

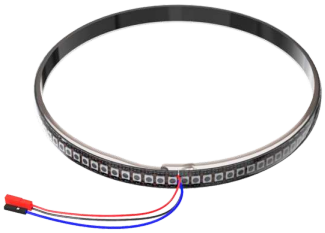
1. Соберите обруч для светодиодной ленты, объединив замок на концах.



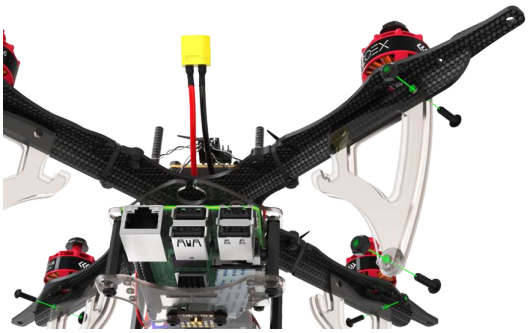
2. Припаяйте JST-папа к площадке питания и Dupont-мама к сигнальной площадке.



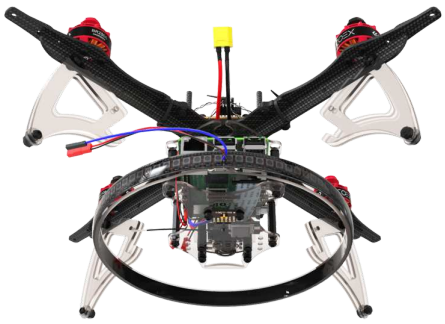
3. Наклейте светодиодную ленту на обруч, для большей крепкости притяните ее с помощью 3–4 хомутов.



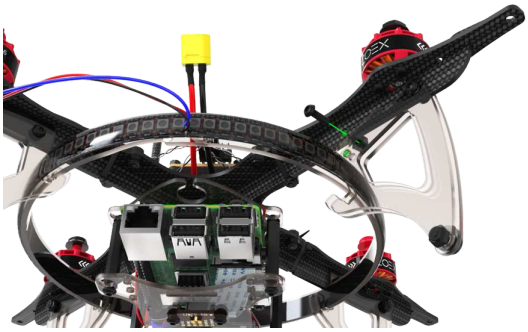
4. Установите ножки на пластину жесткости с помощью самоконтрящихся гаек и винтов M3x10 используя только крайние монтажные отверстия. Снизу, между пластинами ножек, установите демпферное силиконовое колечко.



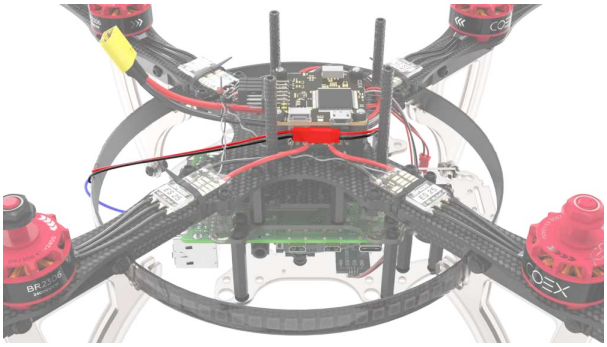
5. Отогните ножки назад и в специальный паз на них установите обруч со светодиодной лентой, таким образом, чтобы кабели подключения выходили с хвостовой стороны коптера.



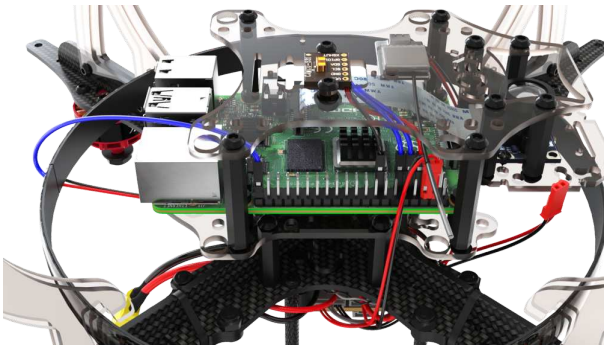
6. Закрепите ножки с помощью самоконтрящихся гаек и винтов M3x10.



7. Подключите питание светодиодной ленты в разъем JST 5V на плате распределения питания.



8. Подключите сигнальный выход светодиодной ленты в Raspberry Ri, к пину *GPIO21*.

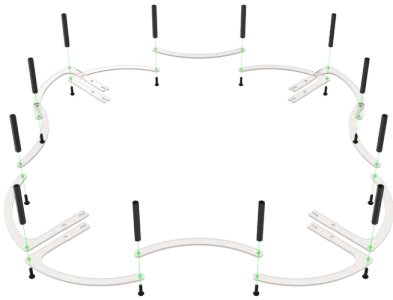


9. Установите монтажную деку и закрепите ее винтами M3x8.

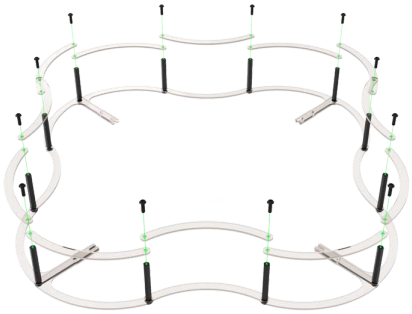


Установка защиты

1. Соберите нижний уровень защиты с помощью стоек 40 мм и винтов M3x10.



2. Соберите верхний уровень защиты с помощью винтов M3x10.



3. Установите защиту и закрепите на лучах с помощью самоконтрящихся гаек и винтов M3x10.



4. Подключите полетный контролер к Raspberry Pi с помощью USB к кабеля.



5. Установите ремешок для крепления АКБ.



Дрон собран, далее произведите настройку.

Сборка Клевера 4.1

Особенность набора. Паечный набор с версией рамы 4.0, новыми регуляторами spedix e25 (они аналогичны старым dys 30a в плане работы и настройки), PDB и полётным контроллером производства COEX, радиоприёмником FS-A8S, новыми версиями пластин и деталей из поликарбоната. Использование новой COEX PDB избавляет от необходимости пользоваться дополнительными стабилизаторами на 5V, поэтому 2 платы BEC больше не включены в набор.

Особенность сборки. Сборка рамы, крепление моторов и PDB аналогичны сборке из старой версии. Отличается крепление Raspberry Pi к центральной деке. Направление вращения моторов во время сборки проверить нельзя из-за особенностей радиоприёмника, однако можно однозначно запаять провода моторов к регуляторам. Остальная сборка аналогична сборке нового беспаечного набора.



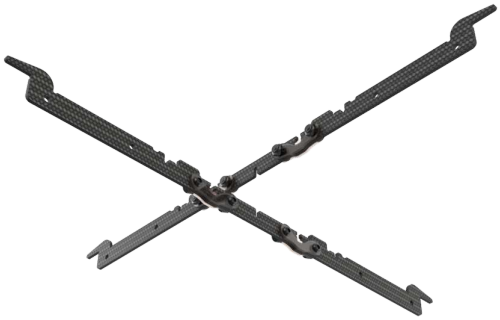
Сборка основы для рамы

> Обратите внимание! Далее в инструкции некоторые изображения взяты из сборки старого паечного или нового беспаечного набора, если в текущей сборке требуются аналогичные действия.

1. В случае наличия, закрепите рем-накладки на пластинах жесткости, иначе продолжайте без них.



2. Совместите 2 карбоновые пластины жесткости, используя центральные пазы.





3. Используя пазы, установите сверху карбоновую центральную деку.





4. Стяните конструкцию с помощью винтов M3x8 и стальных гаек с нейлоновой вставкой, установленных в пазах пластин.

Установка моторов

1. Распакуйте моторы.
2. Используя бокорезы, укоротите провода на моторах:
    - Обрежьте половину длины (оставив 30 мм).
    - Зачистите (снимите 5 мм изоляции с конца провода, не повредив медные жилы).



    - Скрутите медные жилы.
    - Залудите провода, используя пинцет.
3. Установите мотор на луч.
4. Прикрепите мотор к лучу винтами M3x5, используя шестигранный ключ или отвёртку.



Повторите эти действия для остальных моторов.

Сборка рамы

1. Установите 4 луча с моторами на базу рамы, используя пазы, согласно схеме вращения моторов.



> Для правильной установки моторов обратите внимание на цвета гаек. Моторы с красными гайками следует установить на передний правый и задний левый лучи, с чёрными - на передний левый и задний правый.

2. Зафиксируйте лучи на раме, используя 8 винтов M3x8 и 8 стальных гаек.

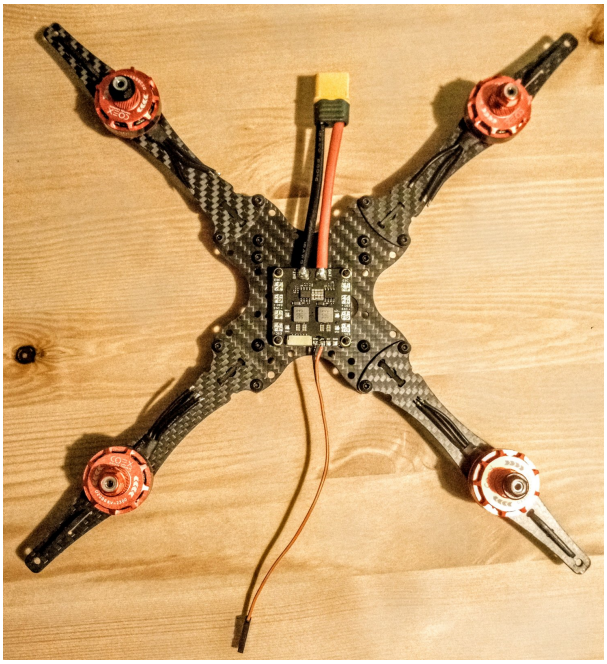


Монтаж PDB

1. Установите 4 стойки "папа-мама" 6 мм на центральную деку винтами M3x6.



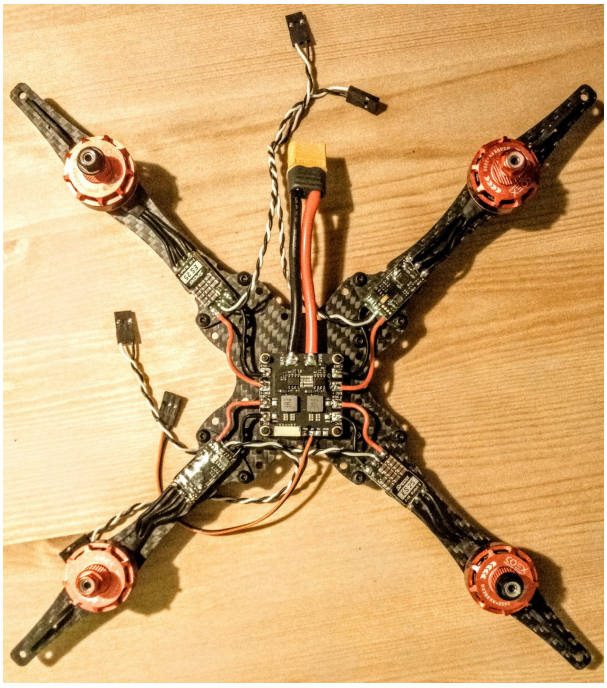
2. Установите COEX PDB на стойки. Припаяйте к площадкам GND и 5V провод для питания Raspberry Pi. Его можно сделать из 3 pin провода для телеметрии, который есть в комплекте.



3. Разъём для подключения аккумулятора должен быть направлен к задней части рамы.

Пайка регуляторов и BEC

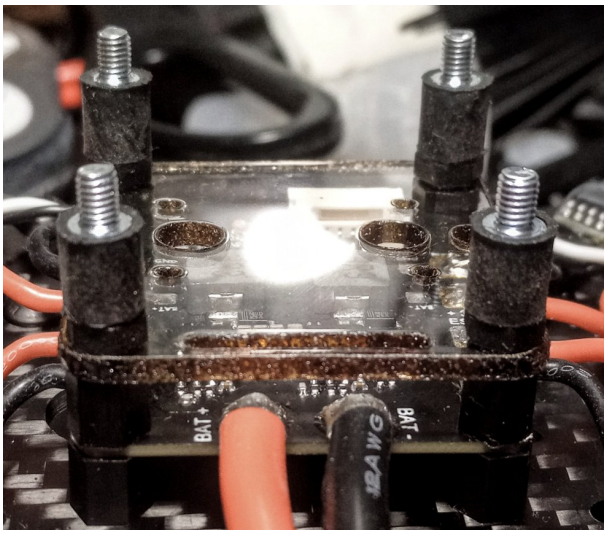
1. Расположите регуляторы *логотипом вверх* около моторов, которые вращаются *по часовой стрелке*. Расположите регуляторы *логотипом вниз* около моторов, которые вращаются *против часовой* стрелки.
2. Припаяйте фазные провода моторов к регуляторам, чтобы провода не пересекались и проходили к контактам регулятора по кратчайшему расстоянию.
3. Припаяйте силовые провода регуляторов к контактным площадкам платы (**красный** к **«+»**, **черный** к **«-»**).



4. С помощью мультиметра проверьте, что в цепи нет короткого замыкания.

Установка пластины для полётного контроллера COEX Pix

1. Установите 4 стойки "папа-мама" 6 мм на PDB, затем установите поликарбонатную пластину, затем закрутите её нейлоновыми гайками, затем установите резиновые стойки 6мм.

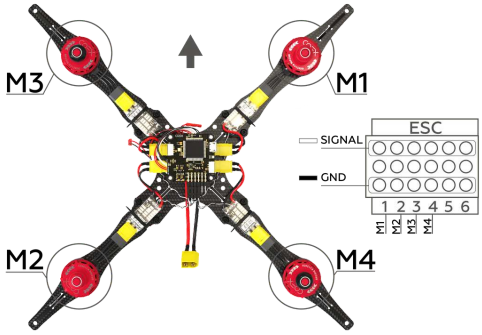


2. Закрепите *COEX Pix* на резиновых стойках с помощью нейлоновых гаек.



Подключение полетного контроллера

1. Подключите регуляторы оборотов к полетному контроллеру в соответствии со схемой.



2. Подключите кабель питания к плате распределения питания и соответствующему разъему на полетном контроллере.

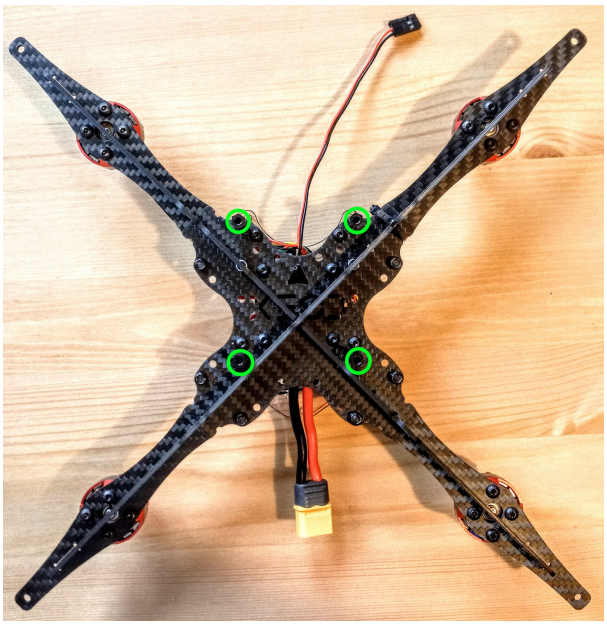


3. Установите алюминиевые стойки 40мм на болты M3x12.



Установка Raspberry Pi, камеры, дальномера и радиоприёмника

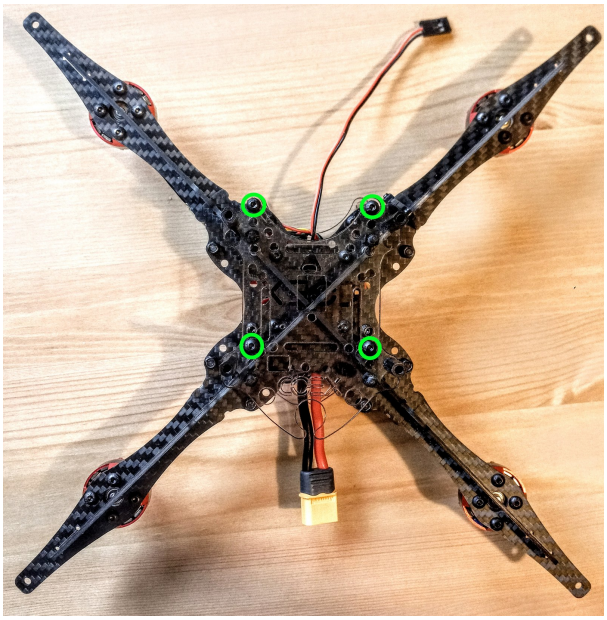
1. На нижнюю грань основной деки установите 4 стойки 15мм, закрепите их с помощью болтов M3x8.



2. На монтажную деку установите 4 стойки 6мм и 4 стойки 30мм, закрепите их с помощью болтов M3x6 и M3x12.



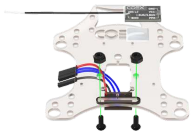
3. Установите собранную монтажную деку на основную и закрепите с помощью болтов M3x8.



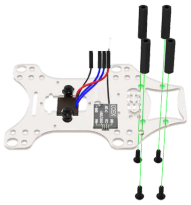
4. Установите плату *Raspberry Pi* и зафиксируйте с помощью нейлоновых гаек.



5. На деку захвата установите дальномер с помощью самоконтрящихся гаек и болтов M3x8, и приклейте радиоприемник `FS-A8S` с помощью двустороннего скотча. Подключение приёмника к полётному контроллеру и его сопряжение с пультом описано статье Работа с приёмником Flysky FS-A8S.



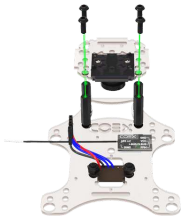
6. Установите 4 стойки 20мм и закрепите их с помощью болтов M3x8.



7. На малую монтажную деку установите камеру и зафиксируйте ее с помощью 2ух маленьких саморезов, в верхнем левом и нижнем правом углах.



8. Установите модуль камеры на деку захвата и закрепите с помощью болтов M3x8.



9. Установите собранную деку захвата и зафиксируйте с помощью болтов M3x8.



10. Подключите к *Raspberry Pi* дальномер и кабель питания.



11. Подключите шлейф камеры к *Raspberry Pi*.



Установка LED ленты

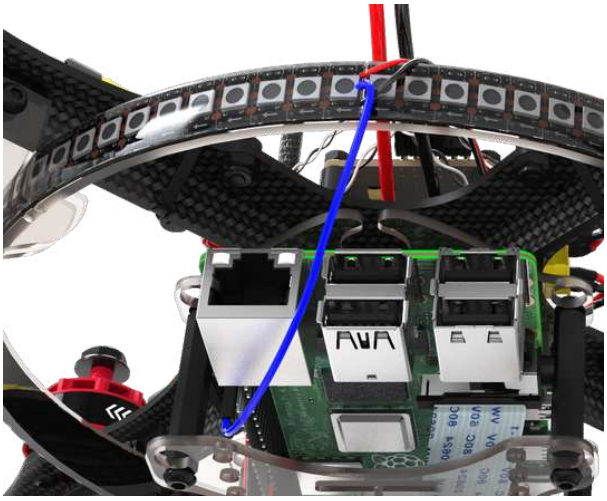
1. Соберите обруч для светодиодной ленты, объединив замок на концах.



2. Наклейте светодиодную ленту на обруч, для большей прочности притяните ее с помощью 3-4 хомутов. Припаяйте к ленте 2 силовых провода и один сигнальный провод для подключения к Raspberry Pi.



3. Установите обруч на раму. Припаяйте питание светодиодной ленты (красный, черный кабели) к свободным площадкам 5V и GND соответственно на COEX PDB. Подключите сигнальный выход светодиодной ленты(белый кабель) в *Raspberry Ri*, к пину *GPIO 21*.

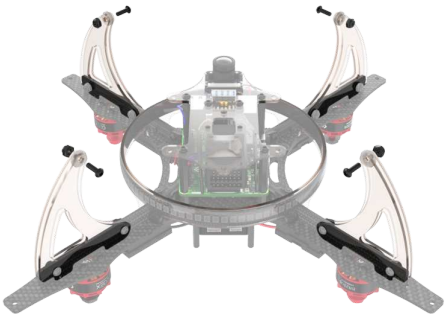


Монтаж ножек

1. Установите 8 ножек с помощью винтов M3x10 и стальных гаек.

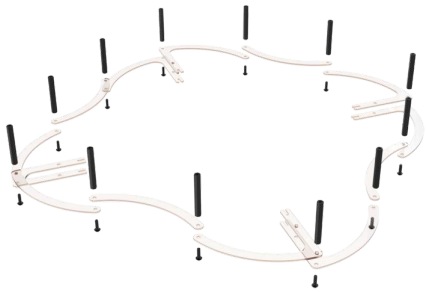


2. Установите демпфирующие прокладки на ножки с помощью винтов M3x10 и стальных гаек.

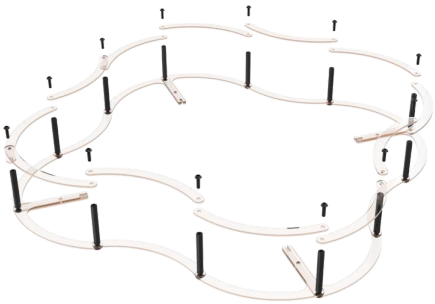


Установка защиты

1. Соберите нижнюю часть защиты, используя 12 винтов M3x10 и 12 нейлоновых стоек 40 мм.



2. Установите верхнюю часть, используя 12 винтов M3x10.



3. Установите монтажную деку и закрепите ее болтами M3x8.

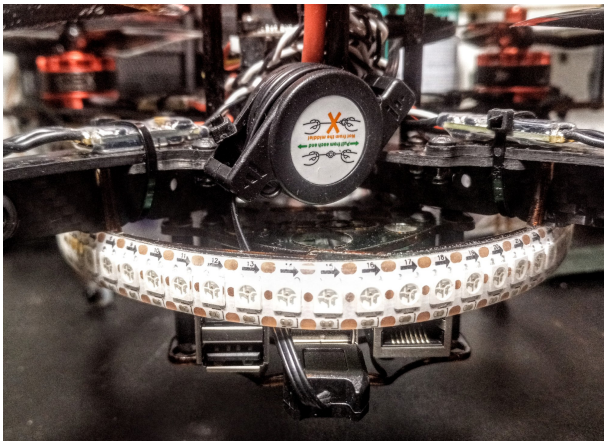


4. Установите защиту на коптер, с помощью 4 винтов M3x10 и стальных гаек.



Установка пропеллеров и подготовка к полёту

1. Подключите полётный контроллер по micro USB проводу к Raspberry Pi. Закрепите улитку с проводом таким образом, чтобы она находилась ниже уровня установки пропеллеров, чтобы пропеллеры не повредили провод во время полёта. Для крепления улитки можно воспользоваться стяжкой.



2. Произведите настройку компонентов квадрокоптера, используя раздел "Настройка".

> Установка пропеллеров должна производиться **только после окончательной настройки коптера**, непосредственно перед полетом.

3. Установите 4 пропеллера, согласно схеме вращения. При установке пропеллеров АКБ должна быть отключена.

   При установке будьте внимательны, чтобы пропеллер не был перевернут. На лицевой стороне пропеллера имеется маркировка его характеристик, а также направление вращения, которое должно совпадать с направлением вращения моторов.





Установка АКБ

> Убедитесь, что все провода спрятаны и движению пропеллеров ничего не мешает.

Проверьте сборку квадрокоптера:

- Балансировочный разъем АКБ должен быть спрятан под утягивающим ремешком.
- Регуляторы должны быть зафиксированы хомутами.
- Все провода, идущие от PDB и полетного контроллера, должны быть зафиксированы липучкой или обмотаны вокруг алюминиевых стоек.
- Пропеллеры установлены правильной стороной и соответствуют направлению вращения моторов.



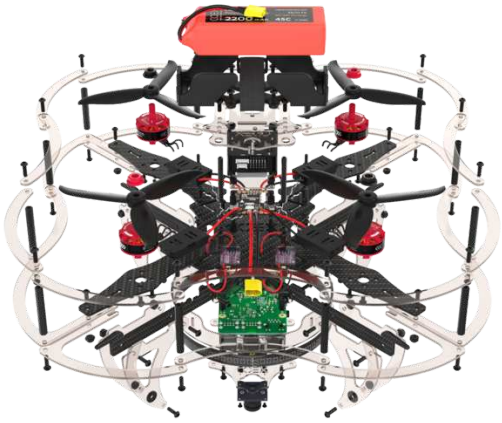
Обязательно установите и настройте индикатор напряжения перед полетом, чтобы не переразрядить аккумулятор. Для настройки индикатора используйте кнопку, расположенную в его основании. Отображаемые цифры во время настройки обозначают минимально возможное напряжение в каждой ячейке аккумулятора, рекомендуемое значение **3.5**.

> Звуковая индикация означает, что ваш аккумулятор разряжен и его нужно зарядить.



Дрон готов к полету!

Сборка Клевера 4



Сборка основы для рамы

> Для увеличения прочности рамы вы можете распечатать на 3D принтере или нарезать на лазерном резаке рем-накладки.

1. В случае наличия, закрепите рем-накладки на пластинах жесткости, иначе продолжайте без них.



2. Совместите 2 карбоновые пластины жесткости, используя центральные пазы.



3. Используя пазы, установите сверху карбоновую центральную деку.



4. Стяните конструкцию с помощью винтов M3x8 и стальных гаек с нейлоновой вставкой, установленных в пазах пластин.

Установка моторов

1. Распакуйте моторы.
2. Используя бокорезы, укоротите провода на моторах:
    - Обрежьте половину длины (оставив 30 мм).
    - Зачистите (снимите 5 мм изоляции с конца провода, не повредив медные жилы).



- Скрутите медные жилы.
- Залудите провода, используя пинцет.

3. Установите мотор на луч.
4. Прикрепите мотор к лучу винтами M3x5, используя шестигранный ключ или отвёртку.



Повторите эти действия для остальных моторов.

Сборка рамы

1. Установите 4 луча с моторами на базу рамы, используя пазы, согласно схеме вращения моторов.



> Для правильной установки моторов обратите внимание на цвета гаек. Моторы с красными гайками следует установить на передний правый и задний левый лучи, с чёрными - на передний левый и задний правый.

2. Зафиксируйте лучи на раме, используя 8 винтов M3x8 и 6 стальных гаек, а также 2 стойки "мама-мама" 15 мм.



Подготовка платы распределения питания

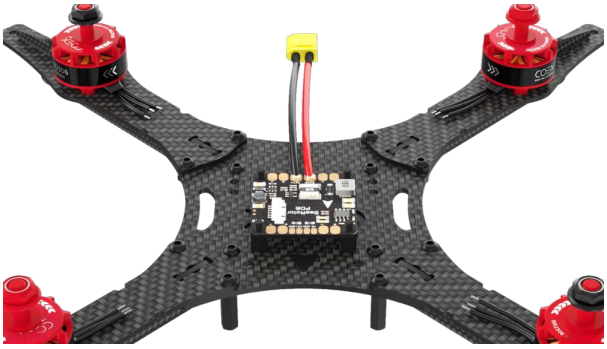
1. Залудите контактные площадки платы питания.
2. С помощью мультиметра проверьте отсутствие короткого замыкания (прозвонить):
   - Установите мультиметр в режим прозвонки.
   - Проверьте работу мультиметра путем замыкания щупов между собой. При корректной работе прибор издаст характерный звук.
   - Попарно один щуп прикладывается к контакту **«+»**, а второй к **«-»/GND**. Если в цепи есть короткое замыкание, издается звук.

Монтаж PDB

1. Установите 4 стойки "папа-мама" 6 мм на центральную деку винтами M3x6.



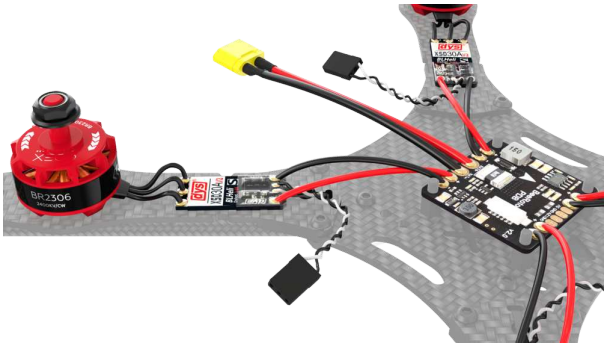
2. Установите PDB на стойки.



3. Стрелки на PDB и центральной деке должны быть направлены в одну сторону.

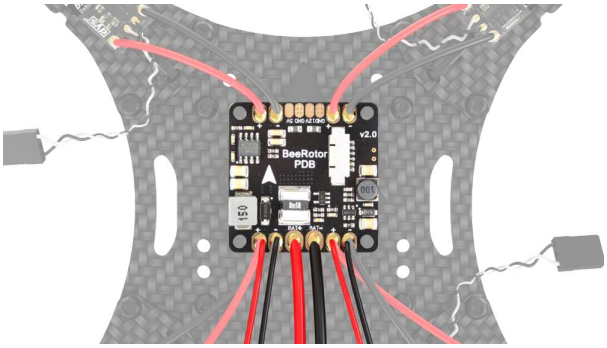
Пайка регуляторов и BEC

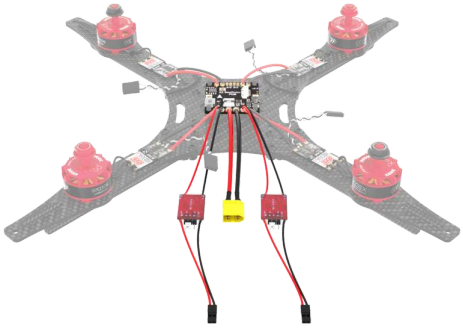
1. Припаяйте фазные провода моторов к регуляторам.
2. Припаяйте силовые провода регуляторов к контактным площадкам платы (**красный** к **«+»**, **черный** к **«-»**).





3. Припаяйте силовые провода каждого BEC к контактным площадкам одного из регуляторов (**красный** к **«+»**, **черный** к **«-»**).





4. С помощью мультиметра проверьте, что в цепи нет короткого замыкания.

Перевод пульта в режим PWM

Включите пульт с помощью слайдера **POWER**. Если пульт заблокирован, необходимо перевести все стики в начальное положение:

1. Левый стик в **центральной нижней позиции**.
2. Правый стик в **центре**.
3. Переключатели A, B, C, D в положение **"от себя"**.



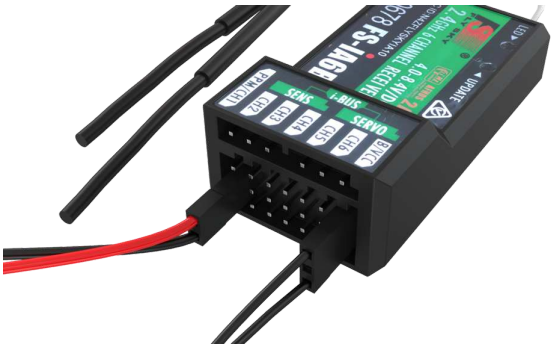
Убедитесь, что PPM в меню RX Setup отключен:

1. Убедитесь, что питание дрона выключено.
2. Для входа в меню удерживайте нажатой кнопку "OK".
3. Кнопками Up/Down выбираем меню "System setup", кнопкой "OK" подтвердите выбор.
4. Выберите "RX Setup".
5. Выберите "Output mode".
6. Убедитесь, что в открывшемся меню выбран пункт "PWM".
7. Чтобы сохранить настройки, удерживайте нажатой кнопку "Cancel".

Сопряжение приёмника и пульта

1. Выключите пульт с помощью слайдера **POWER**.
2. Подключите радиоприемник к разъему BEC 5В. Чёрный провод подключите к одному из нижних пинов, красный - к одному из центральных.
3. Установите джампер на вход (B/VCC).
4. Подключите АКБ.
5. Светодиод на радиоприемнике должен мигать.





6. Зажмите кнопку **BIND KEY** на пульте.
7. Включите пульт (перещелкните **POWER**, не отпуская **BIND KEY**).

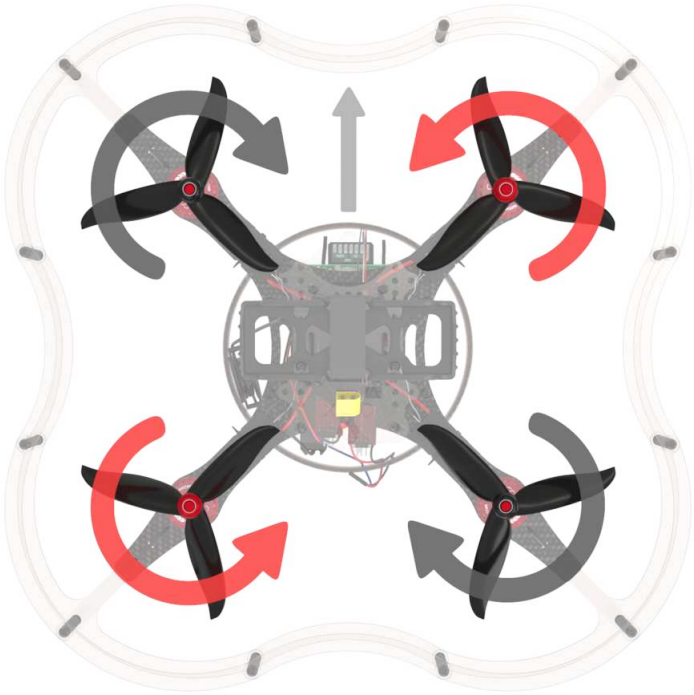


8. Ждите синхронизации.
9. Отсоедините джампер.

10. Светодиод на приемнике должен гореть непрерывно.

Проверка направления вращения моторов

Моторы с **красными** гайками должны вращаться **против** часовой стрелки, с **чёрными** - **по** часовой стрелке. Правильные направления вращения также указаны на самих моторах. Для проверки направления вращения можно использовать сервотестер или радиоприёмник с пультом.



1. Отключите АКБ и пульт.
2. Подключите сигнальный провод от ESC к выходу CH3 на приёмнике. Белый провод должен подходить к верхнему пину, чёрный - к нижнему.
3. Включите пульт. Левый стик должен быть в нижнем положении.
4. Подключите АКБ.
5. Медленно поднимайте левый стик до тех пор, пока мотор не начнёт вращаться.

Если мотор вращается в неправильную сторону, поменяйте местами два любых фазных провода.

> Направление вращения также можно изменить программно. Процесс описан в статье про прошивку ESC.

Повторите процесс для каждого мотора.

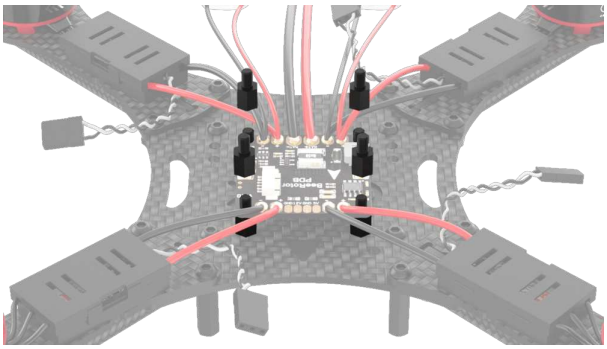
Перевод пульта в режим PPM

Полётный контроллер не может работать с пультом в режиме PWM, поэтому следует произвести перевод пульта в режим PPM.

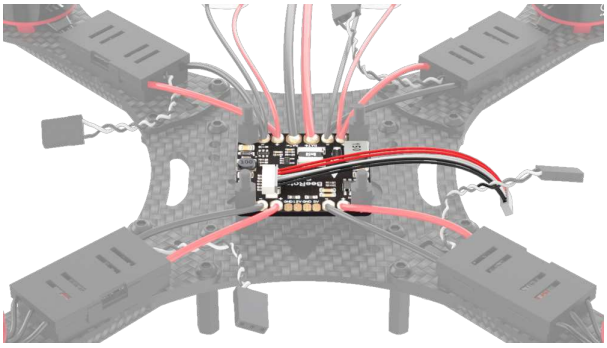
1. Убедитесь, что питание дрона выключено.
2. Для входа в меню удерживайте нажатой кнопку "OK".
3. Кнопками Up/Down выбираем меню "System setup", кнопкой "OK" подтвердите выбор.
4. Выберите "RX Setup".
5. Выберите "Output mode".
6. Убедитесь, что в открывшемся меню выбран пункт "PPM".
7. Чтобы сохранить настройки, удерживайте нажатой кнопку "Cancel".

Установка пластины для полётного контроллера

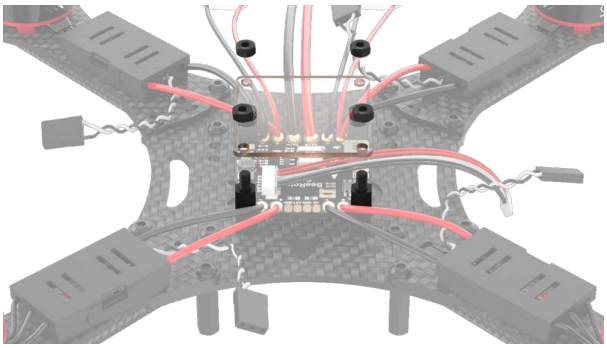
1. Установите 4 стойки "папа-мама" 6 мм на PDB.



2. Подключите шлейф питания к PDB.



3. Установите поликарбонатную пластину на стойки и зафиксируйте нейлоновыми гайками.

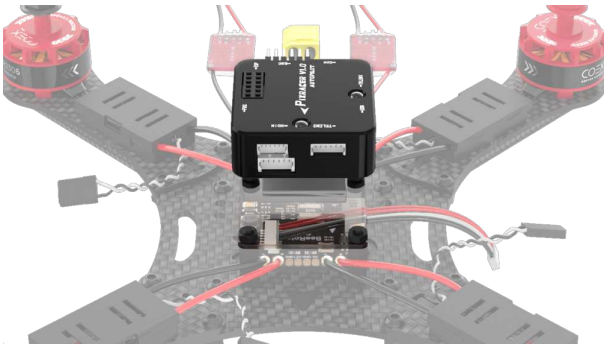


Установка полётного контроллера

1. Вставьте карту microSD в полётный контроллер.

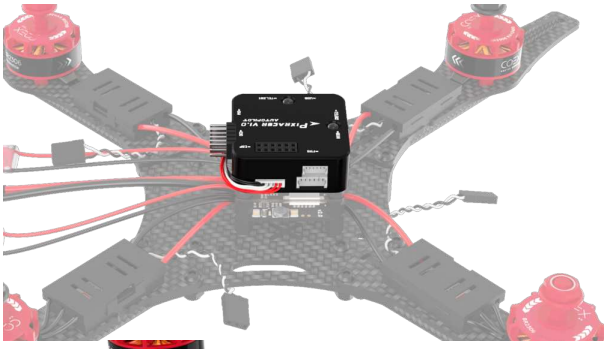


2. Установите полетный контроллер на пластину с помощью двухстороннего скотча.



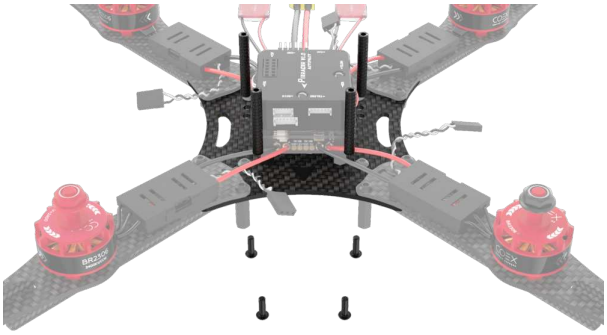
3. Стрелки на полетном контроллере и центральной деке должны быть направлены в одну сторону.

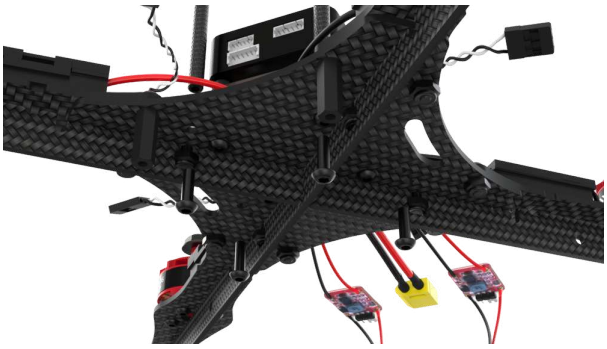
4. Подключите шлейф питания PDB к разъему *"POWER"* полетного контроллера, закрутив его в "косичку" для взаимной фиксации проводов.



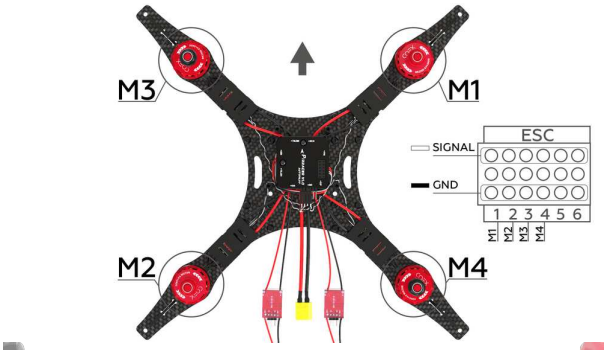


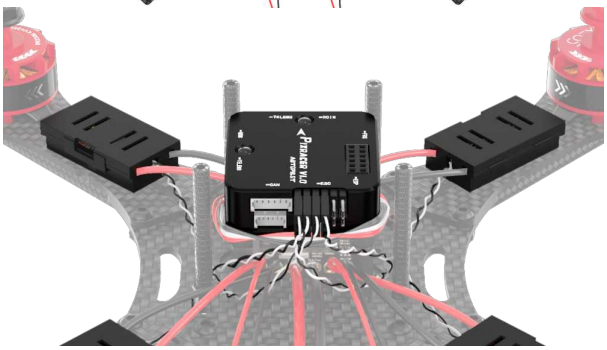
5. Установите 4 алюминиевые стойки 40 мм с помощью винтов M3x10.



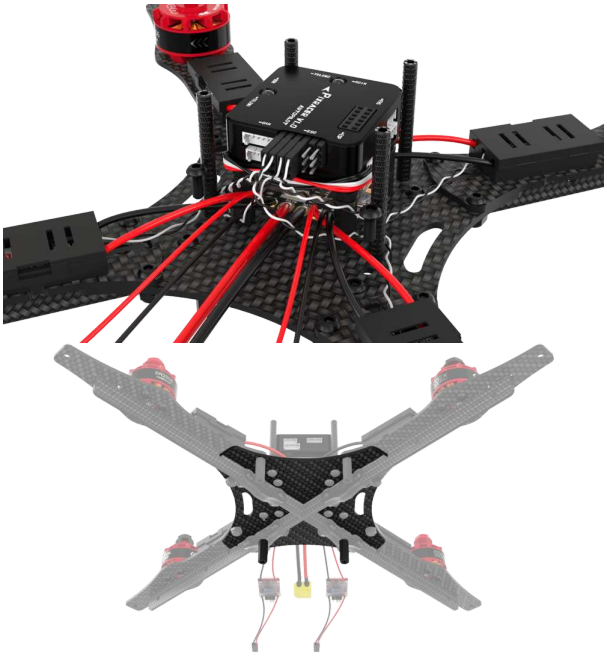


6. Подключите сигнальные провода регуляторов к полетному контроллеру следующим образом:





7. Установите 2 стойки "мама-мама" 15 мм на центральную деку с помощью винтов M3x8.



8. Другие 2 стойки были установлены ранее в разделе "Сборка рамы", п. 2.

Установка обруча для светодиодной ленты

1. Согните поликарбонатную заготовку в обруч и зафиксируйте ее концы в замке.
2. Установите обруч на раму, используя пазы.

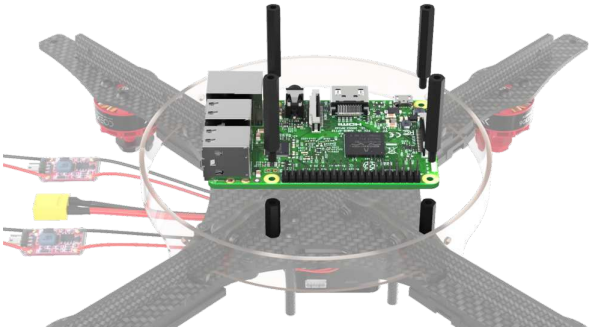


Установка Raspberry Pi

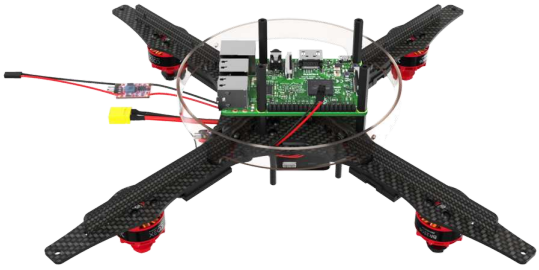
1. Вставьте карту microSD с записанным образом в Raspberry Pi



2. Установите плату Raspberry Pi на стойки, используя 4 стойки "папа-мама".



3. Протяните провода от BEC через паз в центральной раме.



4. Подключите провод питания от BEC к Raspberry, согласно схеме:



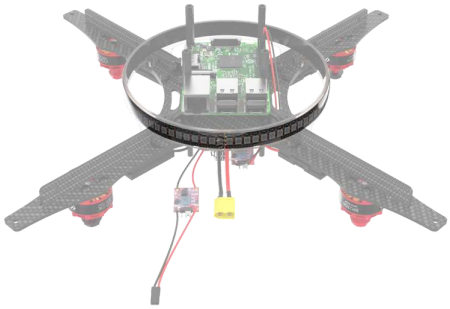
Установка светодиодной ленты на обруч

1. Проверьте наличие напаянных пинов на контактах ленты (при отсутствии - напаять).



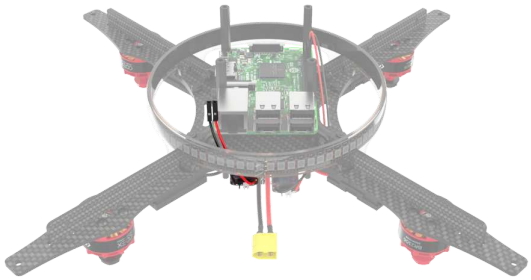


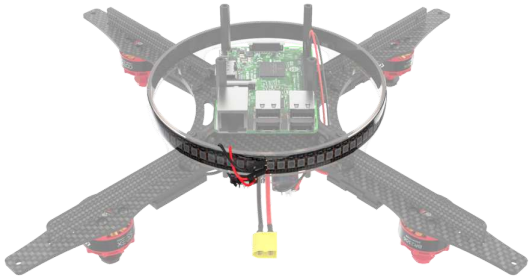
2. Установите светодиодную ленту на обруч (используя клеевой слой на ленте) так, чтобы контакты были в задней части коптера. Для дополнительной фиксации используйте **стяжки**.



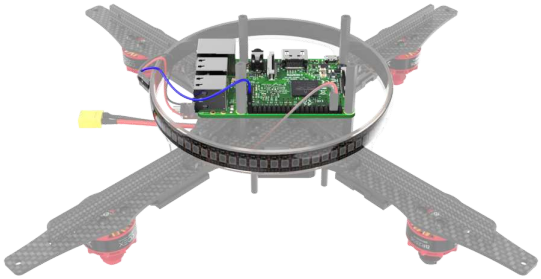
Подключение светодиодной ленты к Raspberry Pi

1. Питание для ленты берется от второго BEC. Подключите контакты *«-»* и *«+»* к *Ground* и *5v* на ленте соответственно.



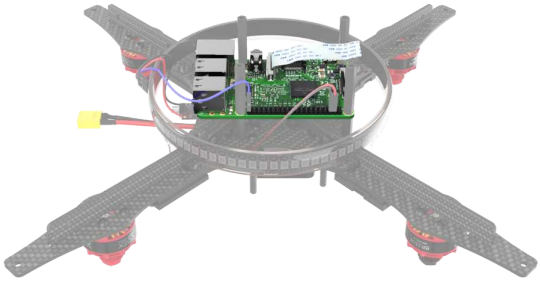


2. Подключите контакт *D* к GPIO-пину на Raspberry. Рекомендуется использовать пин GPIO21.



Установка шлейфа для камеры

1. Поднимите защелку.
2. Подключите шлейф.
3. Закройте защелку.



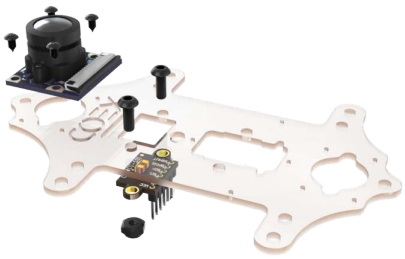
Установка оборудования на нижнюю монтажную деку

1. Подготовьте лазерный дальномер к монтажу, предварительно напаяв на него контакты.
2. Установите камеру на 4 самореза 2x5.

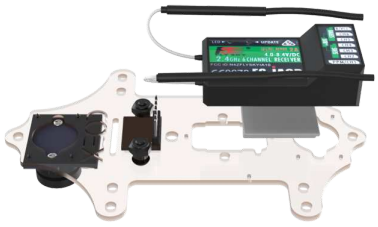
   > Убедитесь, что саморезы не касаются деталей на печатной плате камеры. В противном случае камера может не заработать.

3. Установить на деку лазерный дальномер с помощью 2 винтов M3x8 и стальных гаек.

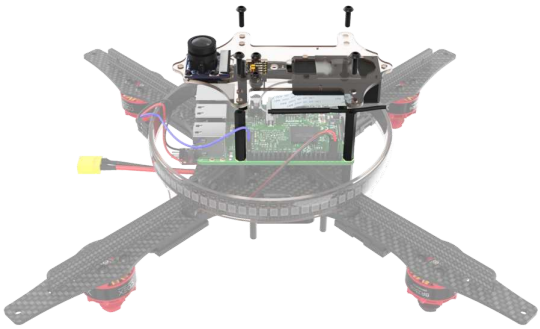




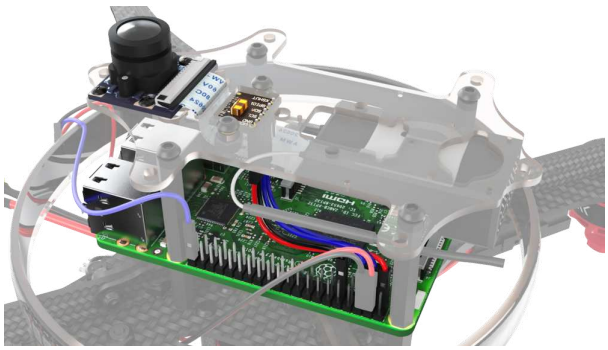
4. Установите приемник на нижнюю деку с помощью двухстороннего скотча.



5. Установите нижнюю деку с помощью 4 винтов M3x10.



6. Подключите шлейф к камере.
7. Подключите лазерный дальномер к Raspberry Pi с помощью проводов типа "мама-мама":
    - Разъем *VCC* к пину 1 (*3.3v*).
    - Разъем *GND* к пину 9 (*Ground*).
    - Разъем *SDA* к пину 3 (*GPIO02*).
    - Разъем *SCL* к пину 5 (*GPIO03*).

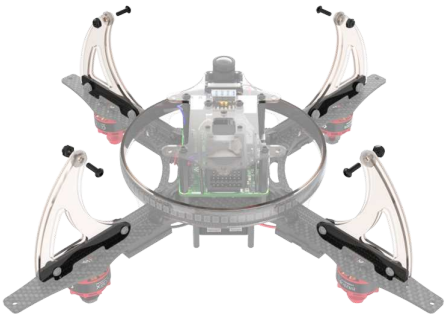


Монтаж ножек

1. Установите 8 ножек с помощью винтов M3x10 и стальных гаек.



2. Установите демпфирующие прокладки на ножки с помощью винтов M3x10 и стальных гаек.



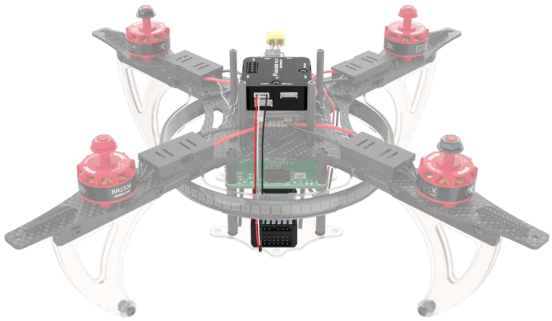
Подключение кабелей

1. Подключите кабель радиоприемника в *RCIN* разъем полетного контроллера.



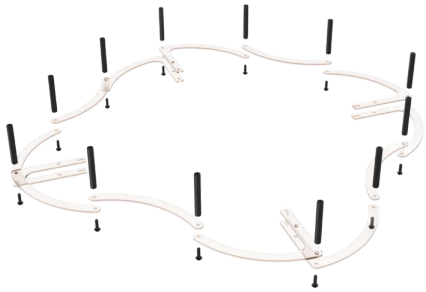
2. Подключите кабель к приемнику, соответственно изображению.



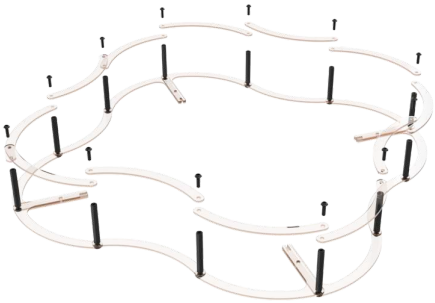


Сборка защиты винтов

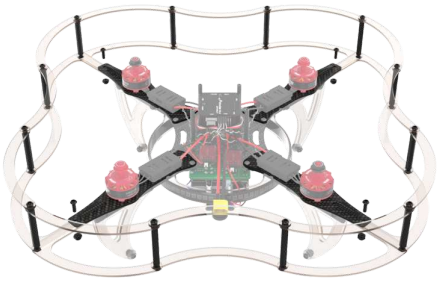
1. Соберите нижнюю часть защиты, используя 12 винтов M3x10 и 12 нейлоновых стоек 40 мм.



2. Установите верхнюю часть, используя 12 винтов M3x10.



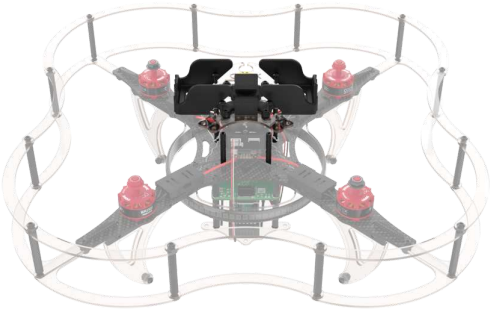
3. Установите защиту на коптер, с помощью 4 винтов M3x10 и стальных гаек.





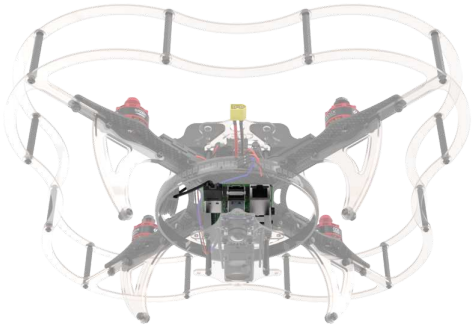
Установка верхней деки на коптер

1. Установите на верхнюю деку держатель АКБ с помощью 4 винтов M3x8 и стальных гаек.
2. Проденьте в пазы ремешок для фиксации АКБ.
3. Установите верхнюю деку на коптер с помощью 4 винтов M3x10.



4. Подключите USB кабель к разъему на полетном контроллере и USB разъему Raspberry Pi.





5. Зафиксируйте "улитку" кабеля в удобном месте с помощью двухстороннего скотча так, чтобы провод не мешал вращению винтов.



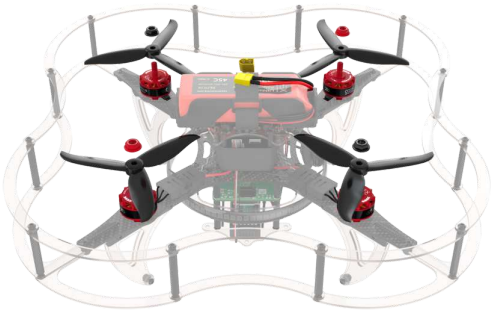
Установка пропеллеров и подготовка к полёту

Произведите настройку компонентов квадрокоптера, используя раздел "Настройка".

> Установка пропеллеров должна производиться **только после окончательной настройки коптера**, непосредственно перед полетом.

Установите 4 пропеллера, согласно схеме вращения. При установке пропеллеров АКБ должна быть отключена.

При установке будьте внимательны, чтобы пропеллер не был перевернут. На лицевой стороне пропеллера имеется маркировка его характеристик, а также направление вращения, которое должно совпадать с направлением вращения моторов.





Установка АКБ

> Убедитесь, чтобы все провода были спрятаны и движению пропеллеров ничего не мешает.

Проверьте сборку квадрокоптера:

- Балансировочный разъем АКБ должен быть спрятан под утягивающим ремешком.
- Регуляторы должны быть зафиксированы хомутами.
- Все провода, идущие от PDB и полетного контроллера, должны быть зафиксированы липучкой или обмотанной вокруг алюминиевых стоек.
- Пропеллеры установлены правильной стороной и соответствуют направлению кручения моторов.



Обязательно установите и настройте индикатор напряжения перед полетом, чтобы не переразрядить аккумулятор. Для настройки индикатора используйте кнопку расположенную в его основании. Отображаемые цифры во время настройки обозначают минимально возможное напряжение в каждой ячейке аккумулятора, рекомендуемое значение **3.5**.

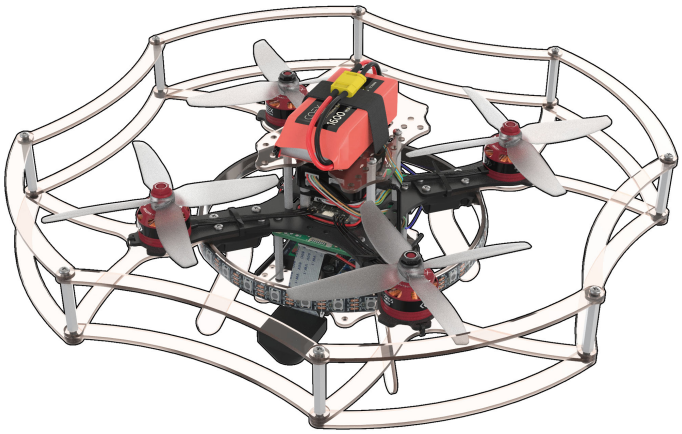
Звуковая индикация означает, что ваш аккумулятор разряжен и его нужно зарядить.



Дрон готов к полету!

Сборка Клевера 3

В данной инструкции рассматривается сборка комплекта COEX Clover 3 с платой регуляторов 4в1.



> Перед использованием паяльного оборудования обязательно ознакомьтесь с техникой безопасности при пайке.

Дополнительное оборудование



Условные обозначения



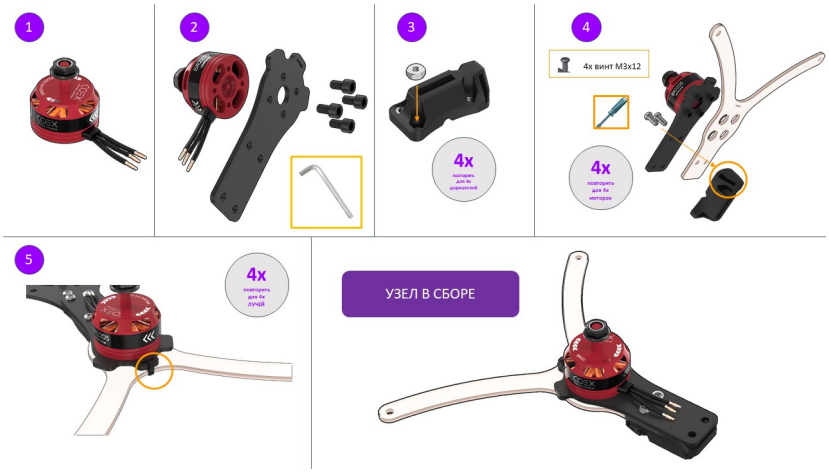
Установка моторов

1. Распаковать моторы.
2. Закрепить мотор на луче шестигранными винтами M3x6 (самые короткие винты в комплекте с моторами).

   Шестигранный ключ в комплекте.
3. Вставить гайки M3 (4 шт) в пластиковый держатель.

   Для удобства можно использовать длинный винт, либо плоскогубцы
4. Закрепить луч, нижнюю защиту луча и держатель винтами M3x12, используя крестовую отвертку.
5. Скрепить хомутом луч и нижнюю защиту луча.

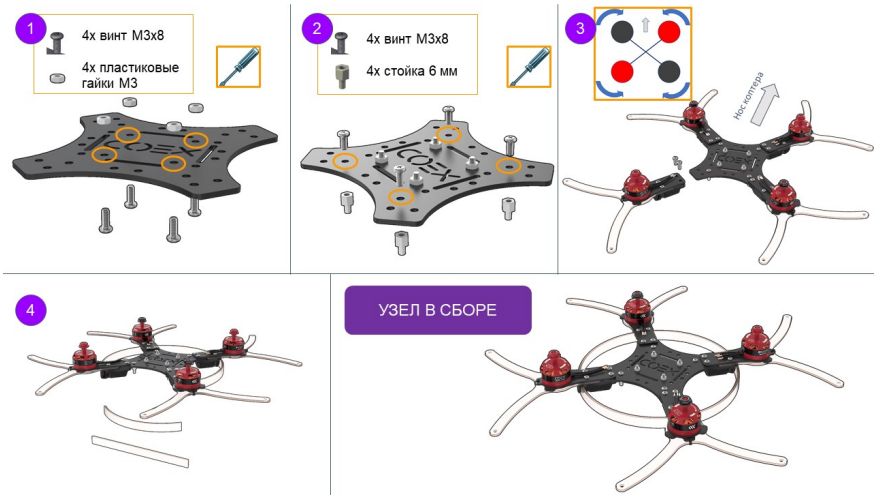
   Хвост от хомута (стяжки) отрезать ножницами.



Монтаж каркасных элементов

1. Установить пластиковые гайки M3 (4 шт) для крепления PDB на раму винтами M3x8.

2. Установить стойки 6 мм (4 шт) для крепления Raspberry Pi на раму винтами M3x8.
3. Установить на раму собранную конструкцию, соблюдая схему, винтами M3x16.
4. Установить каркас для светодиодной ленты, используя прорези в держателях для ножек.



Монтаж преобразователя напряжения BEC (припаять и проверить)

1. Распаковать плату питания и установить шлейф питания.
2. Включить мультиметр в режим измерения постоянного напряжения (диапазон 20В или 200В).
3. Проверить работоспособность платы питания, подключив АКБ
   - Выходное напряжение на разъеме XT30 должно равняться напряжению на АКБ (от 10В до 12.6В).
   - Выходное напряжение на шлейфе питания должно быть в пределах 4.9В до 5.3В.
   - Измеряем между черным и красным проводами.
4. Распаковывать преобразователь напряжения и снимаем прозрачную изоляцию.
5. Припаять два дополнительных провода на BEC

   - Взять из набора 3 провода папа-мама (красный, черный и любого цвета)
   - Красный и черный залудить с обеих сторон, используя пинцет. На синем проводе залудить со стороны коннектора ПАПА.

     Залудить - это:

     - Нанести флюс на оголенную часть провода.
     - Покрыть припоем.
   - Припаять красный и черный провода к BEC:

```
ЧЕРНЫЙ -> OUT-
КРАСНЫЙ -> OUT+
```

6. Проверить работу BEC.
    - Припаять BEC на плату питания:

      ```
      ЧЕРНЫЙ -> -
      КРАСНЫЙ -> +
      ```

    - Подключить АКБ и проверяем напряжение на припаянных проводах к BEC (из пункта 5).

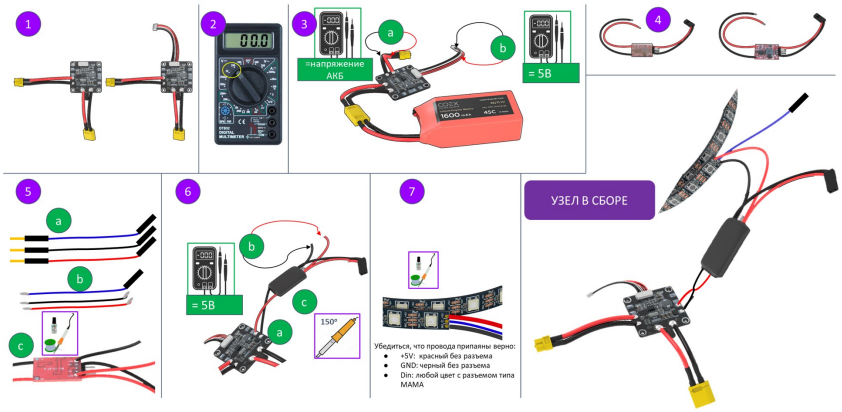
      5В - все правильно!

      больше 10В - отключите питание и переставьте желтую перемычку на другой пинцет.

      0В - плохо спаяли.

    - Если BEC выдает 5В, то изолируем паячное соединение черной термоусадкой.
7. Монтаж светодиодной ленты.
    - Припаять провода от BEC (из пункта 5) к светодиодной ленте.
    - Удалить силиконовый слой на ленте (надрезать ножом и оторвать).
    - Залудить контакты светодиодной ленты.

      ```
      Красный -> +5V
      Черный ->  GND
      Синий -> Din
      ```



Монтаж регуляторов

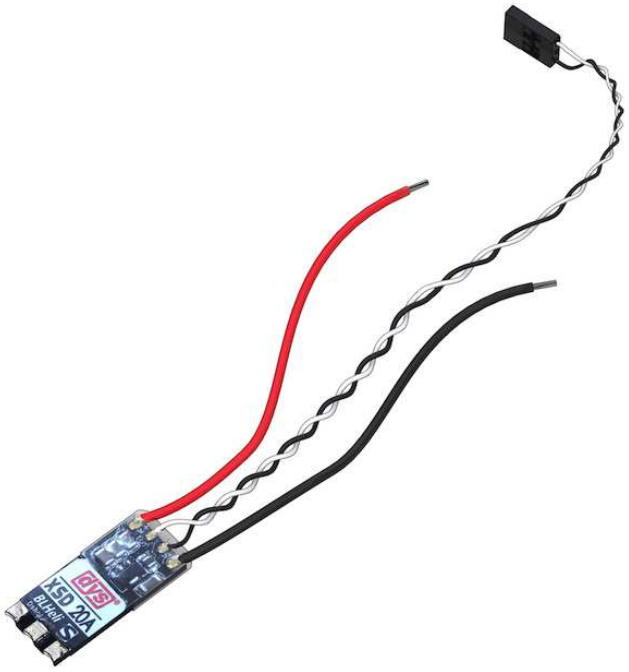
Монтаж 4 отдельных регуляторов

Залудить три контактные площадки регулятора

- Нанести флюс

- Нанести припой

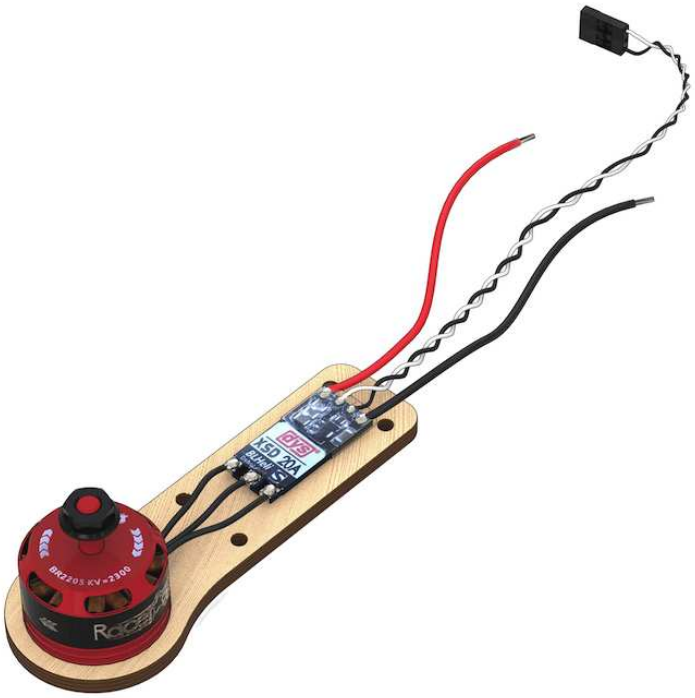
Чтобы припой аккуратно заполнил всю площадку, необходимо прогреть площадку регулятора. Для этого нужно удерживать жало паяльника на контактной площадке в течение 2 сек (или больше, если потребуется)



- Повторить данную операцию для оставшихся трех регуляторов

Припаять провода моторов к регуляторам

Припаять ранее приготовленные провода моторов к контактным площадкам регуляторов.



- Повторить данную операцию для оставшихся трех регуляторов

Монтаж платы регуляторов 4в1 и платы питания PDB

1. Установить плату регуляторов 4в1, как показано на картинке.

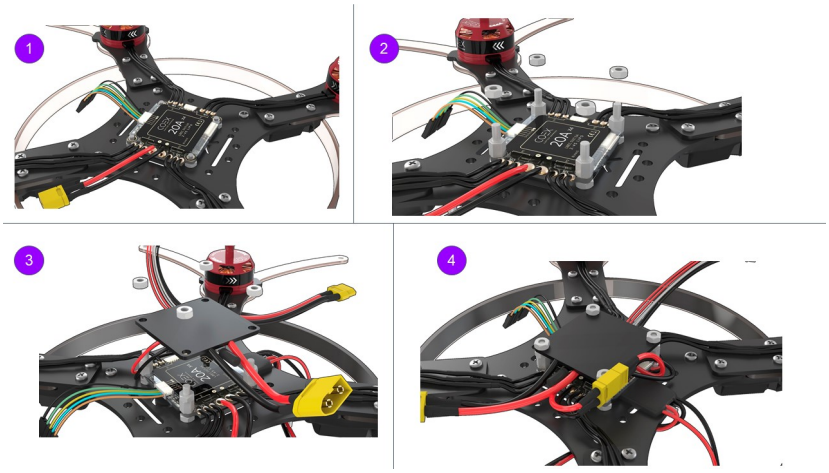
   Соединить фазные провода моторов с проводами регуляторов.

2. Закрепить плату регуляторов стойками 6 мм (4 шт.).

   На стойки накрутить пластиковые гайки M3 (4 шт.).

3. Установить плату распределения питания PDB, как показано на картинке (разъем XT60 направлен к хвосту коптера).

4. Соединить разъемы питания платы питания и платы регуляторов XT30.



Сопряжение приемника и пульта

1. Подключить провод 5В от BEC в разъем приемника.

   Установить BIND разъем в крайний правый порт B/VCC.

2. Подключить АКБ. Индикатор на приемники должен быстро мигать (режим сброса).

3. Зажать и удерживать кнопку BIND на пульте и включаем пульт.

   На пульте отображается процесс сопряжения RXBinding

4. После установки сопряжения (появление допю строк на дисплее пульта):

   - Убрать BIND разъем из приемника.
   - Отключить АКБ.



Если пульт не включается или заблокирован, см. статью неисправности пульта.

Проверка направления вращения моторов

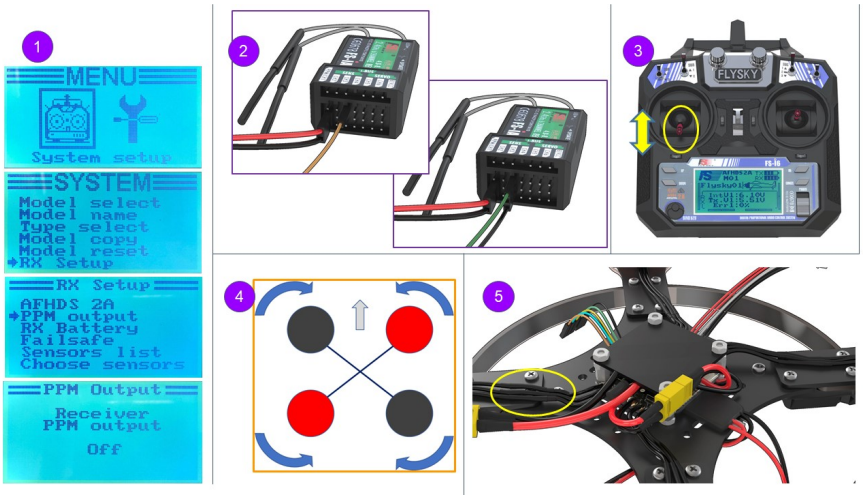
1. Включить пульт.

   Убедиться, что PPM в меню RX Setup отключен (раздел "Нет связи с полетным контроллером")

   В пункте 3 выберите “RX setup” > “PPM OUTPUT” > “Off”.

Сохраните изменения (удерживаем нажатой кнопку “CANCEL”).

2. Подключить оранжевый провод S1 от платы регуляторов в CH3 на приемнике. Подключить внешнее питание.
3. Подать левым стиком газ (throttle) на 10%.
4. Проверить направления вращения мотора по схеме.Повторить для каждого мотора. Таким образом, будет понятно каким именно мотором мы управляем.
5. Если необходимо изменить направление вращения, то меняем любые два фазных провода мотора (нужно переподключить).

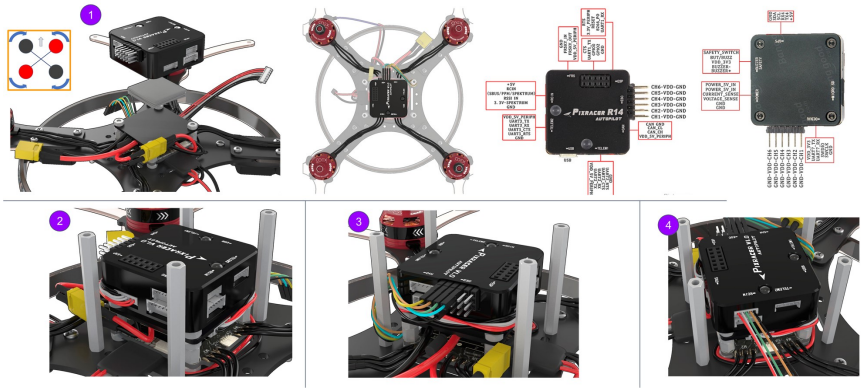


Монтаж и подключение полетного контроллера Pixracer

1. Установить Полетный контроллер Pixracer на двухсторонний скотч 3M (2-3 слоя). Также полетный контроллер можно извлечь из корпуса и жестко установить на стойке M3x6.
2. Установить стойки 40 мм, используя винты M3x8.

   Подключить разъем POWER.
3. Подключить регуляторы, как на картинке.

   Подробно про подключение регуляторов 4в1.
4. Подключить шлейф радиоприемника в разъем RCIN в Pixracer.



Монтаж Raspberry

1. Перевернуть коптер.

   Установить Raspberry на стойки, используя монтажные отверстия Raspberry.

   USB-разъемы направлены к хвостовой части коптера.

2. Установка шлейфа для камеры:

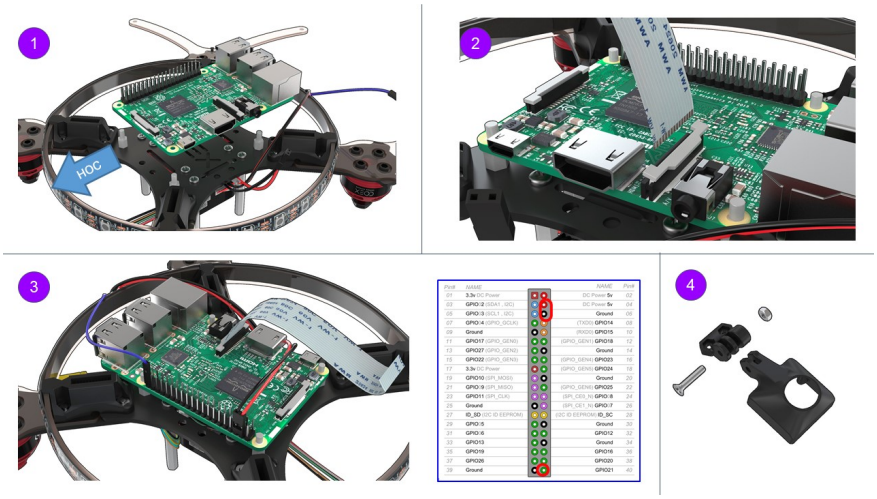
   - поднять защелку;
   - подключить шлейф;
   - закрыть защелку.

3. Подключение питания Raspberry:

```
5B -> pin 04 (DC power 5v)
GND -> pin 06 (Ground)
Подключение светодиодной ленты pin 40 (GPIO21)
```

4. Сборка маунта для камеры RPi.

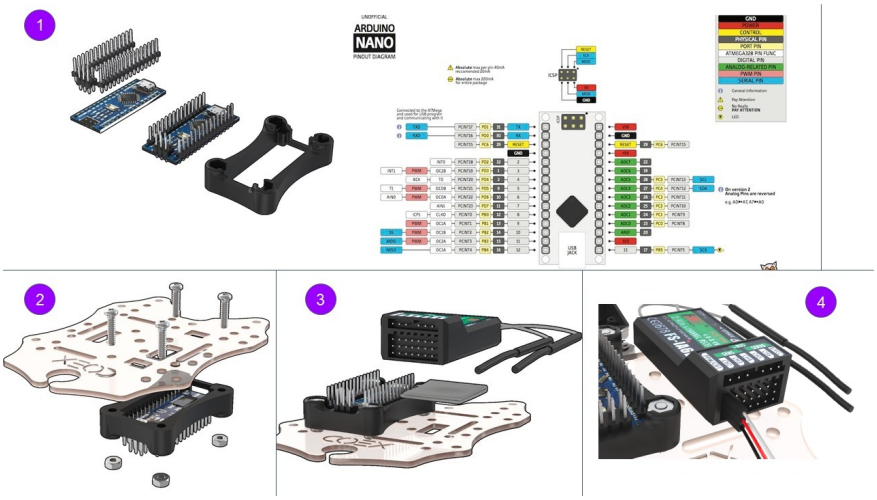
   Используйте винт M3x16 и гайку M3



Монтаж Arduino и радиоприемника FlySky

1. Произвести монтаж пинов микроконтроллера Arduino Nano, используя пайку.
2. Установить микроконтроллер в специальной маунт и прикрепите к нижней деке, используя винты M3x16 (4 шт.).
3. Используя двухсторонний скотч, прикрепить приемник, как показано на рисунке.
4. Подключить шлейф радиоприемника от Pixracer как на рисунке.

```
белый -> PPM
красный -> 5V
черный -> GND
оранжевый, зеленый -> не используются. Выньте эти провода из разъёма или
```

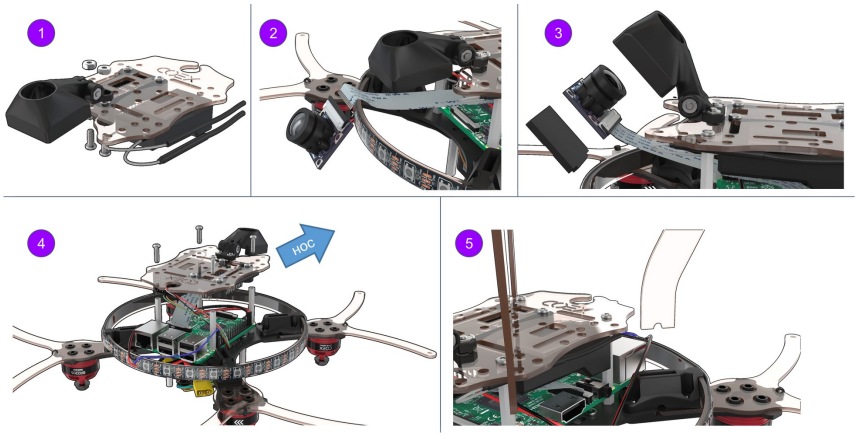


Монтаж камеры RPi

1. Установить маунт для камеры RPi в сборе на нижнюю деку винтами M3x12 (2 шт.).
2. Подключить шлейф к камере RPi.
3. Установить камеру в маунт, закрепить саморезами M2.
4. Закрепить Raspberry стойками 30 мм (4 шт.).

   Установить нижнюю деку в сборе на стойки винтами M3x8 (4шт.)

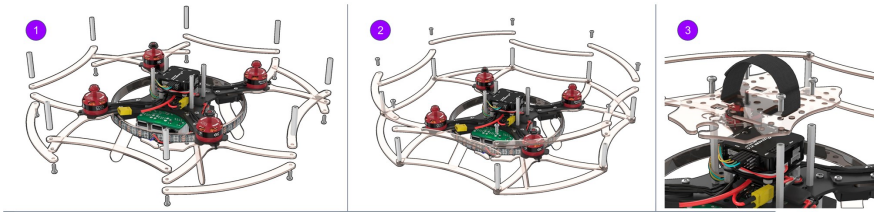
5. Установить ножки в маунты (4 шт.).



Монтаж остальных конструктивных элементов

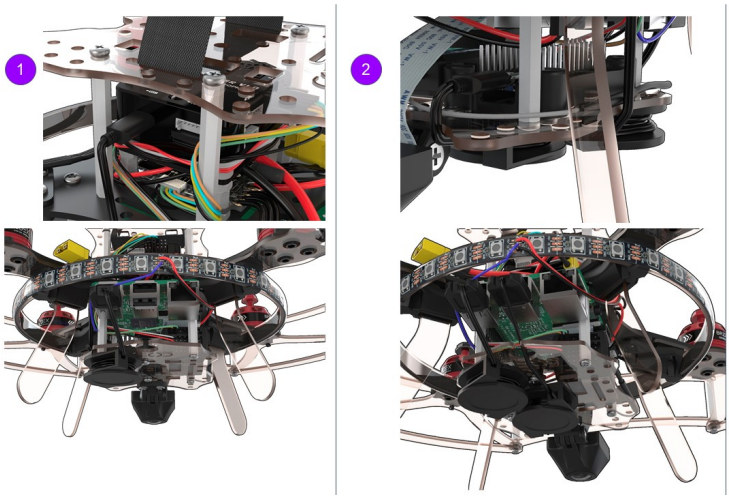
1. Установить нижней защиты, используя винты M3x12 (8 шт.) и стойки 30 мм (8 шт.).
2. Установить верхней защиты, используя винты M3x12 (8 шт.).
3. Установить ремешок в верхнюю деку для фиксации АКБ.

   Закрепить верхнюю деку винтами M3x8 (4 шт.)



Монтаж USB соединителей

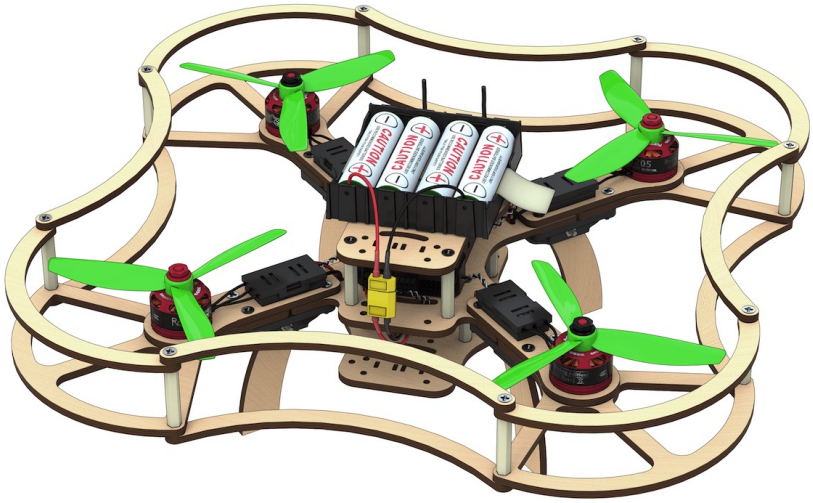
1. Соедините Pixracer и Raspberry, используя micro USB - USB кабель.
2. Соедините Arduino и Raspberry, используя micro USB - USB кабель.



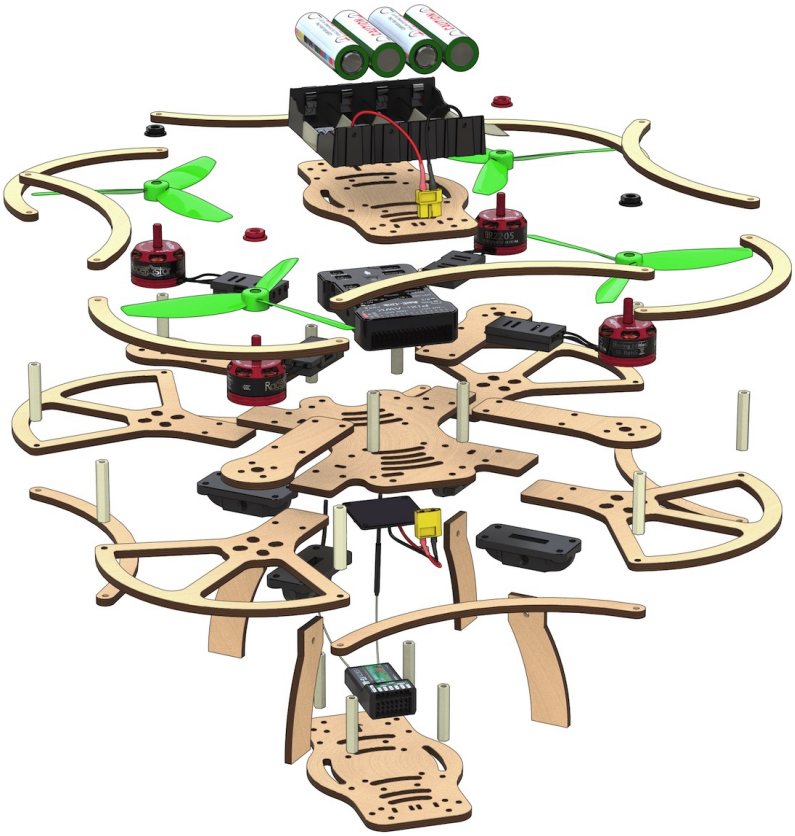
.

Подробнее про подключение см. статью.

Инструкция по сборке конструктора Клевер 2



Состав конструктора



- Рама центральная x2.
- Рама дополнительная x4.

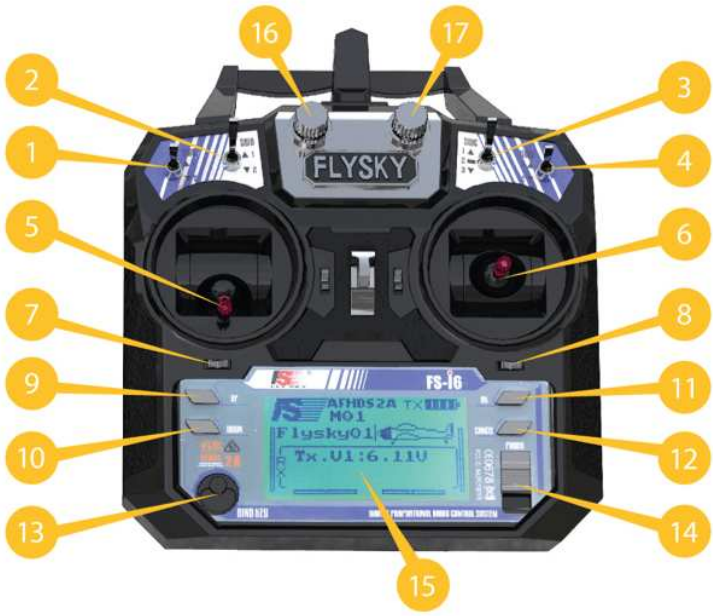
- Луч x8.
- Ножки x8.
- Защита для лучей x8.
- Защита пропеллеров x16.
- Защита боковая x16.
- Пропеллер пластиковый Dalprop 5045 x4.
- Бесколлекторный электродвигатель Racerstar BR2205 2300kV x4.
- Регуляторы хода ESC, DYS XSD20A x4.
- Разъем силовой XT60 pin x1.
- Разъем силовой XT60 socket x1.
- Трехпроводной шлейф “мама-мама” x2.
- Провод медный многожильный с силиконовой изоляцией 14AWG (красный, черный), длина 50 см.
- Плата распределения питания PDB BeeRotor Power Distribution Board V2.0 x1.
- Аккумуляторная батарея (АКБ) Li-ion 18650 x8.
- Зарядное устройство EFEST Luc V4 Li-Ion x1.
- Защитный бокс регуляторов x4.
- Крепление под ножки x8.
- Полетный контроллер Pixhawk x1.
- Радиоприемник FlySky i6 x1.
- Радиопульт FlySky i6 x1.
- Зарядное устройство EFEST LUC V4 x1.
- Провод Micro USB - USB x1
- Батарейный отсек 18650 Li-ion x1
- Провод медный многожильный с силиконовой изоляцией 18AWG (красный, черный), длина 100 см.
- Батарейка AA x4
- Джампер, Bind-разъем

Крепежные элементы

- Пластиковые стойки 6 мм x28.
- Пластиковые стойки 30 мм x32.
- Винты M3x8 x48.
- Винты M3x12 x24.
- Винты M3x16 x40.
- Гайки Пластиковые x8.
- Гайки Металлические x48.
- Наклейки для отсека АКБ x8 .
- Термоусадка ф15 , .50 см
- Термоусадка ф5, 100 см
- Двухсторонний скотч 3M x16.
- Отвертка x1 (нужна визуализация)
- Изоляционная лента x1
- Ножницы канцелярские x1
- Ремешок для батареи 250 мм x1

Функционал радиопульта Flysky i6

1. Переключатель A (SwA).
2. Переключатель B (SwB).
3. Переключатель C (SwC).
4. Переключатель D (SwD).
5. Левый стик.
6. Правый стик.
7. Левый триммер.
8. Правый триммер.
9. Кнопка Вверх.
10. Кнопка Вниз.
11. Кнопка Ок.
12. Кнопка Отмены.
13. Кнопка BIND KEY.
14. Переключатель питания POWER.
15. ЖК-дисплей.
16. Ручка A (VrA).
17. Ручка B (VrB).



Дополнительное оборудование

Данное оборудование не входит в состав конструктора Клевер 2, но оно необходимо для реализации сборочного процесса

1. Паяльник
2. Канифоль/ Флюс (нейтральный)
3. Припой
4. Фен промышленный

5. Плоскогубцы
6. Пинцет
7. Канцелярский нож
8. Мультиметр



Порядок сборки

Установка моторов

- Распаковать моторы. Используя плоскогубцы, укоротить провода на моторах, обрезать половину длины (оставив 25 мм).



Зачистить

- снять 2мм термоизоляции с конца провода не повредив медные жилы.

Скрутить провода.

Залудить

- Нанести флюс на оголенную часть провода.
- Покрыть припоем, используя пинцет.

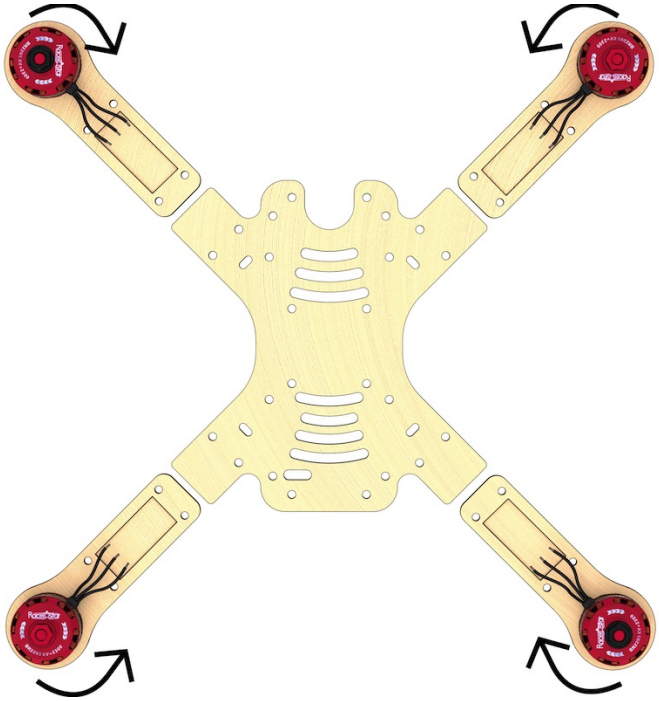


Закрепить мотор на луче

- Установить мотор на сторону луча с гравировкой.
- Прикрепить моторы к лучам винтами M3x8, используя отвертку.



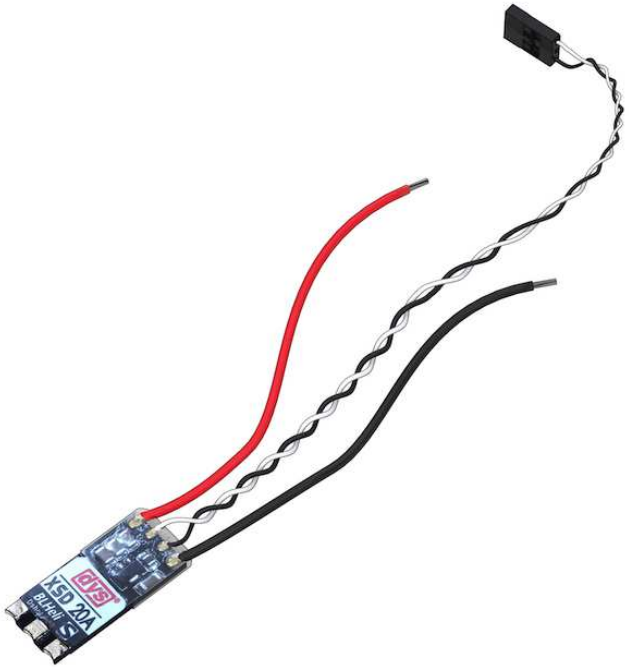
- Лучи с моторами необходимо расположить согласно схеме. Стрелками указано направление вращения моторов.



Залудить три контактные площадки регулятора

- Нанести флюс
- Нанести припой

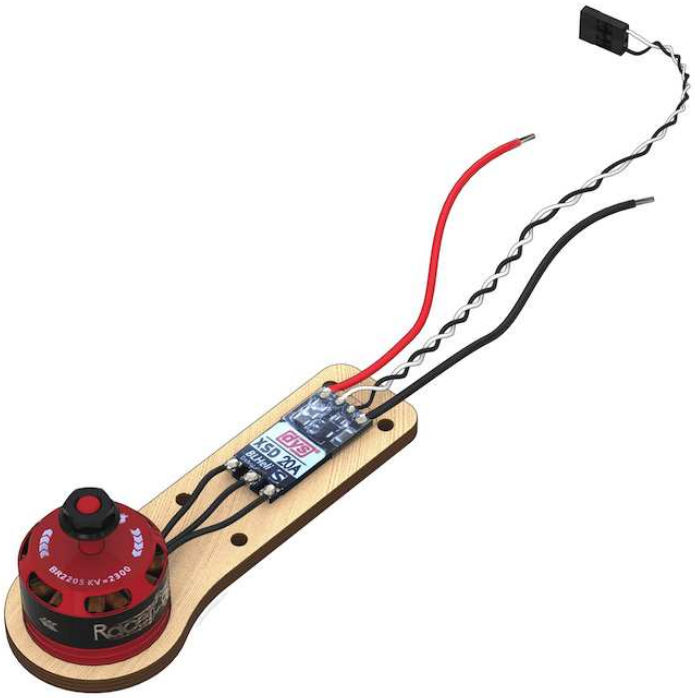
Чтобы припой аккуратно заполнил всю площадку, необходимо прогреть площадку регулятора. Для этого нужно удерживать жало паяльника на контактной площадке в течение 2 сек (или больше, если потребуется)



- Повторить данную операцию для оставшихся трех регуляторов

Припаять провода моторов к регуляторам

Припаять ранее приготовленные провода моторов к контактным площадкам регуляторов.

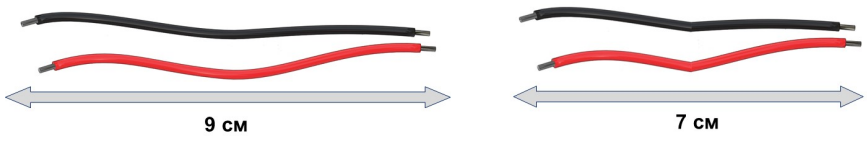


- Повторить данную операцию для оставшихся трех регуляторов

Монтаж разъемов питания

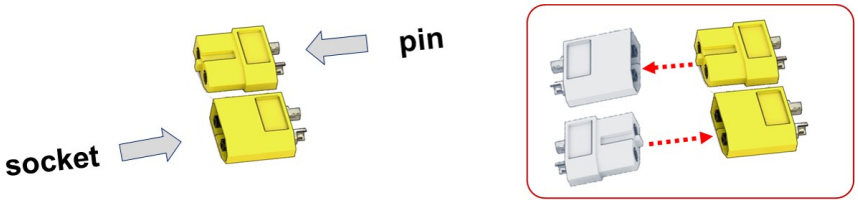
Подготовка проводов для силовых разъемов XT60

1. Взять моток красных и черных проводов, промаркированных как 14AWG
2. Отрезать 4 куска провода следующей длины
3. Длина 7 см (Для силового разъема XT60 pin) - 1 красный, 1 черный
4. Длина 9 см (Для силового разъема XT60 socket) - 1 красный, 1 черный

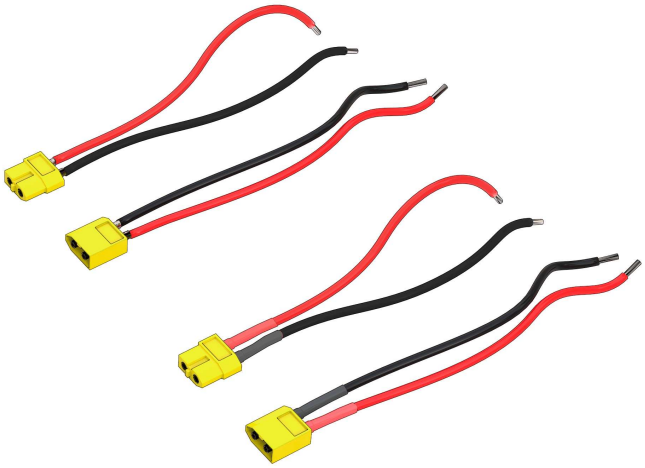


Подготовка силовых разъемов питания XT60 pin и XT60 socket

Статья про силовые разъемы и их обозначения



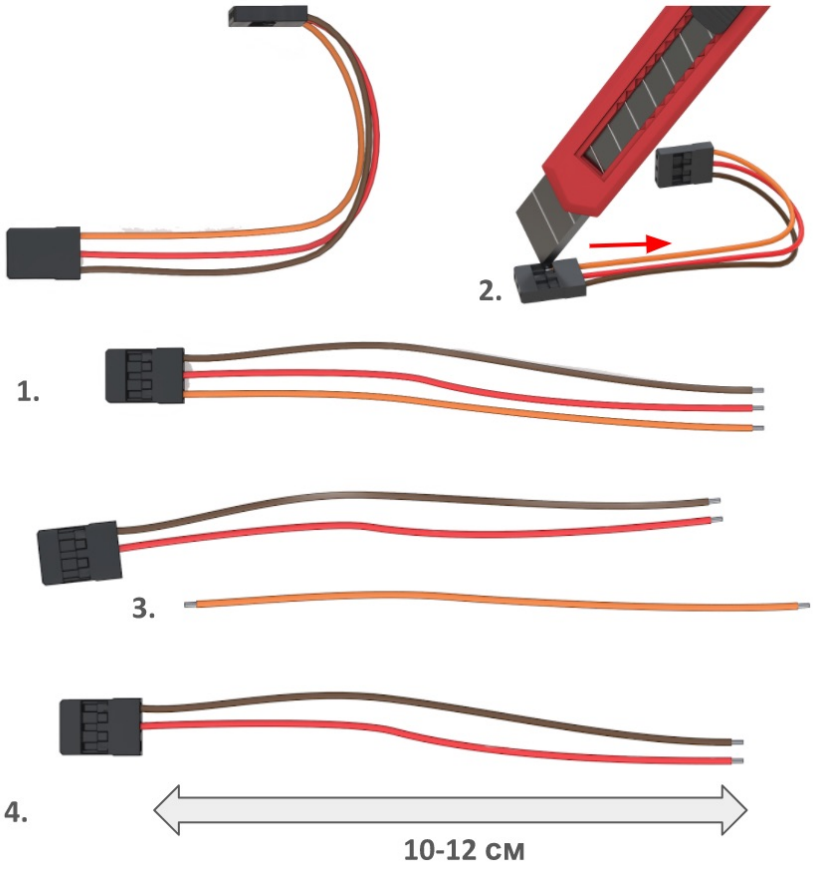
1. Под разъем XT60 pin залудить два силовых провода красный и чёрный 14AWG длиной 7 см.
2. Залудить контактные площадки разъема XT60 pin.
3. Припаять черный провод к “-” контакту разъема.
4. Припаять красный провод к “+” контакту разъема .
5. Нарезать термоусадку ф5 (2 отрезка по 10 мм).
6. Надеть термоусадку ф5 на провода так, чтобы она закрывала контактные площадки проводов с XT60 .
7. Усадить термоусадку феном.



8. Повторить процедуру для разъема XT60 socket.

Подготовка разъема питания управляющей цепи 5В

1. Обрезать/вытащить все пины из одного из разъемов. Отсоединить его.
2. Поддеть канцелярским ножом фиксатор на оставшемся разъеме, чтобы освободить 3-й провод.
3. Убрать 3-й (оранжевый) провод из разъема, за ненадобностью.
4. Длина оставшихся черного и красного проводов 10-12 см.

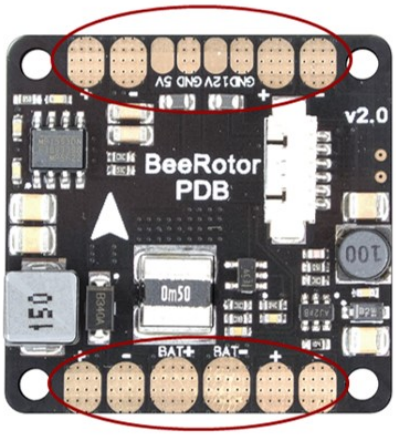


Монтаж платы распределения питания

Предпаячная проверка

Статья про прозвонку





Прозвонить следующие цепи на НЕЗАМКНУТОСТЬ (отсутствие звукового сигнала мультиметра):

- “BAT+” и “BAT-”
- “12V” и “GND”
- “5V” и “GND”

Прозвонить следующие цепи на ЗАМКНУТОСТЬ (появление звукового сигнала мультиметра):

- “BAT-” с каждым контактом, обозначенным “-” и “GND”
- “BAT+”, с каждым контактом, обозначенным “+”

Залудить контактные площадки платы питания

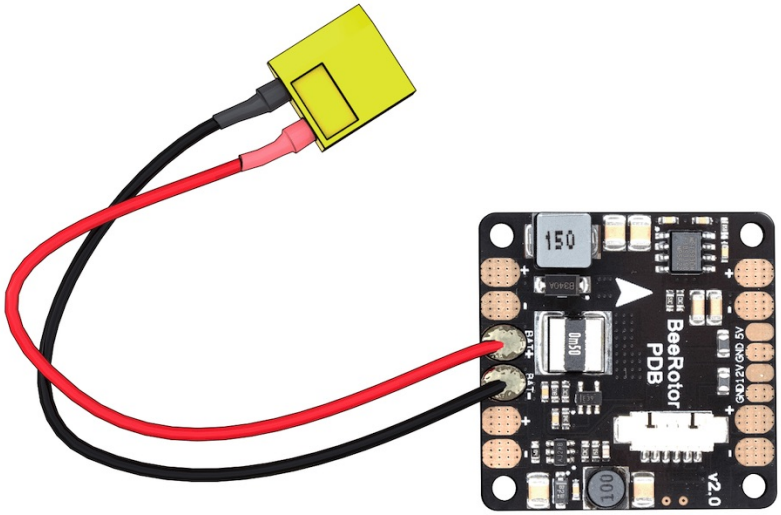
1. Залудить* контактные площадки платы питания.
2. С помощью мультиметра проверить отсутствие контактного замыкания на плате (прозвонить)



Чтобы припой аккуратно заполнил всю площадку, необходимо её прогреть. Для этого нужно удерживать жало паяльника на контактной площадке в течение 2 сек (или больше, если потребуется)

Пайка силового разъема питания XT60

Припаять разъем для АКБ, соблюдая полярность на контактных площадках.

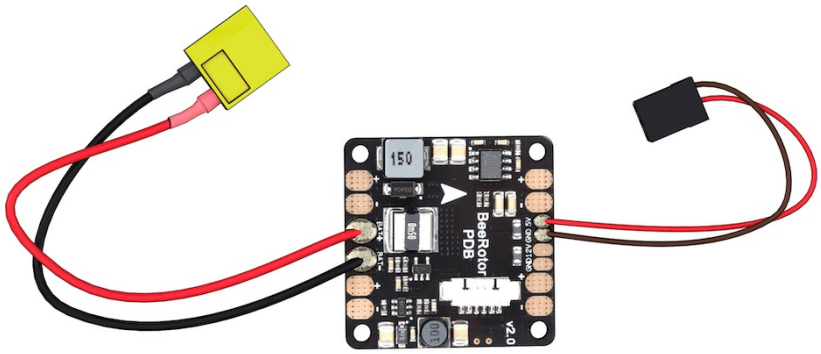


ВАЖНО о полярности

- красный провод - это “+”
- черный провод - это “-”

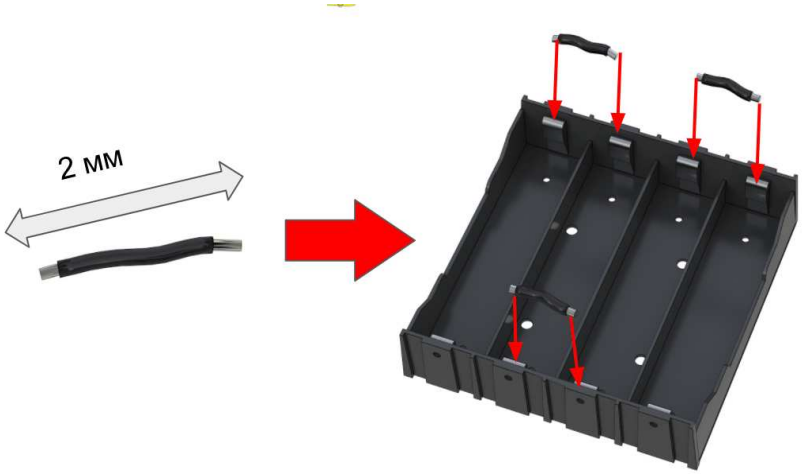
Пайка разъема питания управляющей цепи 5В

Припаять разъем 5В, соблюдая полярность на контактных площадках. (на изображении: красный провод - это питание “+”)



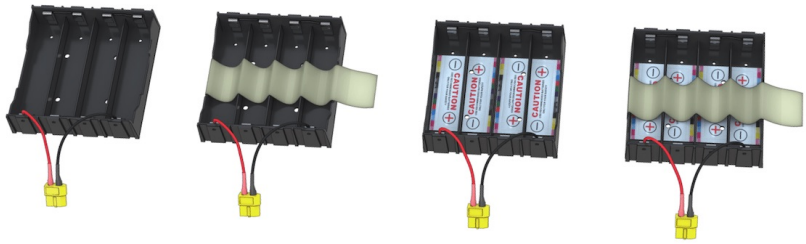
Монтаж отсека АКБ

Подготовка перемычек (3 шт.)



- Отрезать силовой провод длиной 2 см.
- Зачистить с обеих сторон.
- Залудить.
- Сделать 3 перемычки
- Припаять перемычки по схеме.
- Прозвонить мультиметром. В случае необходимости зачистить наждачной бумагой.

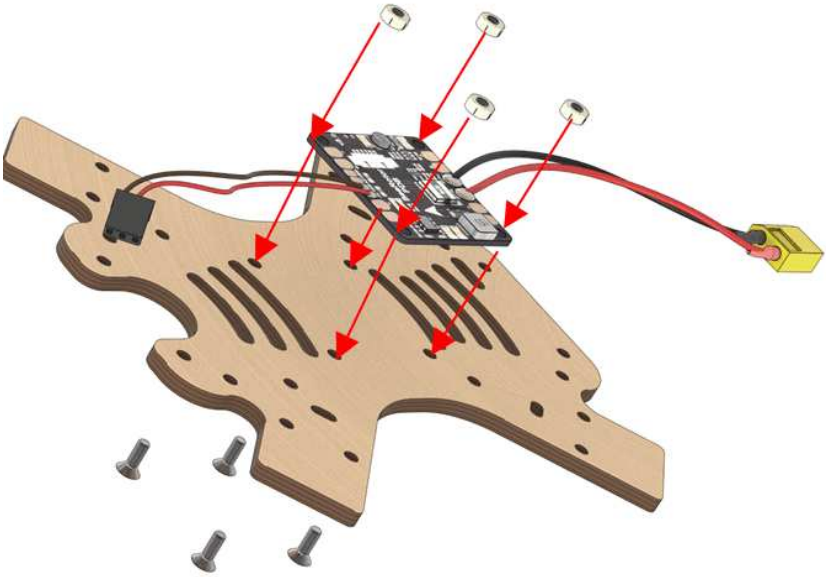
Подготовка отсека АКБ



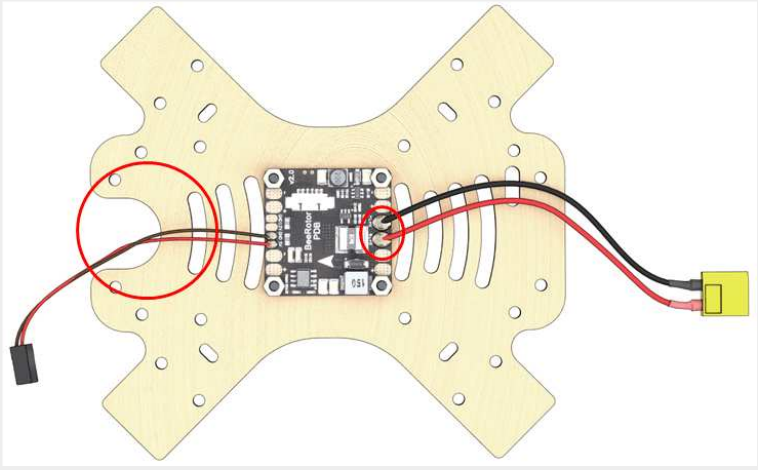
- Приклеить наклейки с разметкой внутрь отсека АКБ, в соответствии с полярностью.
- Приклеить ленту из скотча на дно отсека.

Монтаж платы распределения питания

- Установить плату питания на раму винтами М3х8 и пластиковыми гайками.



Стрелочка на плате направлена в сторону носового выреза

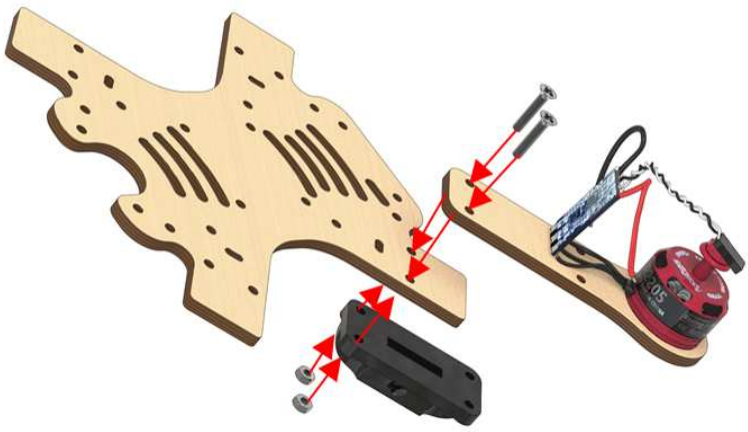


Монтаж элементов

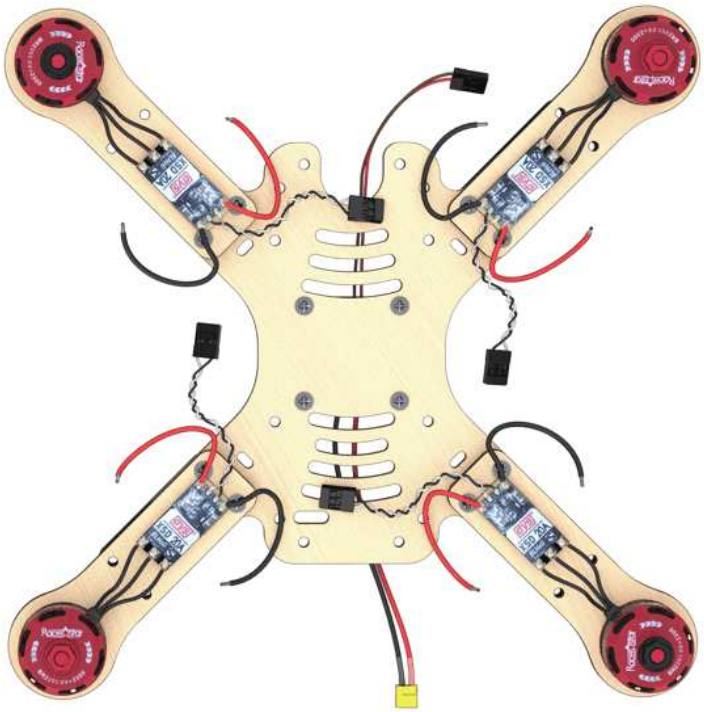
1. Установить гайки в пластиковые держатели.



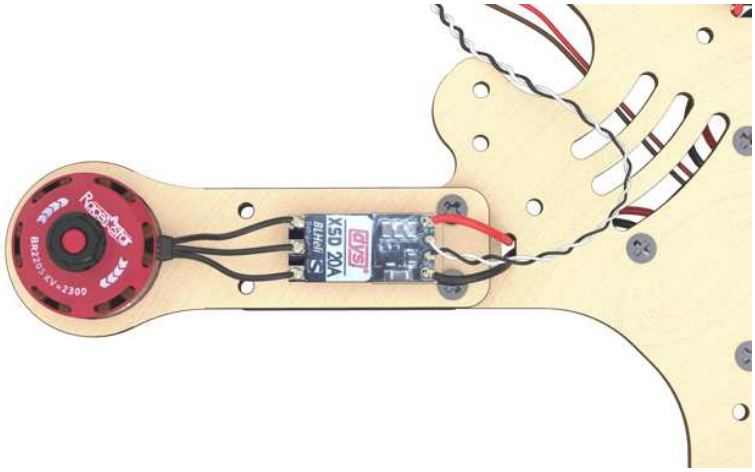
2. Установить лучи на раму винтами M3x16
    - Лучи устанавливаются поверх рамы
    - Пластиковые держатели устанавливаются снизу рамы.



3. Расположение моторов. Проверить расположение моторов (моторы с черной гайкой в левом верхнем углу и в правом нижнем).

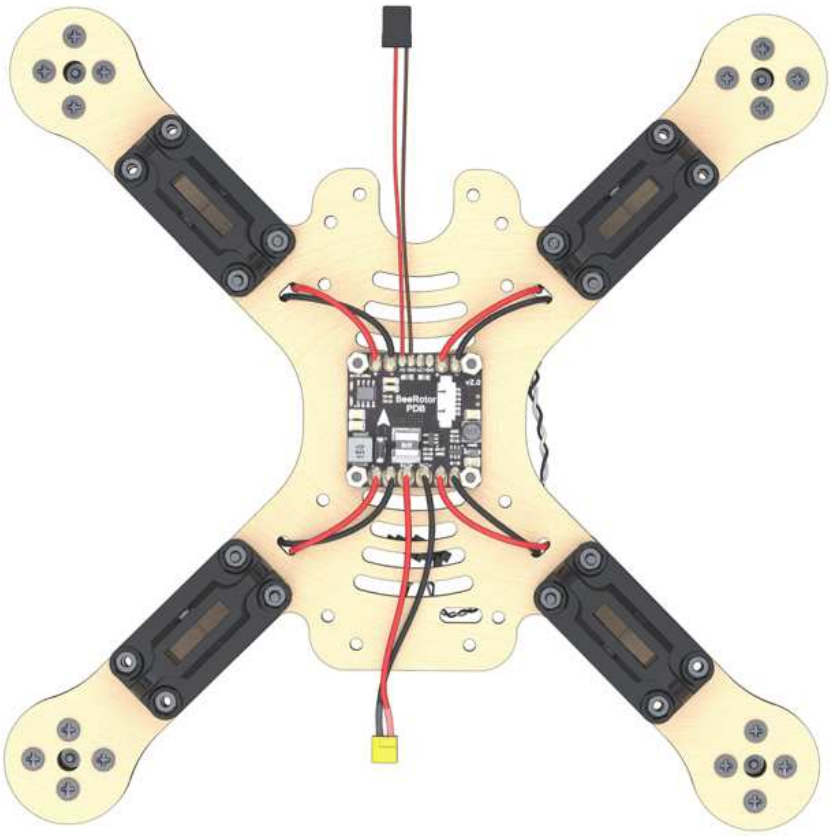


4. Продеть силовые провода регуляторов в отверстия.



Пайка силовой цепи платы питания

Припаять силовые провода регуляторов к плате питания, соблюдая полярность.

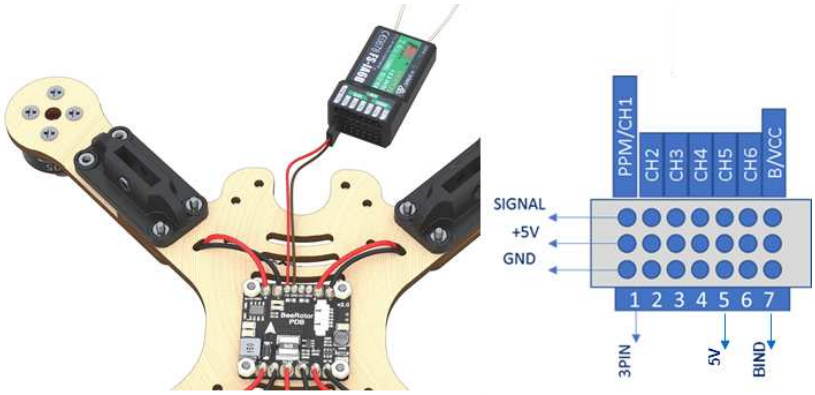


ВАЖНО о полярности

- красный провод - это “+”
- черный провод - это “-”

Сопряжение приемника и пульта

1. Подключить радиоприемник к разъему 5В. В любой разъем, GND внизу. На схеме питание обозначено как 5V



2. Подключить АКБ. Светодиод на радиоприемнике должен мигать. ! [Подключение АКБ]

БЕЗОПАСНОСТЬ при работе с АКБ

Безопасность при работе с Li-ion аккумуляторами 18650

- Обращаться с аккумуляторами бережно. Не допускать падений, ударов деформаций.
- При подключении (отключении) аккумуляторов держаться только за разъёмы, тянуть или дергать за провода запрещается.
- В случае обрыва разъемов, нарушения целостности изоляции или корпуса аккумулятора, не трогая его, немедленно сообщить преподавателю.



Включение радиопульта

1. Вставить джампер в B/VCC радиоприемника (замкнуть "землю" и "сигнал")
2. На пульте зажать кнопку BIND KEY.
3. Включить пульт (перещелкнуть POWER, BIND KEY не отпускаем).
4. Подключить аккумулятор к коптеру.
5. Ждем синхронизацию.
6. Отсоединить джампер.
7. Светодиод горит непрерывно.

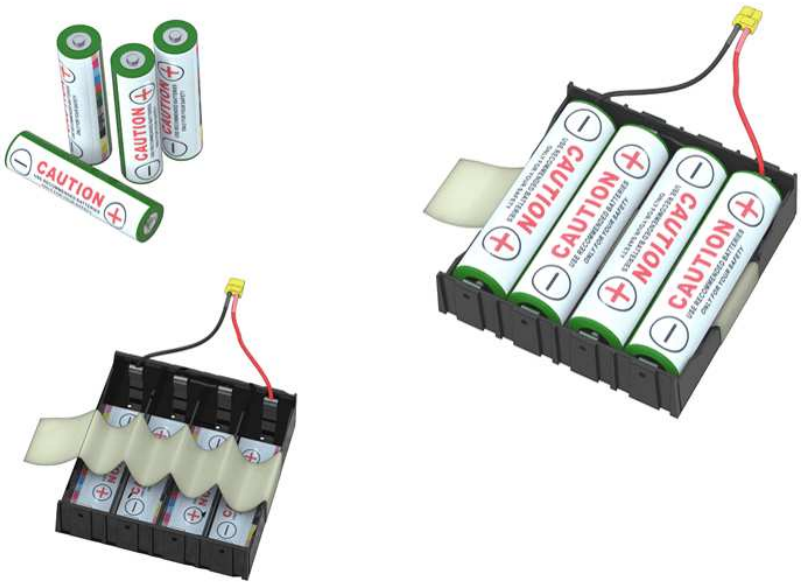


Мануал по неисправностям радиоаппаратуры

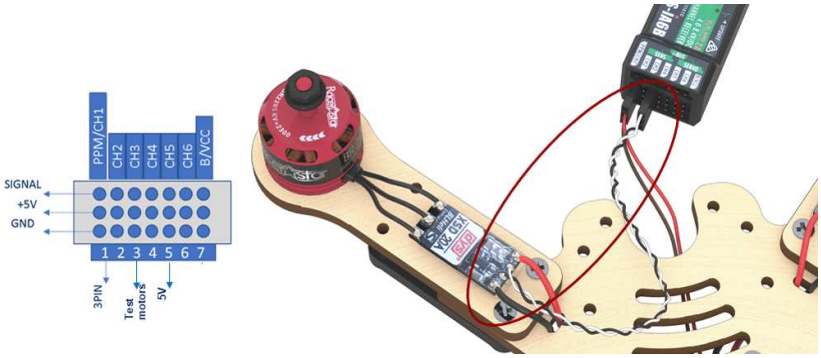
Проверка направления вращения моторов

1. Наклеить наклейки на АКБ 18650.

2. Установить 18650 в отсек АКБ, соблюдая полярность.



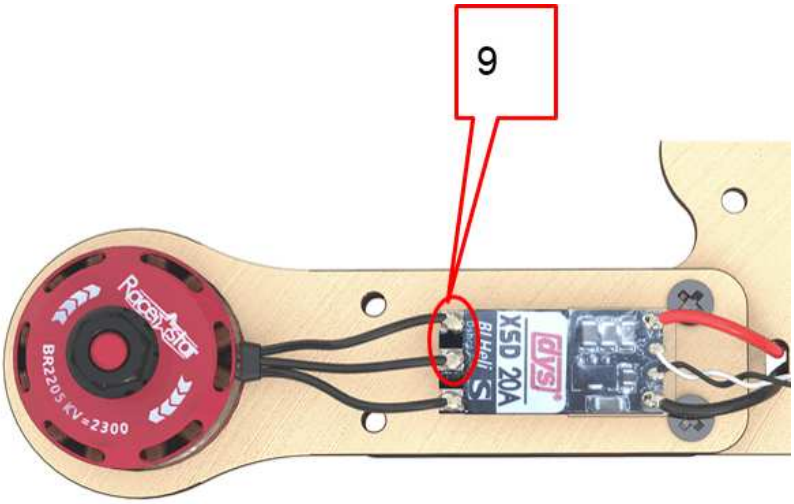
3. Проверить, что разъем питания 5В подключен к приемнику по схеме.
4. Подключить регулятор мотора к 3 каналу приемника CH3 по схеме.



5. Подключить внешнее питание (АКБ).
6. Включить пульт.
7. Подать левым стиком газ (throttle) на 10%.
8. Проверить направления вращения мотора по схеме.

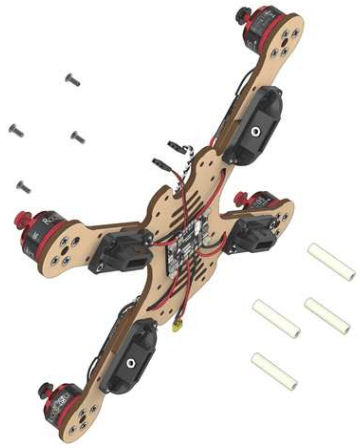


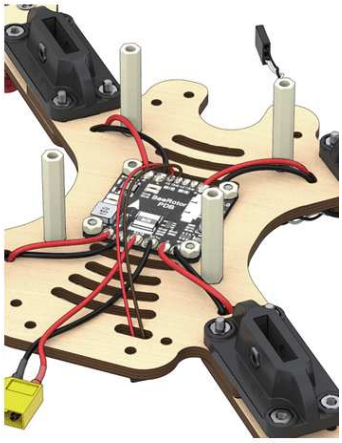
9. Если необходимо изменить направление вращения, то меняем любые два фазных провода мотора (нужно перепаять).



Монтаж радиоприемника

1. Установить пластиковые стойки 30 мм на раму винтами M3x8.
2. Разъем питания 5В продеть в прорезь.

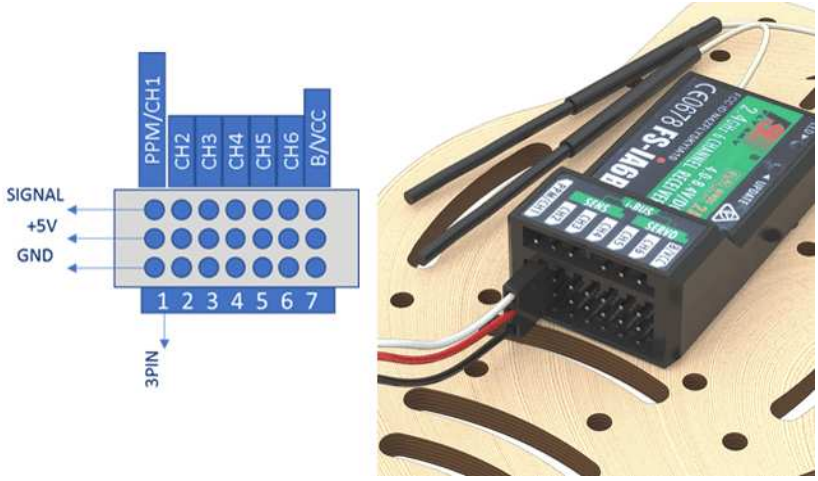




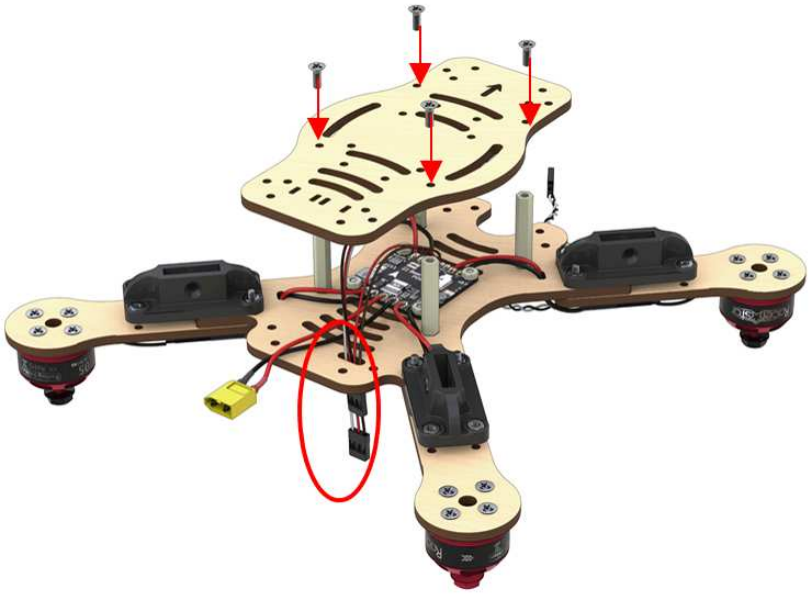
3. Приемник прикрепить к нижней дополнительной раме, используя двухсторонний скотч и ориентируясь на гравировку. Антенны направлены вперед.



4. Установить 3х проводной шлейф в канал PPM / CH1.



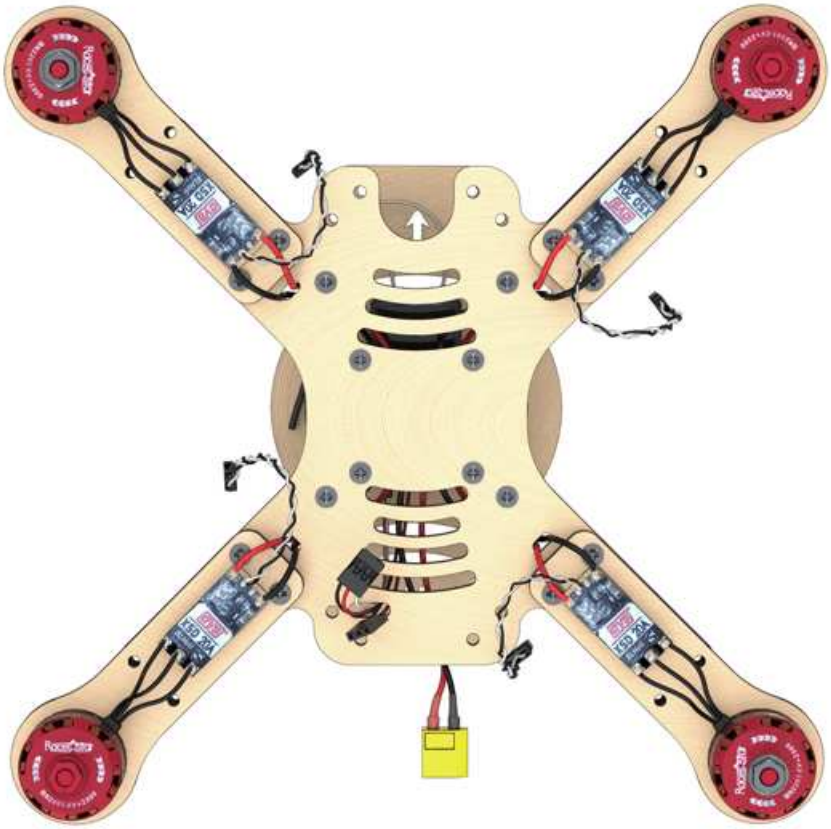
5. Продеть в прорезь к разъему 5 В.
6. Прикрутить нижнюю дополнительную раму к стойкам на центральной раме винтами М3х8.



Направление стрелок на плате питания и на дополнительной раме совпадают

Монтаж полетного контроллера

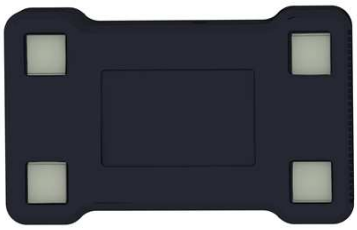
Переворачиваем сборку



Установка полетного контроллера Pixhawk

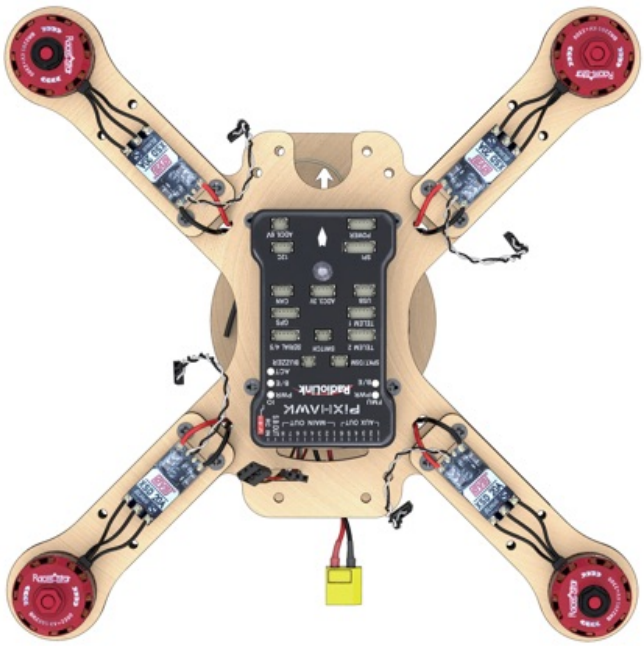
1. Клеим 2х сторонний скотч по углам полетного контроллера.





При работе моторов возникают вибрации, отрицательно влияющие на показания датчиков полетного контроллера Pixhawk. Чтобы избежать этого эффекта, количество слоев двустороннего скотча лучше увеличить до 4-5.

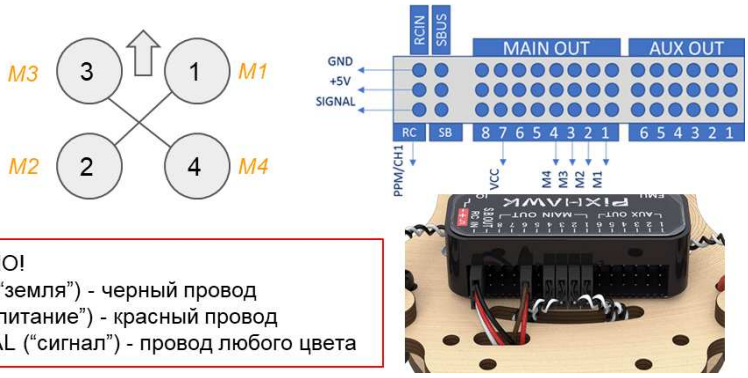
2. Установить полетный контроллер в центр рамы.



Стрелки на раме и Pixhawk должны быть сонаправлены

Подключение полетного контроллера по схеме

1. PPM (трехпроводной шлейф) подключить к порту RCIN
2. Моторы к 1,2,3,4 портам MAIN OUT, согласно схеме
3. Питание от PDB (5В/VCC) в любой порт, кроме SB (SBUS)



ВАЖНО!
GND (“земля”) - черный провод
+5V (“питание”) - красный провод
SIGNAL (“сигнал”) - провод любого цвета

Сборка регуляторов

1. Клеим 2х сторонний скотч на основание защитного бокса регуляторов.
   Скотч на бокс регулей
2. Укладываем регуляторы в защитные боксы. Крепим полученную сборку к лучам рамы.

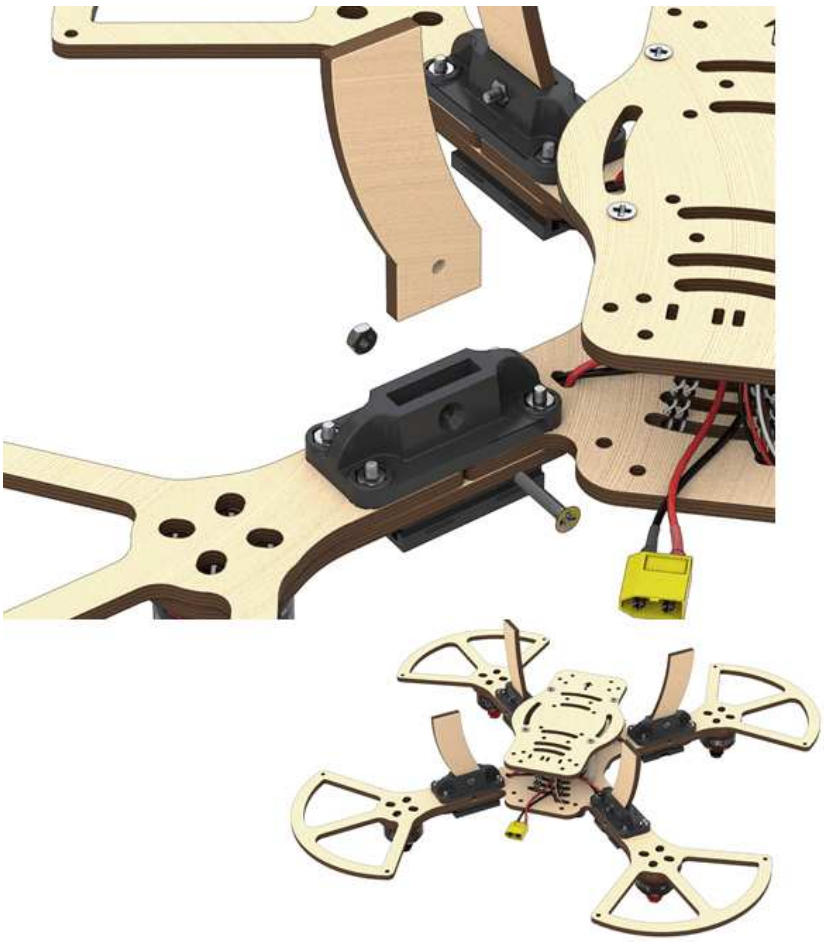


Установка защиты

1. Закрепить нижнюю защиту винтами M3x16 на лучах рамы.



2. Закрепить ножки к пластиковым держателям винтами M3x16.



3. Закрепить стойки 30 мм в отверстия нижней защиты винтами M3x12.



4. Закрепить верхнюю защиту винтами M3x12.

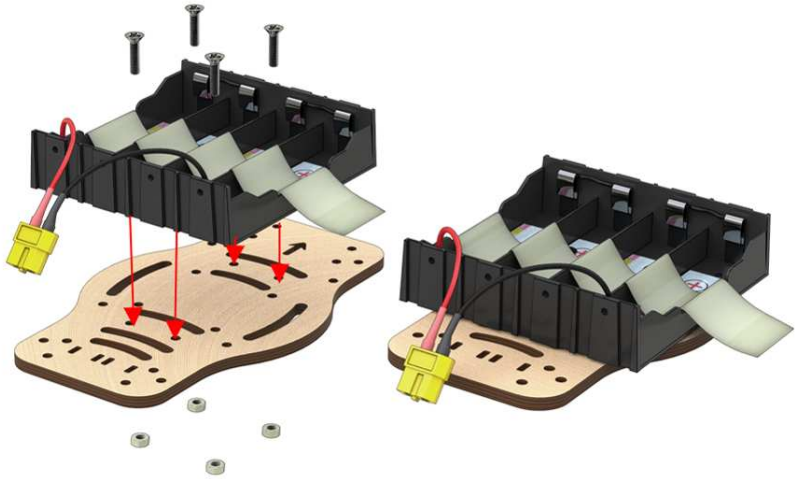


Монтаж отсека АКБ

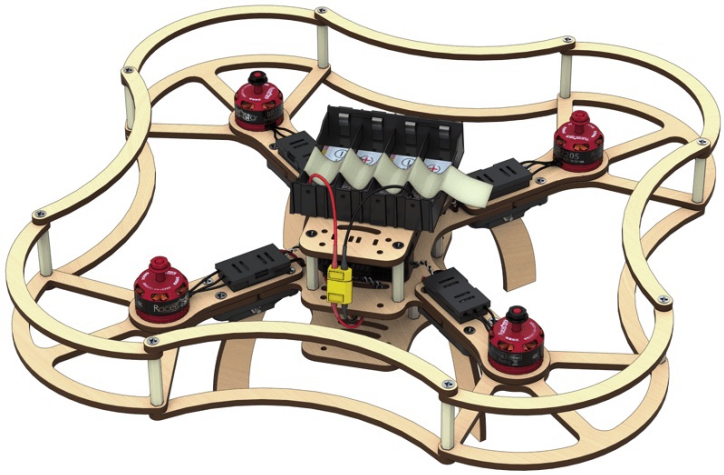
Необходимые компоненты:

- Винты M3x12 (4 шт)
- Гайки M3 (4 шт)
- Рама дополнительная (1 шт)
- Батарейный отсек (1 шт)

- Прикрепить батарейный отсек на верхнюю дополнительную раму винтами M3x12 и гайками.



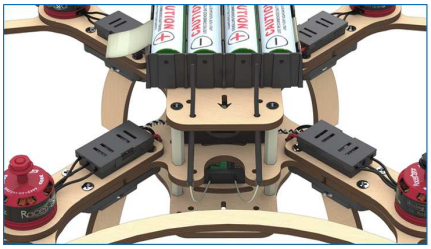
- Прикрепить верхнюю дополнительную раму на стойки винтами M3x8.

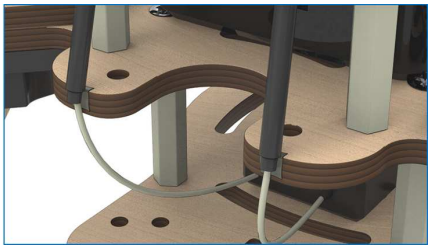


- Установить АКБ в отсек.

Монтаж антенн

1. Крепим антенны на 2х сторонний скотч или изоленту, а усики продеваем в передние отверстия верхней дополнительной рамы.

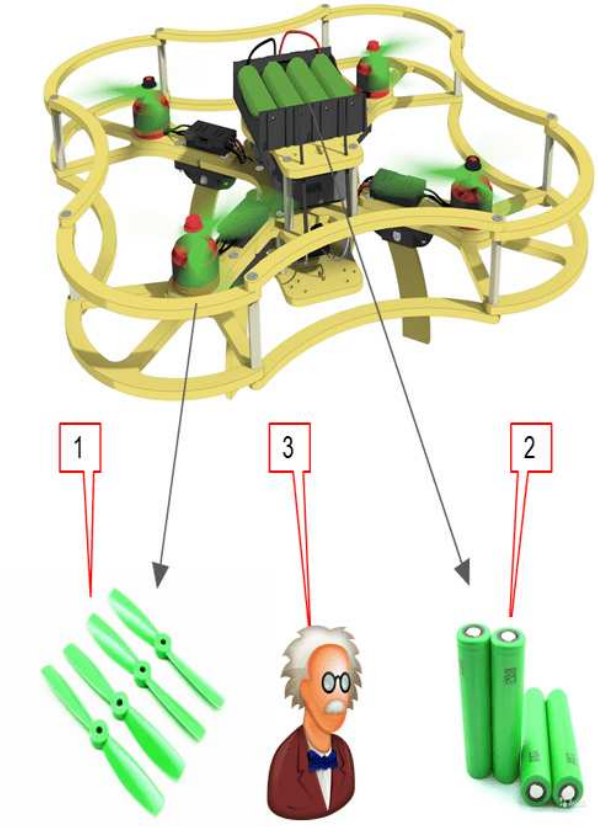




Коптер готов к настройке!

Безопасность при сборке и настройке

1. Снять пропеллеры.“Все наземные операции производить со снятыми пропеллерами. Устанавливать пропеллеры на моторы только перед полётом.”
2. Отключить аккумулятор. Держать питание выключенным. “Сборку, настройку и ремонт производить с отключенным питанием. Подключать питание только для тестирования электронных компонентов коптера. После тестирования перед другими работами питание сразу отключить.”
3. Позвать на помощь. “Если при выполнении работ возникли какие-либо проблемы, необходимо обратиться к преподавателю или учителю, а не пытаться решить проблему самостоятельно.”



Безопасность при работе с Li-ion аккумуляторами 18650

1. Обращаться с аккумуляторами бережно. Не допускать падений, ударов деформаций.
2. При подключении (отключении) аккумуляторов держаться только за разъёмы, тянуть или дергать за провода запрещается.
3. В случае обрыва разъемов, обнаружения нарушений целостности изоляции или корпуса аккумулятора, не трогая его, немедленно

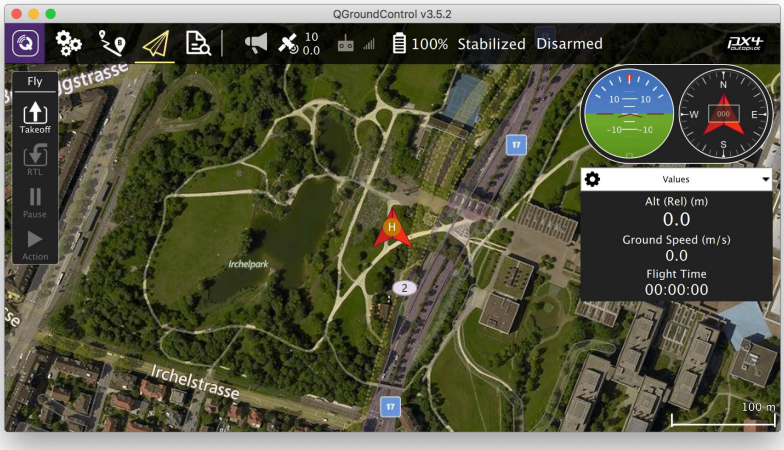
сообщить преподавателю.

См. статью техника безопасности при пайке и лётной эксплуатации коптеров.

Первоначальная настройка

Установка QGroundControl

QGroundControl – программное обеспечение, необходимое для прошивки, настройки и калибровки полетного контроллера Клевера.



Скачайте и установите установочный файл для Windows/Linux/macOS с официального сайта QGroundControl. В случае необходимости согласитесь с установкой дополнительных драйверов при установке.

См. также основную документацию по QGroundControl.

MicroSD-карта

Подготовьте MicroSD-карту для полетного контроллера.



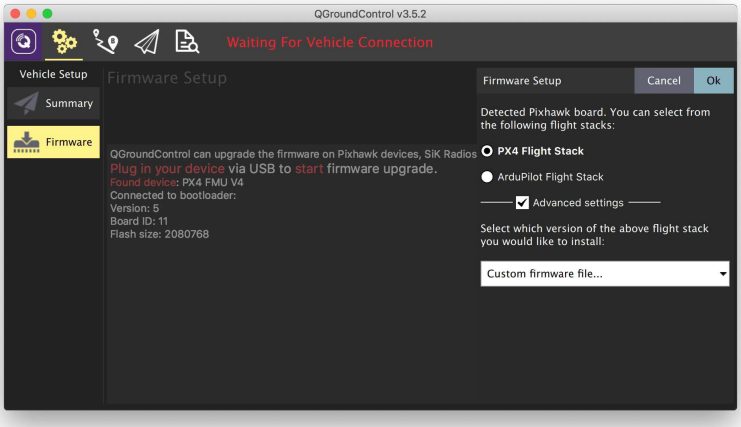
1. Установите карту в компьютер (используйте адаптер при необходимости).
2. Отформатируйте карту в файловую систему FAT32. Для этого кликните на значок SD-карты в "Проводнике" и нажмите "Форматирование" в Windows. Используйте "Дисковую утилиту" в macOS.
3. Выполните "Безопасное извлечение" карты, извлеките карту.
4. Установите карту в полетный контроллер.

Загрузка прошивки в полетный контроллер

Наиболее оттестированной, в особенности для осуществления автономных полетов, является версия прошивки с патчами COEX. Скачайте актуальную версию прошивки на GitHub — **скачать**.

Для использования всех наиболее актуальных функций PX4 вы также можете использовать последнюю официальную версию прошивки (в экспериментальном режиме).

1. Отключите полетный контроллер от компьютера (если он подключен).
2. Запустите программу QGroundControl.
3. Перейдите в панель *Vehicle Setup* (кликнув на логотип QGroundControl в левом верхнем углу) и выберите меню *Firmware*.
4. Подключите полетный контроллер к компьютеру по USB.
5. Выберите в появившемся меню справа *PX4 Flight Stack*.



6. Для загрузки **прошивки COEX**:
   - Выберите *Advanced settings*.
   - В выпадающем меню выберите *Custom firmware file...*
   - Нажмите *OK* и выберите скаченный файл прошивки.

   Для загрузки последней версии **стандартной прошивки** сразу нажмите *OK*.

Дождитесь, пока QGroundControl загрузит прошивку и выполнит перезагрузку полетного контроллера.

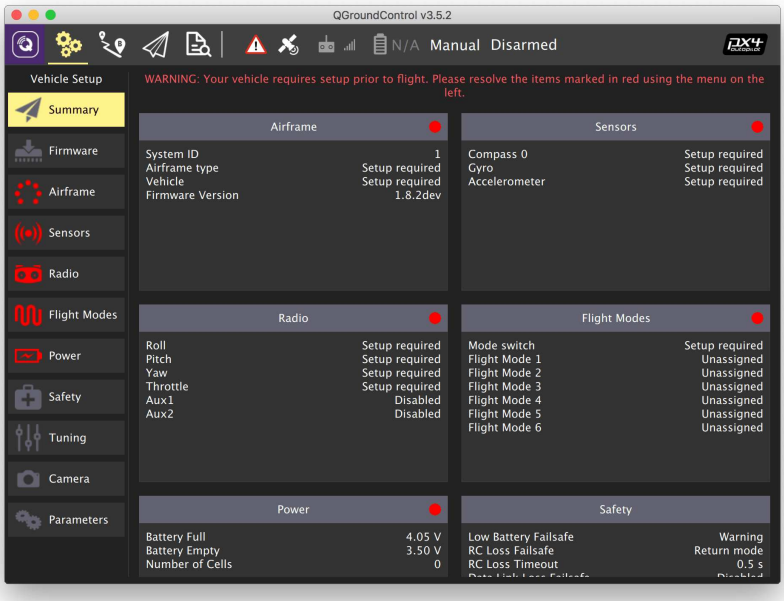
> Не отключайте полетный контроллер от компьютера в процессе загрузки прошивки.

Дополнительную информацию о прошивке читайте в статье "Прошивка Pixhawk".

Настройка полетного контроллера

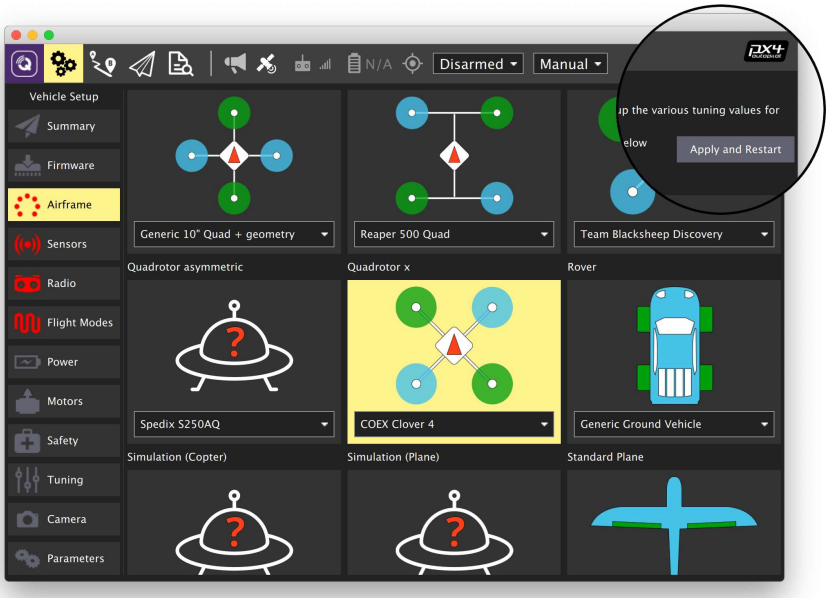
> Все дальнейшие настройки и калибровки полетного контроллера могут быть выполнены без проводов с применением доступа к полетному контроллеру по Wi-Fi через Raspberry Pi.

Обзор главного окна настроек QGroundControl:



1. Параметры, нуждающиеся в настройке: *Airframe*, *Radio*, *Sensors*, *Flight Modes*.
2. Текущая прошивка контроллера.
3. Текущий полетный режим.
4. Сообщения об ошибках.

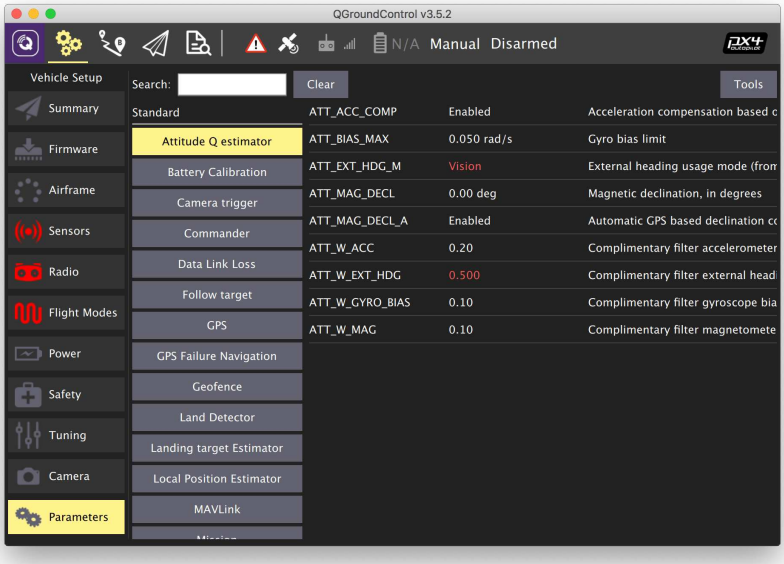
Выбор рамы



1. Зайдите во вкладку *Vehicle Setup*.
2. Выберите меню *Airframe*.
3. Выберите тип рамы *Quadrotor X*.
4. Для Клевера 4 выберите подтипа рамы *COEX Clover 4*. В ином случае – *Generic Quadrotor X*.
5. Переместитесь в начало списка и нажмите кнопку *Apply and Restart*, подтвердите нажатием *Apply*.
6. Дождитесь применения настроек и перезагрузки полетного контроллера.

Параметры

Для настройки параметров полетного контроллера войдите во вкладку *Vehicle Setup* и выберите меню *Parameters*. Вы можете использовать поле *Search* для поиска параметров по имени. Рекомендуемые параметры для Клевера приведены в дальнейшей документации а также в соответствующей сводной статье.



После установки параметра необходимо нажать кнопку *Save*. При необходимости – перезагрузить полетный контроллер, нажав кнопку *Tools*, затем *Reboot vehicle*.

Настройка PID-регуляторов

> Использование типа рамы *COEX Clover 4* не требует ввода коэффициентов PID.

Усредненные коэффициенты PID для Клевера 4

- `MC_PITCHRATE_P` = 0.087
- `MC_PITCHRATE_I` = 0.037
- `MC_PITCHRATE_D` = 0.0044
- `MC_PITCH_P` = 8.5
- `MC_ROLLRATE_P` = 0.087
- `MC_ROLLRATE_I` = 0.037
- `MC_ROLLRATE_D` = 0.0044
- `MC_ROLL_P` = 8.5
- `MPC_XY_VEL_P` = 0.11
- `MPC_XY_VEL_D` = 0.013
- `MPC_XY_P` = 1.1
- `MPC_Z_VEL_P` = 0.24
- `MPC_Z_P` = 1.2

Усредненные коэффициенты PID для Клевера 3

- `MC_PITCHRATE_P` = 0.145
- `MC_PITCHRATE_I` = 0.050
- `MC_PITCHRATE_D` = 0.0025
- `MC_ROLLRATE_P` = 0.145
- `MC_ROLLRATE_I` = 0.050

- `MC_ROLLRATE_D` = 0.0025

> Необходимо учитывать, что для идеального полета параметры PID-регуляторов подбираются вручную для каждого конкретного собранного квадрокоптера. Вы можете узнать больше об этом в статье "Настройка PID-регуляторов".

Параметры Circuit breaker

1. Чтобы коптер мог летать с подключением по USB, установите параметр `CBRK_USB_CHK` в 197848.
2. Отключите проверку Safety Switch: `CBRK_IO_SAFETY` = 22027.

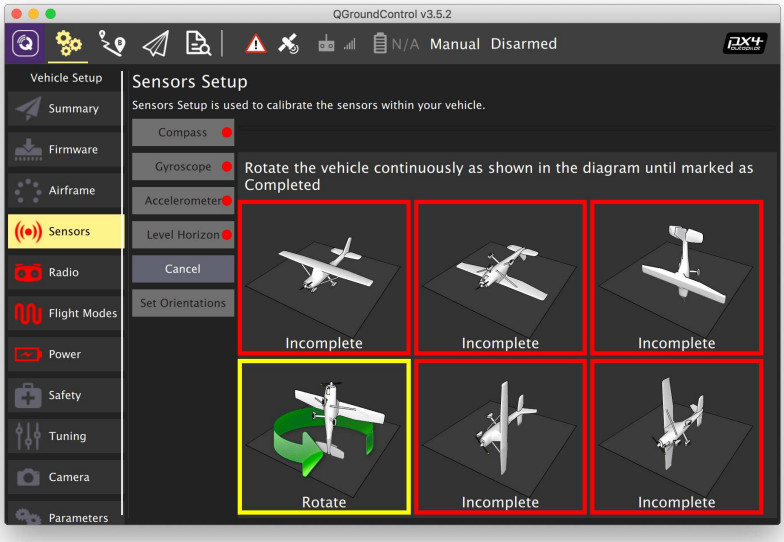
Далее: Калибровка датчиков.

Калибровка датчиков

Чтобы откалибровать датчики зайдите во вкладку *Vehicle Setup* и выберите меню *Sensors*.

> Если вы используете полетный контроллер *COEX Pix* и он установлен серворазъемами назад (как на изображениях в инструкции), то во всех графах *Autopilot Orientation* необходимо указать значение `ROTATION_ROLL_180_YAW_90` , иначе полетный контроллер будет некорректно воспринимать наклоны и повороты коптера.

Компас

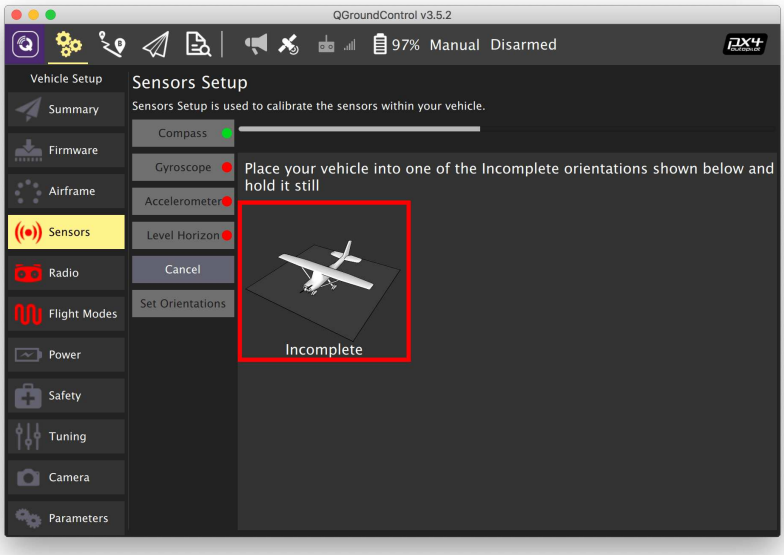


1. Выберите меню *Compass*.
2. Выберите ориентацию полетного контроллера – *ROTATION_NONE* при условии, что полетный контроллер ориентирован передом к носу квадрокоптера.
3. Нажмите *OK*.
4. Последовательно устанавливайте квадрокоптер в каждую из указанных ориентаций до появления желтой рамки.
5. Вращайте квадрокоптер по направлению стрелки до появления зеленой рамки.

> Последние версии прошивки PX4 не поддерживают внутренний компас на полетном контроллере COEX Pix. При появлении ошибки *No mags found* перейдите во вкладку *Parameters*, установите параметры `SYS_HAS_MAG` в `0` , `EKF2_MAG_TYPE` в `None` и перезагрузите полетный контроллер (*Tools* => *Reboot Vehicle*).

Дополнительная информация:
https://docs.px4.io/master/en/config/compass.html.

Гироскоп

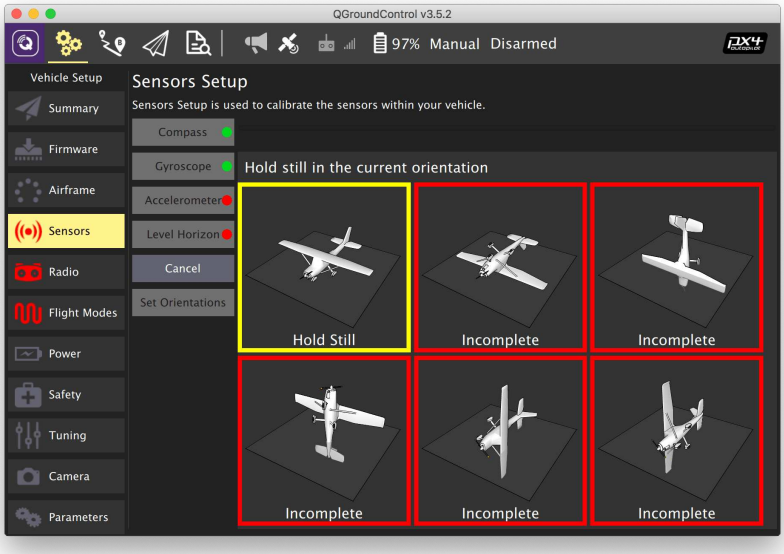


1. Выберите меню *Gyroscope*
2. Установите квадрокоптер на ровную поверхность.
3. Нажмите *OK*.
4. Дождитесь окончания калибровки.

> Во время калибровки гироскопа квадрокоптер не должен менять своего положения, шататься и т. д.

Дополнительная информация:
https://docs.px4.io/master/en/config/gyroscope.html.

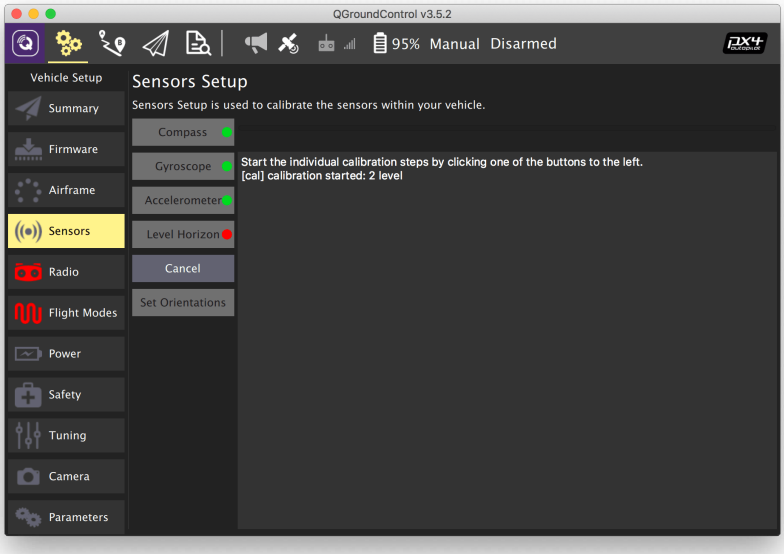
Акселерометр



1. Выберите меню *Accelerometer*.
2. Выберите ориентацию полетного контроллера – *ROTATION_NONE* при условии, что полетный контроллер ориентирован передом к носу квадрокоптера.
3. Последовательно устанавливайте квадрокоптер в каждую из указанных ориентаций до появления желтой рамки.
4. Держите квадрокоптер неподвижно до появления зеленой рамки.

Дополнительная информация:
https://docs.px4.io/master/en/config/accelerometer.html.

Уровень горизонта



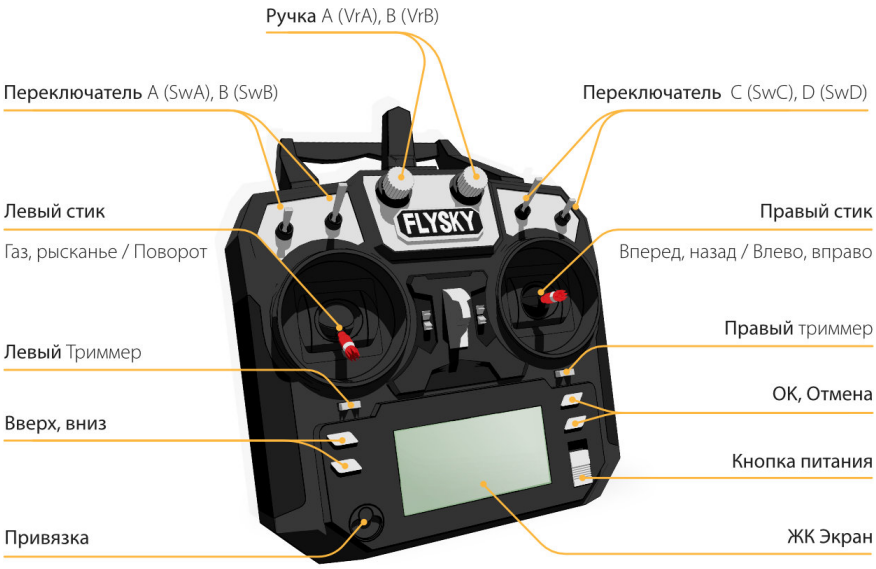
1. Выберите меню *Level Horizon*.

2. Выберите ориентацию полетного контроллера – *ROTATION_NONE* при условии, что полетный контроллер ориентирован передом к носу квадрокоптера.
3. Установите квадрокоптер на ровную поверхность.
4. Нажмите *OK*.
5. Дождитесь окончания калибровки.

Дополнительная информация:
https://docs.px4.io/master/en/config/level_horizon_calibration.html.

Далее: Настройка пульта.

Настройка пульта



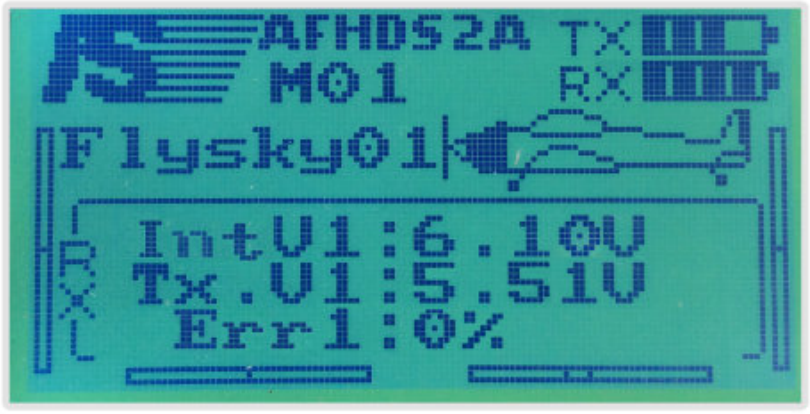
Перед подключением и калибровкой пульта убедитесь, что:

- К коптеру не подключено внешнее питание АКБ.
- Пропеллеры не установлены на моторах.

Подключение пульта

1. В программе QGroundControl перейдите в панель *Vehicle Setup* и выберите меню *Radio*.
2. Включите пульт, переводя переключатель *POWER* в верхнее положение.
3. Убедитесь, что связь с приемником установлена.

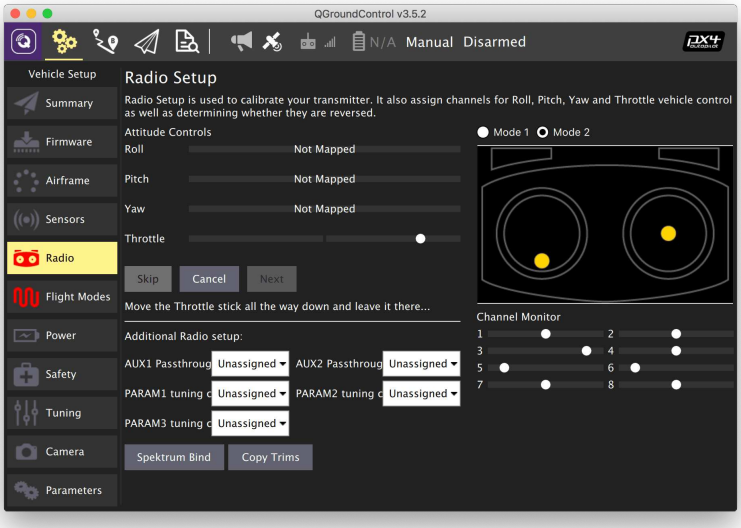
   На ЖК Экране пульта высвечивается индикация:



   Светодиод на приемнике должен гореть непрерывно красным. При наличии проблем с подключением читайте статью "Неисправности радиоаппаратуры".

Калибровка пульта

1. Нажмите кнопку *Calibrate*.
2. Установите триммеры *Throttle*, *Yaw*, *Pitch*, *Roll* в 0.
   - Триммеры позволяют задавать смещение коптеру.
   - Чтобы установить один из триммеров в 0, необходимо на пульте переместить указатель в центр до длительного звукового сигнала (писка).
3. Нажмите *OK*.



4. Переведите левый стик (газа) в минимум и нажмите *Next*.
5. Повторяйте движения стиками вслед за анимацией и читайте подсказки.
6. При появлении надписи *"Move all transmitter switches and/or dials back and forth to their extreme positions"* переключите SwA. SwD, VrA, VrB в их конечные положения.
7. Нажмите *Next*.
8. При появлении надписи *"All settings have been captured. Click Next to write the new parameters to your board"* нажмите *Next*.

Дополнительная информация:
https://docs.qgroundcontrol.com/en/SetupView/Radio.html.

Далее: Настройка полетных режимов.

Работа с приёмником Flysky FS-A8S

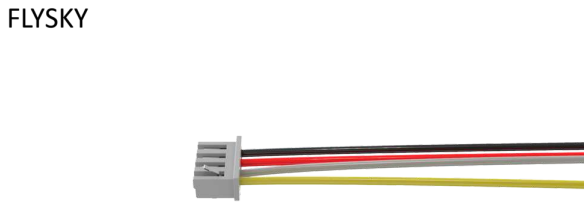
Приёмник Flysky FS-A8S совместим с пультами Flysky FS-i6 и FS-i6x. Связь с полётным контроллером может происходить как с использованием аналогового протокола PPM, так и при помощи цифровых протоколов S.Bus/i-Bus.

Для подключения к полётному контроллеру рекомендуется использовать протокол S.Bus.

Изготовление кабеля для подключения к полётному контроллеру

> Если в вашем наборе уже есть кабель для подключения приёмника к полётному контроллеру, совместимому с вашим, переходите к следующему этапу.

1. Аккуратно извлеките жёлтый провод из коннектора, идущего к приёмнику. Для того, чтобы вытащить провод, приподнимите острым пинцетом замок коннектора:



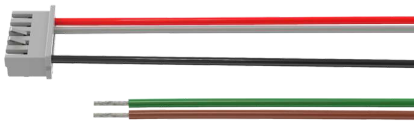


2. [При использовании полётного контроллера Pixracer] Извлеките 2 провода - зелёный и коричневый - из 5-пинового коннектора для полётного контроллера:

PIXRACER



PIXRACER



3. [При использовании полётного контроллера COEX Pix] Извлеките зелёный (или синий, в зависимости от комплектации) провод из 4-пинового коннектора для полётного контроллера:

COEX PIX

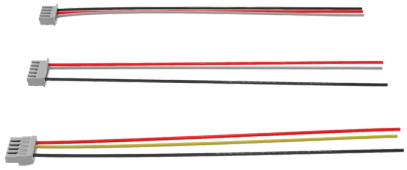


COEX PIX



4. С помощью бокорезов обрежьте концы кабелей с разъёмами Dupont (чёрными):





5. Зачистите и залудите 5-7 мм провода с каждой стороны:



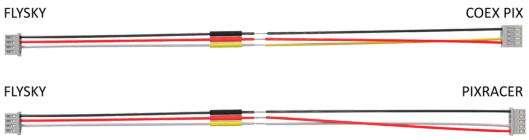
6. Наденьте термоусадочные трубки на провода:

FLYSKY

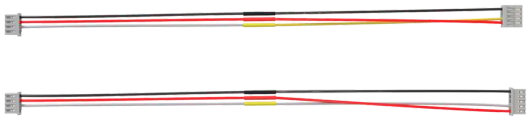


7. Спаяйте провода по следующей схеме:

- чёрный провод приёмника с чёрным проводом кабеля полётного контроллера;
- красный провод приёмника с красным проводом кабеля полётного контроллера;
- белый провод приёмника с белым (при использовании Pixracer) или жёлтым (при использовании COEX Pix) проводом кабеля полётного контроллера:



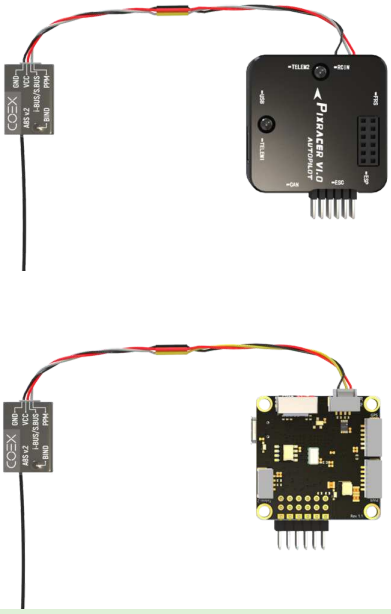
8. Переместите термоусадочные трубки на места пайки и прогрейте их феном:



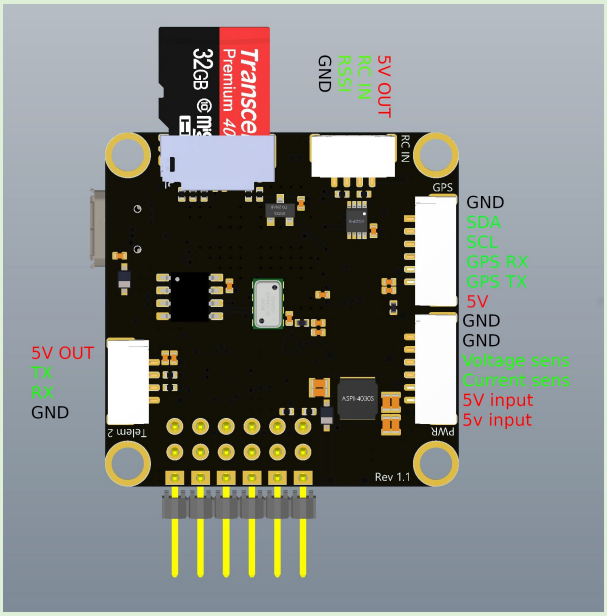
9. Скрутите готовые кабели:



С помощью изготовленного кабеля подключите приёмник к полётному контроллеру к порту RC IN:



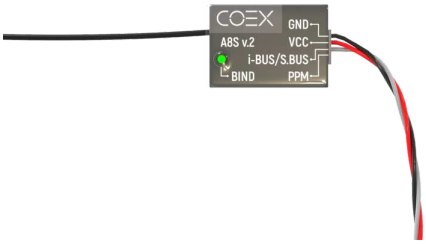
Убедитесь, что провод, идущий в COEX Pix, подключен к порту RC IN:



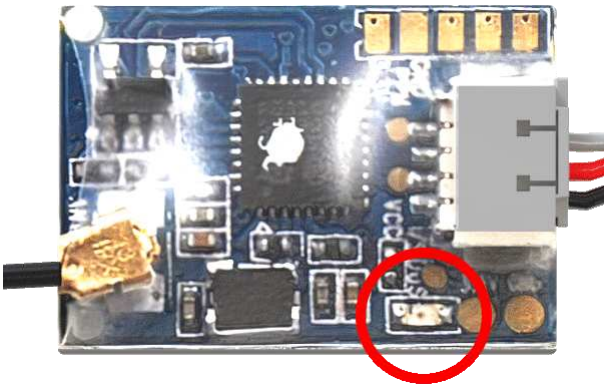
Сопряжение приёмника с пультом

Для сопряжения приёмника с пультом:

1. Убедитесь, что полётный контроллер выключен.
2. Зажмите кнопку **BIND** на приёмнике:



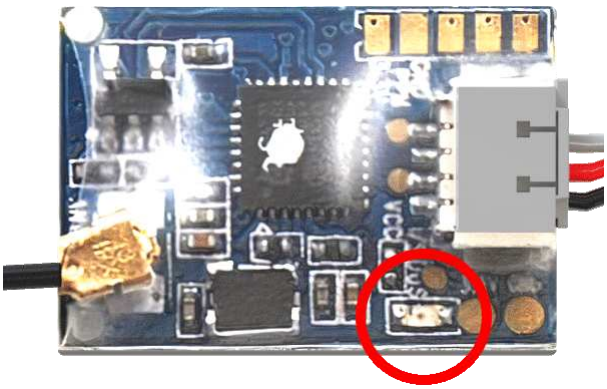
3. Включите полётный контроллер. Светодиод на приёмнике должен замигать с высокой частотой.



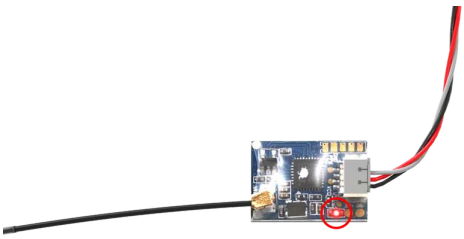
4. Зажмите клавишу **BIND KEY** на пульте и включите его. На пульте должно появиться сообщение **RX Binding...**



5. Светодиод на приёмнике должен замигать с низкой частотой.



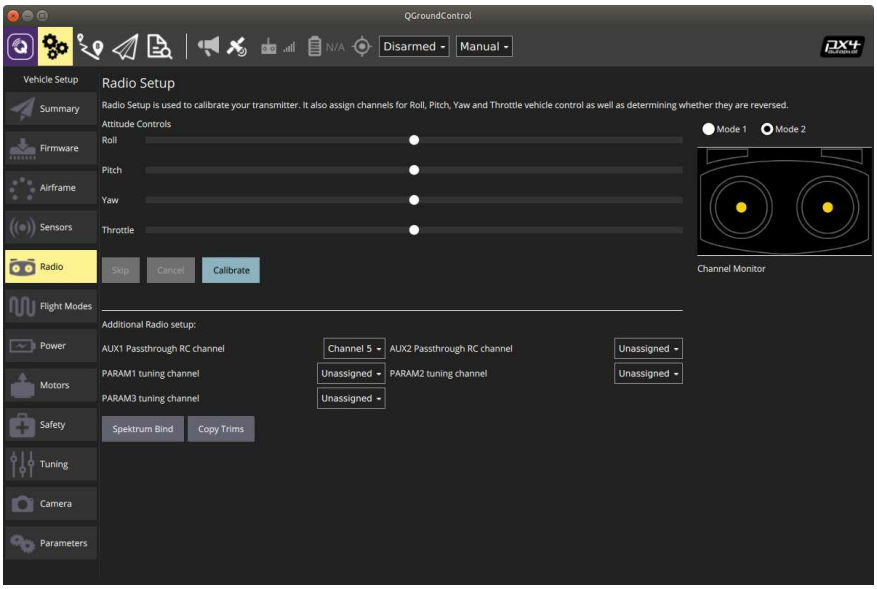
6. Выключите и включите пульт. Светодиод на приёмнике должен светиться непрерывно.



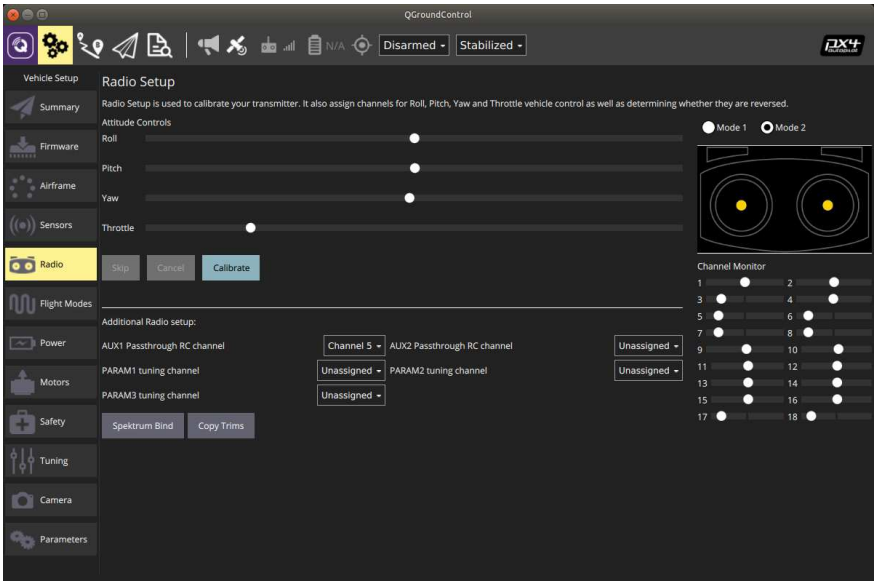
> Данный приёмник не передаёт телеметрию обратно на пульт. Это значит, что на экране пульта не будет отображаться информация об уровне сигнала и напряжении на приёмнике. Эта особенность приёмника не является неисправностью и не влияет на управление.

Выбор режима приёмника

Откройте QGroundControl и подключите полётный контроллер к компьютеру. Откройте вкладку Radio:



Если справа (под изображением пульта) не показано ни одного канала, зажмите кнопку **BIND** на приёмнике на 2 секунды. Должно появиться 18 каналов:


QGroundControl
Disarmed
Stabilized
Vehicle Setup
Summary
Firmware
Airframe
Sensors
Radio
Flight Modes
Power
Motors
Safety
Tuning
Camera
Parameters
Radio Setup
Radio Setup is used to calibrate your transmitter. It also assign channels for Roll, Pitch, Yaw and Throttle vehicle control as well as determining whether they are reversed.
Attitude Controls
Roll
Pitch
Yaw
Throttle
Skip
Cancel
Calibrate
Mode 1
Mode 2
Channel Monitor
Additional Radio setup:
AUX1 Passthrough RC channel
Channel 5
AUX2 Passthrough RC channel
Unassigned
PARAM1 tuning channel
Unassigned
PARAM2 tuning channel
Unassigned
PARAM3 tuning channel
Unassigned
Spektrum Bind
Copy Trims

Полетные режимы

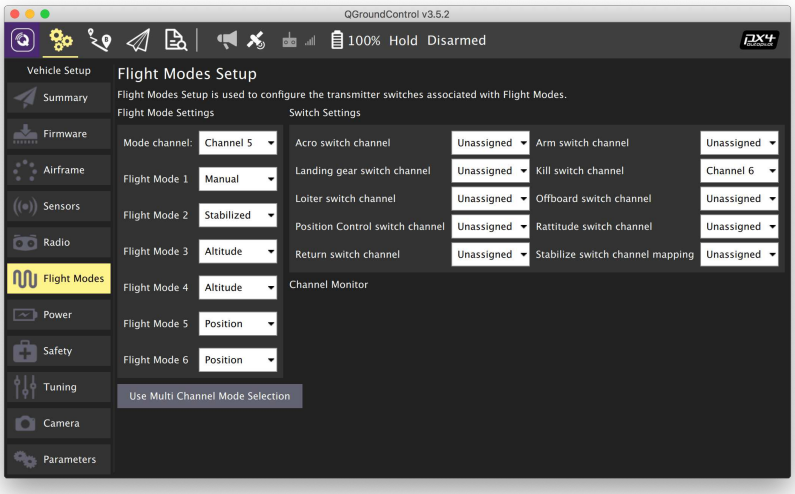
Режим полетного контроллера PX4 определяет, как именно квадрокоптер (или другой аппарат) должен себя вести: каким образом интерпретировать входящие команды и сигналы с пульта. Режим переключается одним из переключателей на пульте радиоуправления.

Чтобы настроить полетные режимы:

1. В программе QGroundControl перейдите в панель *Vehicle Setup*.
2. Выберите меню *Flight Modes*.
3. Установите переключатель режимов (*Mode Channel*) на переключатель SwC (*Channel 6*).
4. Опционально, установите экстренное отключение пропеллеров (*Emergency Kill Switch Channel*) на переключатель SwA (*Channel 5*).
5. Выберите необходимые полетные режимы.

   Рекомендуемые полетные режимы:

   - Flight Mode 1: *Stabilized*.
   - Flight Mode 4: *Altitude*.
   - Flight Mode 6: *Position*.
6. Проверьте корректность переключения режимов, переключая переключатель на пульте.
7. Назначьте аварийное отключение моторов (*Kill switch*) на переключатель SwA (Channel 5).



Подробное описание полетных режимов

Ручное управление

При ручном управлении пилот управляет квадрокоптером напрямую. GPS, данные с компьютерного зрения и барометр не используются. Для полетов в этих режимах необходимы хорошие навыки пилотирования мультикоптеров.

- **STABILIZED**/**MANUAL** — режим стабилизации горизонтального положения. Управление газом, углами наклона коптера по тангажу и крену, угловой скоростью по рысканью.
- **ACRO** — управление газом и угловой скоростью коптера по тангажу, крену и рысканью. Используется дрон-рейсерами и в шоу 3D-пилотирования для выполнения трюков.
- **RATTITUDE** — в центре правый стик аналогичен STABILIZED, по краям переходит в режим ACRO.

С использованием дополнительных датчиков

- **ALTCTL** (*Altitude*) — управление скоростью изменения высоты полета, углами по тангажу и крену и угловой скоростью по рысканью. Используется барометр (или иной датчик высоты).
- **POSCTL** (*Position*) — управление скоростями набора высоты, скоростью движения вперед/назад и вправо/влево, угловой скоростью по рысканью. Наиболее простой для полетов режим. Используется барометр, GPS, компьютерное зрение, другие датчики.

Автоматический полет

В этих режимах квадрокоптер игнорирует сигналы с пульта и летает по какой-либо автоматической программе.

- **OFFBOARD** — управление полетом с внешнего компьютера (например, Raspberry Pi). Этот режим используется в Клевере для программирования автономных полетов.
- **AUTO.MISSION** – квадрокоптер выполняет заранее загруженную в квадрокоптер миссию. Миссия загружается при помощи QGroundControl, или по MAVLink. Этот режим чаще всего применяется для автоматических полетов по точкам с использованием GPS, например, для фотограмметрии.
- **AUTO.RTL** – коптер автоматически возвращается в точку взлета.
- **AUTO.LAND** – коптер выполняет посадку.

Дополнительная информация: https://dev.px4.io/en/concept/flight_modes.html.

Далее: Настройка питания.

Настройка питания

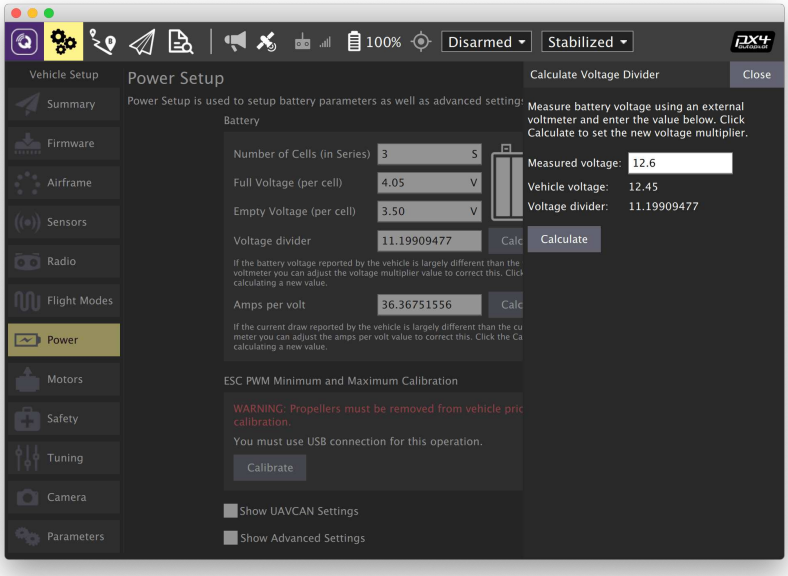
Чтобы откалибровать параметры, связанные с электропитанием, зайдите во вкладку *Vehicle Setup* и выберите меню *Power*.

Калибровка делителя напряжения

> Калибровка делителя напряжения должна выполняться с подключенным АКБ.

1. В программе QGroundControl перейдите в панель *Vehicle Setup* и выберите меню *Power*.
2. Установите параметр *Number of cells* в соответствии с количеством банок в АКБ (*3S* для Клевера 4).
3. Откалибруйте делитель напряжения:
   - Подключите индикатор напряжения к балансировочному разъему АКБ.
   - Нажмите кнопку *Calculate* напротив надписи *Voltage Divider*.
   - Введите в открывшемся поле суммарное значение напряжения с индикатора напряжения.
   - Нажмите *Close*, чтобы сохранить рассчитанное значение.

В случае отсутствия индикатора напряжения или невозможности ручной калибровки, установите усредненное значение делителя напряжения для комплекта Клевер 4 (*Voltage divider* = 11).

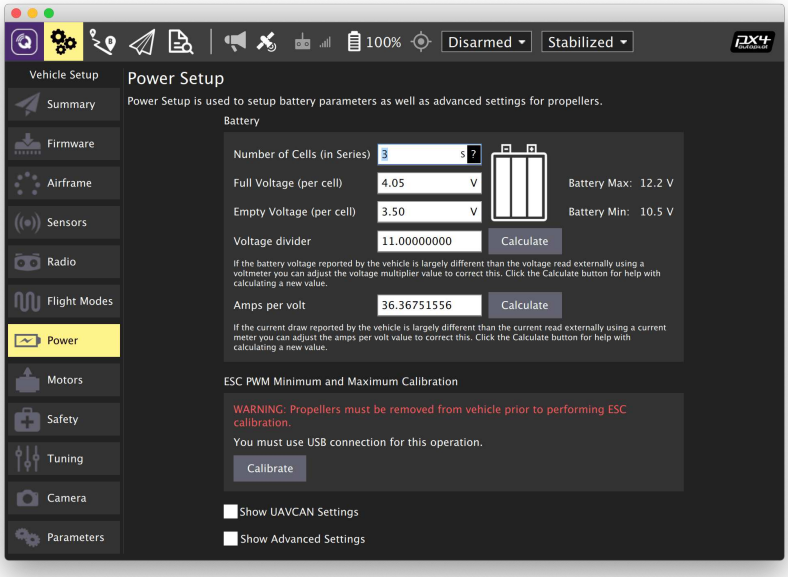


Дополнительная информация:
https://docs.qgroundcontrol.com/en/SetupView/Power.html.

Калибровка регуляторов (ESC)

> **Никогда не выполняйте калибровку регуляторов с установленными пропеллерами.** В некоторых случаях моторы могут начать вращаться на максимальной скорости.

1. Убедитесь, что АКБ не подключена и пропеллеры сняты
2. Нажмите *Calibrate*.
3. После появлении надписи *Connect the battery now* подсоедините АКБ.
4. Дождитесь появления надписи *Calibration complete*.



Дополнительная информация:
https://docs.px4.io/master/en/advanced_config/esc_calibration.html.

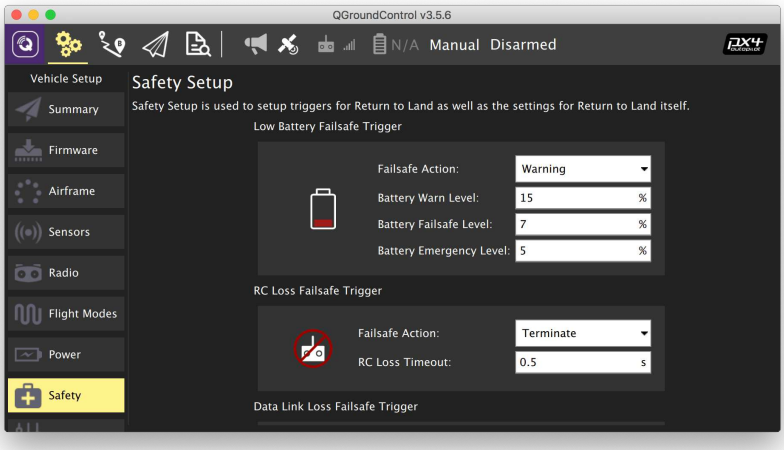
Далее: настройка Failsafe.

Настройка Failsafe

Основная статья: https://docs.px4.io/master/en/config/safety.html.

Во вкладке *Safety* настраиваются реакции квадрокоптера на различные нештатные ситуации. Рекомендуется включить как минимум реакцию на потерю связи с пультом управления:

1. В программе QGroundControl перейдите в панель *Vehicle Setup* и выберите меню *Safety*.
2. В блоке *RC Loss Failsafe Trigger* выберите один из рекомендуемых вариантов реакции на потерю связи с пультом:
   - *Land mode* – переход в режим посадки;
   - *Terminate* – аварийное отключение моторов.
3. В поле *RC Loss Timeout* выберите значение таймаута, по истечении которого связь с пультом считается потерянной. Рекомендуемое значение – 2 s.



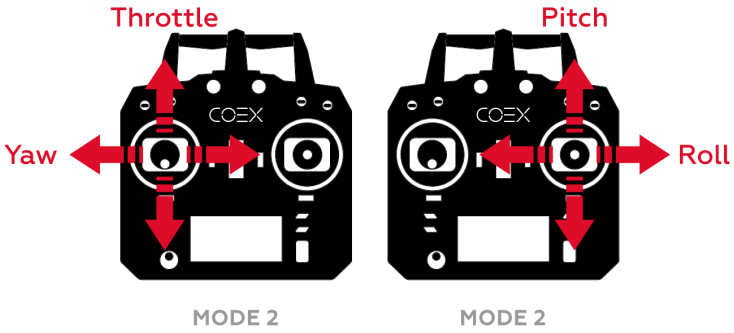
Полет

Этот раздел объясняет основы управление квадрокоптером с использование пульта радиоуправления в различных режимах (для автономных полетов смотрите раздел "Программирование").

Основные возможности радиоаппаратуры

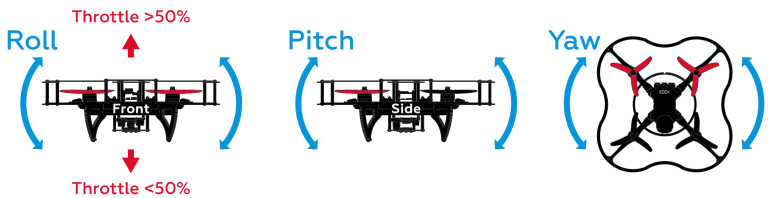
Прежде чем запускать ваш коптер, необходимо разобраться, как работает пульт радиоуправления ("аппаратура" в общепринятой терминологии авиамоделизма).

Управление дроном происходит с помощью двух стиков на аппаратуре. По умолчанию левый стик отвечает за газ и рысканье, а правый за крен и тангаж. Данные термины используются для всех летательных судов, от самолетов до квадрокоптеров.



- Газ (*throttle*) – отвечает за скорость вращения двигателей.
- Рысканье (*yaw*) – отвечает за повороты вокруг вертикальной оси (Z), по часовой (при наклоне вправо) и против часовой (при наклоне влево) стрелки.
- Тангаж (*pitch*) – отвечает за наклон или движение вперёд/назад.
- Крен (*roll*) – отвечает за наклон или движение влево/вправо.

Данные описания предполагают, что коптер находится задней частью к пилоту.



Полетные режимы

Ручное полет с использованием полетного контроллера PX4 может происходит с использованием разных полетных режимов, которые определяют назначения стиков радиопульта и другие характеристики полета. Полный список полетных режимов приведен в статье "Полетные режимы".

Основные ручные режимы разобраны далее.

STABILIZED - режим стабилизации горизонтального положения. В данном режиме коптер будет удерживать горизонт, если им не управлять. Назначение стиков:

- Газ – усредненная скорость вращения моторов.
- Рысканье – угловая скорость вокруг вертикальной оси.
- Тангаж – угол наклона вокруг поперечной оси (вперед/назад).
- Крен – угол наклон вокруг продольной оси (влево/вправо).

POSCTL – режим удержания позиции (требуется включенная система позиционирования). Назначение стиков:

- Газ - вертикальная скорость полета.
- Рысканье - угловая скорость вокруг вертикальной оси.
- Тангаж - линейная скорость полета дрона (вперед/назад).
- Крен - линейная скорость полета дрона (влево/вправо).

ACRO – режим управление средней скоростью вращения моторов и угловыми скоростями дрона. Этот режим является наиболее сложным для пилотирования и чаще всего применяется дрон-рейсерами и в шоу 3D-пилотирования для выполнения трюков. Назначение стиков:

- Газ – усредненная скорость вращения моторов.
- Рысканье – угловая скорость вокруг вертикальной оси.
- Тангаж – угловая скорость вокруг поперечной оси (вперед/назад).
- Крен – угловая скорость вокруг продольной оси (влево/вправо).

> В других полетных контроллерах аналогичные полетные режимы могут называться по-другому.

Подготовка к полету

Установка пропеллеров и АКБ

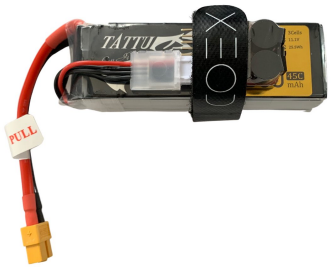
1. Установите ремешок для аккумулятора.



2. Установите пропеллеры в соответствии со схемой направления движения моторов.



3. Закрепите пищалку и установите аккумулятор.





Настройка пищалки

Для того, чтобы не переразрядить и не испортить аккумулятор, рекомендуется использовать индикатор напряжения (*пищалка*).

Для настройки *пищалки* подключите ее к балансировочному разъему вашего аккумулятора. Теперь, нажимая на кнопку в основании будет изменяться минимальное напряжение на ячейках. Оптимальное значение минимального напряжения является *3.5-3.6 V*.





Состояния готовности к полету

Прежде чем начинать полет, необходимо перевести коптер в состояние *Armed*.

- Состояние *Armed* – моторы вращаются в соответствии с положением стика газа, коптер готов к полету.
- Состояние *Disarmed* – моторы не вращаются, коптер не реагирует на стик газа.

По умолчанию коптер находится в состоянии *Disarmed* и переходит в него в случае если вы долго не взлетаете.

Для перевода коптера в состояние *Armed* есть несколько способов:

- С помощью стика – переведите левый стик вниз вправо и подождите пару секунд.



- С помощью тумблера – состояния Armed/Disarmed можно настроить на один из тумблеров. Подробнее о настройке в смотрите в статье про полетные режимы.
- С помощью QGC – вы можете заармить ваш дрон программно. Для этого нажмите на надпись *Disarmed* в шапке и выберите другое состояние.
- С помощью программы - коптер может перейти в состояние *Armed*, если в навигационной команде, такой как `navigate`, `set_position` и т.д., указан параметр `auto_arm=True`.

Kill switch

При активации тумблера *Kill Switch* на моторы перестают посылаться сигналы управления, и моторы перестают вращаться. Эта функция используется в крайних случаях, к примеру, если вы потеряли управление над коптером.

> Будьте внимательны, *Kill Switch* не переводит коптер в состояние *Disarmed*!

Перед отключением *Kill Switch* убедитесь, что стик газа находится в нижнем положении и коптер находится в состоянии *Disarmed*. В случае, если стик газа не находится в нижнем положении, при отключении *Kill Switch* на моторы будет подан сигнал соответствующий положению стика в данный момент, что приведет к резкому рывку коптера.

Далее: Упражнения для управления коптером.

Упражнения для управления коптером

Далее описаны рекомендуемые упражнения для тех, кто учится летать на коптере в первый раз. Повторяйте каждое упражнение необходимое количество раз, пока не будете чувствовать себя уверенно в нем.

> В случае, если рядом есть человек умеющий управлять коптером, используйте режим тренера.

> Настоятельно рекомендуется первые полеты проводить за защитной сеткой. В случае отсутствия таковой, полетная зона должна быть не менее 6x6 м.

Включение, выключение моторов, изменение режимов

> Для удобства подключитесь к коптеру с помощью QGC через Wi-Fi и включите звук. Это позволит наблюдать за изменением полетных режимов. Если не имеется возможности подключиться через Wi-Fi, для проверки полетных режимов подключитесь по USB.

Убедитесь, что настроили полетные режимы на один из тумблеров. Для этого переключите тумблер, в разные позиции и убедитесь, что режимы изменяются.

Рекомендуется настроить *Kill Switch*. Для его проверки совершите следующие действия:

- Включите *Kill Switch*, проверьте, что в QGC появилось соответствующее уведомление.
- Переведите коптер в состояние *Armed*, а затем включите *Kill Switch*. Убедитесь, что моторы выключились. Затем переключите тумблер *Kill Switch* в изначальное положение. Если коптер автоматически не перешел в состояние *Disarmed* из-за бездействия, моторы снова начнут вращаться.

> Переводите коптер в состояние *Armed* только на полетной зоне.

Убедитесь, что режимы переключаются удобными для вас тумблерами. В противном случае измените их в соответствии со статьей по настройке полетных режимов. Повторите приведенные действия несколько раз, для того, чтобы запомнить какие тумблеры за что отвечают.

Работа с газом

Первым делом необходимо почувствовать отзывчивость коптера на движение стика газа и научиться им управлять. Каждый коптер имеет немного различные запасы мощности и соответственно отрывается от земли при разных положениях стика.

В данном упражнении необходимо использовать только стик газа. Во время выполнения упражнения рекомендуется не использовать остальные стики.

Основные задания упражнения:

1. Дрейф коптера по земле, не отрываясь от земли.

Предполетные проверки

Перед взлетом выполняйте следующие действия:

1. Проверьте целостность коптера и возможность вращения пропеллеров.
2. Убедитесь, что коптер находится задней частью к вам.
3. Включите коптер путем подключения АКБ.
4. Отойдите на безопасное расстояние. Рекомендуется соблюдать расстояние до коптера минимум 4–5 м.
5. Убедитесь, что коптер находится в режиме *Stabilized*.

Не пытайтесь сразу оторвать коптер от земли, найдите минимально возможное положение стика для того, чтобы коптер начал дрейфовать по земле. В противном случае это может привести к поломкам или травмам.

> В случае потери контроля над коптером необходимо сразу включать *Kill Switch*.

Упражнение №1. Медленно поднимайте стик газа вверх, пока коптер не начнет двигаться. В этот момент он начнет медленно дрейфовать по земле. Оставьте стик газа в таком положении и подождите пару секунд, затем переведите стик в изначальное положение, чтобы посадить коптер. После посадки коптера выключите моторы переведя в состояние *Disarmed*. Повторите упражнение 5-10 раз, чтобы лучше чувствовать отзывчивость коптера на стик газа.

Упражнение №2. Медленно поднимайте стик газа вверх, пока коптер на начнет немного отрываться от земли. Оставьте стик газа в таком положении и подождите пару секунд, затем посадите коптер аналогично упражнению №1. Повторите упражнение 10-15 раз.

Упражнение №3. Поднимайте стик газа, пока коптер не начнет дрейфовать по земле, подождите секунду и продолжайте увеличивать газ до момента, когда коптер начнет отрываться от земли, подождите секунду и посадите коптер. Для закрепления повторяйте упражнения 10-15 раз, при необходимости увеличивая количество повторений.

Работа с креном и тангажом

После освоения управления газом коптера, необходимо научиться управлять его горизонтальным положением. За это отвечает правый стик на радиоаппаратуре.

Управление данными осями интуитивно понятно:

- Стик наклонен вперед (вверх) – коптер движется вперед.
- Стик наклонен назад (вниз) – коптер движется назад.
- Стик наклонен вправо – коптер движется вправо.
- Стик наклонен влево – коптер движется влево.

Чем сильнее стик будет наклонен в сторону, тем сильнее коптер будет наклоняться в сторону быстрее двигаться.

Основные задания упражнения:

1. Полет по оси X, вперед/назад.
2. Полет по оси Y, влево/вправо.
3. Стабилизация коптера на одном месте.
4. Полет по квадрату по часовой стреле и против.

> Старайтесь всегда находится позади коптера, таким образом, чтобы его задняя часть была направлена к вам, иначе вы можете потерять управление над ним, перепутав стороны.

Как и в случае с управлением газом, перед полетом выполняйте следующие действия.

> Если коптер сильно вращается вокруг своей оси, посадите его и повторно откалибруйте магнитометр и гироскоп.

Упражнение №1. Аналогично упражнениям по управлению газом поднимайте стик газа, пока коптер на начнет дрейфовать по земле или немного подпрыгивать, затем отпустите стик газа, оставив его в таком положении, и поднимайте стик тангажа, сначала вверх, на протяжении секунды, затем вниз. При этом коптер будет постепенно перемещаться сначала от вас, а затем к вам. Повторите упражнение 5-10 раз, пока не почувствуете отзывчивость коптера на движение стика.

Упражнение №2. Поднимайте стик газа, пока коптер не начнет дрейфовать, затем оставьте его и перемещайте стик крена сначала вправо, на протяжении секунды, затем влево. При этом коптер будет постепенно перемещаться сначала вправо, а затем влево. Повторите упражнение 5-10 раз, пока не почувствуете отзывчивость коптера на движение стика.

Упражнение №3. Поднимайте стик газа, пока коптер не начнет дрейфовать, затем оставьте его. Совместите первое и второе упражнение и постарайтесь стабилизировать коптер в одной точке, компенсируя его дрейф с помощью стика. Удерживайте коптер 20-30 секунд.

Упражнение №4. Поднимайте стик газа, пока коптер не начнет дрейфовать, затем оставьте его. Почувствовав отзывчивость коптера на изменения стиков выполните фигуру "квадрат" со стороной 1 м, сначала по часовой стрелке, а

затем против. Выполняйте фигуры по 2-3 раза.

Воздушная подушка и управление в ней

Понятие *воздушной подушки* очень важно во всей летательной технике. Сама по себе воздушная подушка является зоной повышенного давления, возникающая за счет воздуха пропускаемого через пропеллеры. Данная область характеризуется турбулентностями и воздушными потоками влияющими на полет коптера.

Пилоты стараются избегать полетов в воздушной подушке, но на ее границе имеется стабильная область, в которой коптер может зависнуть при минимальном значении газа. В таком случае создается ощущение, что коптер "сел" на воздушную подушку.

Главная особенность и преимущество такого полета заключается в том, что коптер не будет изменять высоту при одном значении газа.

Основные задания:

1. Стабилизация коптера на одном месте.
2. Полет по квадрату.
3. Полет по кругу.

Аналогично с предыдущими упражнениями перед взлетом выполните следующие действия.

Упражнение №1. Поднимайте стик газа, пока коптер не пролетит воздушную подушку и не окажется над ней (высота от пола ~25-30 см, для коптера Клевер 4). Коптер не должен подниматься вверх или проваливаться вниз, высота полета должна стабилизироваться. Как и в предыдущем упражнении корректируйте позицию коптера по осям X, Y с помощью стика крена и тангажа. В результате коптер должен зависнуть в одной точке с небольшими покачиваниями по сторонам. Удерживайте коптер 30-40 секунд.

Упражнение №2. Поднимите коптер на воздушную подушку и стабилизируйте его в одной точке. Далее пролетите по квадрату со стороной 1 м сначала по часовой стрелке, потом против часовой стрелки. Повторите траекторию в каждую сторону 2-3 раза.

Упражнение №3. Поднимите коптер на воздушную подушку и стабилизируйте его в одной точке. Попробуйте описать коптером круг с диаметром 1 м, по часовой и против часовой стрелки. Повторите траекторию в каждую сторону 2-3 раза.

Работа с рысканьем

При визуальном управлении мультикоптерными устройствами, рысканье не играет на столько важной роли, как с самолетной технике, поскольку коптер может передвигаться в любую сторону вне зависимости от того, куда он

направлен.

> Термин *рысканье* (*yaw*) обозначает поворот коптера вокруг вертикальной оси.

Основные задания:

1. Оборот коптера вокруг себя, ориентируя заднюю часть коптера к себе.
2. Оборот вокруг коптера, ориентируя заднюю часть к себе.

Для выполнения представленных упражнений рекомендуется найти большое свободное пространство.

Аналогично с предыдущими упражнениями перед взлетом выполните предполетные проверки.

Упражнение №1. Поднимите коптер на воздушную подушку и стабилизируйте его в одной точке. Описывайте коптером круг вокруг себя, на расстоянии 2-3 м, при этом поворачивая его таким образом, чтобы задняя часть коптера всегда была направлен на вас. Выполняйте упражнение по часовой стрелке и против. Повторите упражнение 4-5 раз.

Упражнение №2. Поднимите коптер на воздушную подушку и стабилизируйте его в одной точке. Обойдите коптер вокруг, при этом поворачивая его таким образом, чтобы задняя часть была направлена на вас. Обходите коптер по часовой стрелке и против. Повторите упражнение 4-5 раз.

> Дополнительные упражнения значительно сложнее обычных и не обязательны к выполнению. Приступайте к ним, только если вы уже уверенно управляете коптером.

Дополнительное упражнение №1. Поднимите коптер на воздушную подушку и стабилизируйте его в одной точке. Разверните коптер передней частью к себе и пробуйте управлять им задом наперед.

Дополнительное упражнение №2. Поднимите коптер на воздушную подушку и стабилизируйте его в одной точке. Выполняйте полет таким образом, чтобы передняя часть коптера всегда смотрела в сторону его движения.

Свободный полет

Если вы можете выполнить каждое из описанных выше упражнений, скорее всего, вы уже умеете свободно взлетать и управлять коптером. Далее будут представлены некоторые упражнения для закрепления полученных навыков.

Упражнения:

- Полет по вертикальному квадрату.
- Полет по граням куба.
- Полет по вертикальному кругу.

- Полет по восьмерке.
- Подъем коптера по спирали.

Закрепляйте полученные навыки необходимое для вас количество раз.

Raspberry Pi

Raspberry Pi – это свободно помещающийся на ладони одноплатный компьютер, созданный на базе мобильного микропроцессора ARM. Он обладает низким энергопотреблением и может работать даже от солнечных батарей. Raspberry Pi входит в комплекты программируемого квадрокоптера Клевер.



Технические характеристики:

- Вес – 45 грамм.
- Тактовая частота 1.2 ГГц.
- Графическое ядро в процессоре Broadcom BCM2837.
- оперативная память – 1 Гб.
- 4 порта USB 2.0.
- HDMI-порт.

В Клевере Raspberry Pi подключается к полетному контроллеру и используется как вспомогательный компьютер. Он позволяет подключаться к дрону по Wi-Fi, программировать автономные полеты, работать с периферией и многое другое.

Далее: образ для Raspberry Pi.

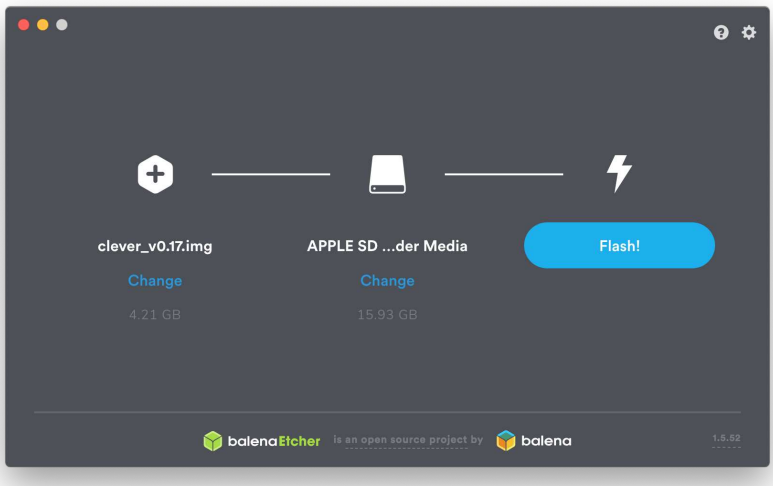
Образ для Raspberry Pi

Образ RPi для Клевера включает в себя все необходимое ПО для удобной работы с Клевером и программирования автономных полетов. Платформа Клевера основана на операционной системе Raspbian и популярном робототехническом фреймворке ROS. Исходный код сборщика образа и всех дополнительных пакетов доступен на GitHub.

Использование

> Начиная с версии v0.22, образ основан на ROS Noetic и использует Python 3. Если вы хотите использовать ROS Melodic и Python 2, используйте версию v0.21.2.

1. Скачайте последний стабильный релиз образа — **скачать**.
2. Скачайте и установите программу для записи образов Etcher (доступна для Windows/Linux/macOS).
3. Установите MicroSD-карту в компьютер (используйте адаптер при необходимости).
4. Запишите скачанный образ на карту, используя Etcher.
5. Установите карту в Raspberry Pi.



После записи образа на SD-карту, вы можете подключаться к Клеверу по Wi-Fi, использовать беспроводное соединение в QGroundControl, получать доступ по SSH и использовать остальные функции. При необходимости узнать версию записанного на карту образа используйте утилиту selfcheck.py.

Далее: Подключение по Wi-Fi.

Подключение к Клеверу по Wi-Fi

> Документация для версий образа, начиная с **0.20**. Для более ранних версий см. документацию для версии **0.19**.

На образе для RPi преднастроена раздача Wi-Fi с SSID `clover-xxxx`, где xxxx – 4 случайных цифры, назначаемых при первом включении Raspberry Pi.

Подключитесь к Wi-Fi, используя пароль `cloverwifi`.

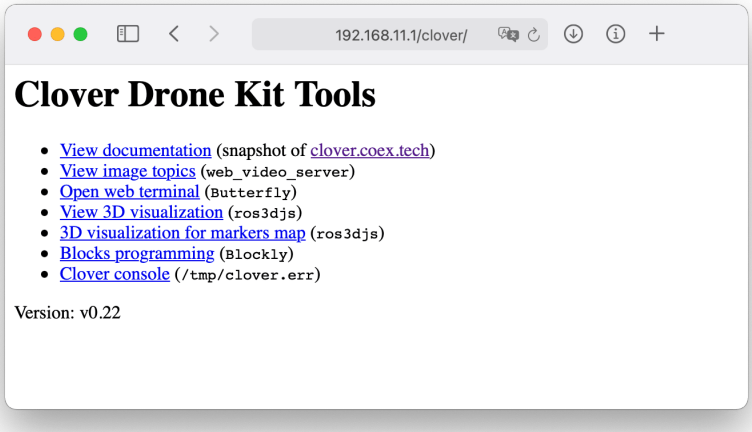




Для изменения настроек Wi-Fi или получения более детальной информации о устройстве сети на Raspberry Pi прочитайте статью "Настройка Wi-Fi".

Веб-интерфейс

После подключения к Клеверу по адресу http://192.168.11.1 будет доступен веб-интерфейс. В нем доступны основные веб-инструменты Клевера: просмотр топиков с изображениями, веб-терминал (Butterfly) а также полная копия данной документации.



Далее: Подключение Raspberry Pi к полетному контроллеру.

Подключение Raspberry Pi к полетному контроллеру

Для программирования автономных полетов, работы с Pixhawk (Pixracer) по Wi-Fi, использования телефонного пульта и других функций необходимо соединение Raspberry Pi и полетного контроллера.

Подключение по USB

Основным способом подключения является подключение по интерфейсу USB.



1. Соедините Raspberry Pi и полетный контроллер micro-USB to USB кабелем.
2. Подключитесь в Raspberry Pi по SSH.
3. Убедитесь в работоспособности подключения, выполнив команду на Raspberry Pi:

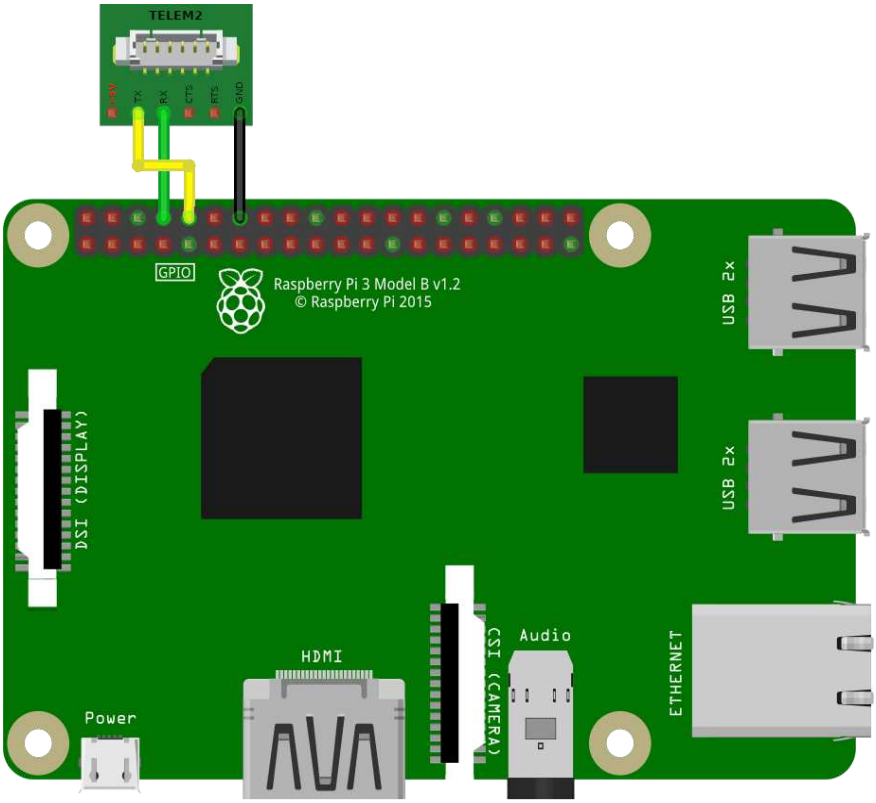
   ```
   rostopic echo /mavros/state
   ```

   Поле `connected` должно содержать значение `True` .

> Для корректной работы подключения Raspberry Pi и Pixhawk по USB необходимо установить значение параметра `CBRK_USB_CHK` на 197848.

Подключение по UART

Дополнительным способом подключения является подключение по интерфейсу UART.



Если обозначенный пин GND занят, можно использовать другой свободный, используя распиновку.

1. Подключите Raspberry Pi к полетному контроллеру по UART. Для этого соедините кабелем порт TELEM 2 на полетном контроллере к пинам на Raspberry Pi следующем образом: черный провод (GND) к Ground, зеленый (*UART_RX*) к *GPIO14*, желтый (*UART_TX*) к *GPIO15*. Красный провод (*5V*) подключать не нужно.
2. В PX4 версии v1.9.0 и выше измените значения параметров PX4: `MAV_1_CONFIG` на TELEM 2, `SER_TEL2_BAUND` на 921600 8N1. В PX4 до версии v1.9.0 необходима установка параметра `SYS_COMPANION` в значение `Companion Link (921600 baud, 8N1)`, для его корректной установки используйте старую версию QGC v3.3.1.
3. Подключитесь в Raspberry Pi по SSH.
4. Проверьте наличие параметров `enable_uart=1` и `dtoverlay=pi3-disable-bt` в файле `/boot/config.txt`, выполнив команду на Raspberry Pi:

   ```
   cat /boot/config.txt | grep -E "^enable_uart=.|^dtoverlay=pi3-disable-bt"
   ```

   Если параметры в файле отличаются или отсутствуют, то отредактируйте файл и перезагрузите Raspberry Pi.

5. Поменяйте в launch-файле Клевера (`~/catkin_ws/src/clover/clover/launch/clover.launch`) тип подключения с `usb` на `uart`:

   ```
   <arg name="fcu_conn" default="uart"/>
   ```

   При изменении launch-файла необходимо перезапустить сервис `clover`:

   ```
   sudo systemctl restart clover
   ```

6. Убедитесь в работоспособности подключения:

   ```
   rostopic echo -n1 /mavros/state
   ```

   Поле `connected` должно содержать значение `True`.

Дополнительная информация: https://docs.px4.io/main/en/peripherals/serial_configuration.html.

Далее: Подключение QGroundControl по Wi-Fi.

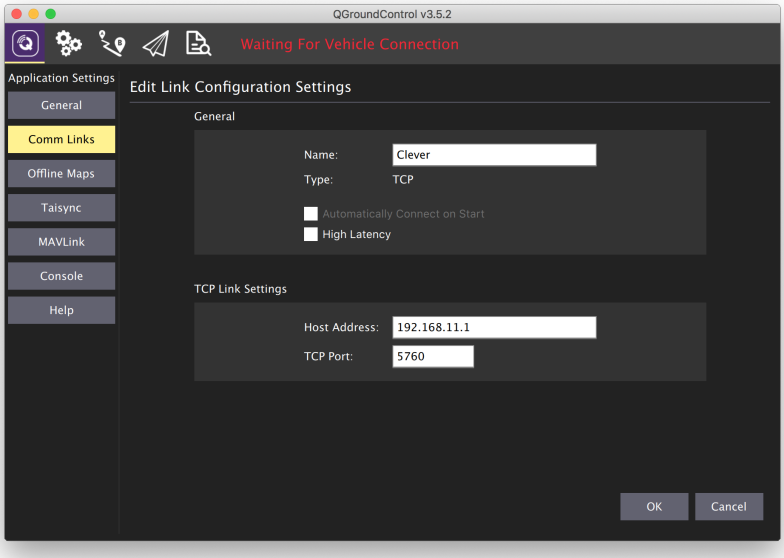
Подключение QGroundControl по Wi-Fi

Возможны контроль, управление, калибровка и настройка полетного контроллера квадрокоптера с помощью программы QGroundControl по Wi-Fi. Для этого необходимо подключиться к Wi-Fi сети `clover-xxxx`.

Подключение

По умолчанию на Клевере настроена возможность подключения QGroundControl по протоколу TCP.

1. На первой вкладке QGroundControl выберите меню *Comm Links*.
2. Нажмите кнопку *Add*, чтобы добавить новое подключение.
3. Введите параметры подключения:
   - Name: *Clover*.
   - Type: *TCP*.
   - Host Address: *192.168.11.1*.
   - TCP Port: *5760*.
4. Нажмите *OK* для сохранения параметров.
5. Выберите созданное подключение и нажмите *Connect*.



UDP

Также возможна настройка подключения по протоколу UDP. Для выбора различных вариантов подключения по UDP необходимо отредактировать параметр `gcs_bridge` в launch-файле `/home/pi/catkin_ws/src/clover/clover/launch/clover.launch`.

После изменения launch-файла необходимо перезагрузить сервис clover:

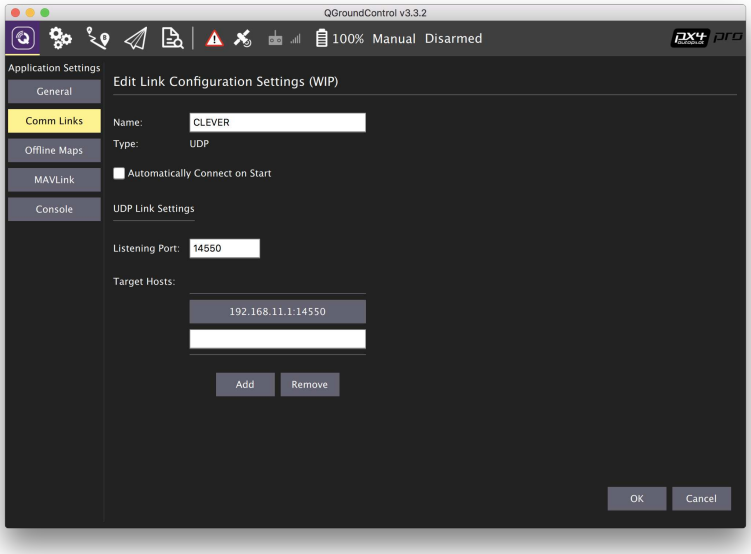
```
sudo systemctl restart clover
```

UDP-бридж с автоматическим подключением

1. Измените параметр `gcs_bridge` на `udp-b`.
2. При открытии программы QGroundControl соединение должно установиться автоматически.

UDP-бридж без автоматического подключения

1. Измените параметр `gcs_bridge` на `udp`.
2. В QGroundControl создайте подключение со следующими настройками:



3. Выберите созданное подключение и нажмите *Connect*.

UDP broadcast-бридж

> Особенностью UDP broadcast-бриджа является возможность просмотра телеметрии дрона одновременно с нескольких устройств (например с телефона и компьютера). Также он хорошо подходит для организации сети из устройств при помощи роутера.

1. Измените параметр `gcs_bridge` на `udp-pb`.
2. При открытии программы QGroundControl соединение должно установиться автоматически.

Далее: Доступ по SSH.

Доступ по SSH к Raspberry Pi

На образе для RPi преднастроен доступ по SSH для редактирования файлов, загрузки данных и запуска программ.

Для доступа по SSH необходимо подключиться к Raspberry Pi по Wi-Fi (также возможно подключение через Ethernet-кабель).

В GNU/Linux или macOS необходимо запустить Терминал и выполнить команду:

```
ssh pi@192.168.11.1
```

Пароль: `raspberry`.

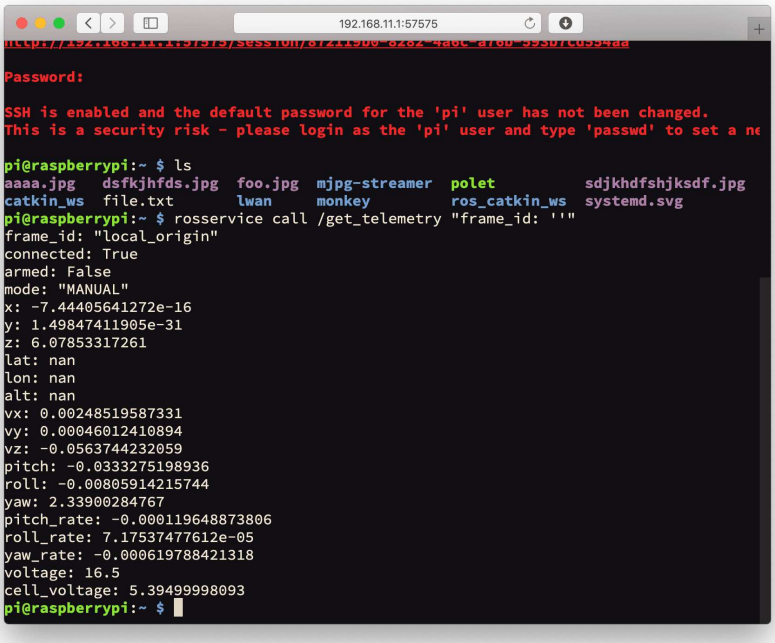
Для доступа по SSH из Windows можно использовать PuTTY или веб-доступ (см. далее). Также можно получить доступ по SSH со смартфона с помощью приложения Termius.

> Для того, чтобы не вводить пароль при каждом подключении по SSH, см. статью об использовании SSH-ключей.

Подробнее: https://www.raspberrypi.org/documentation/remote-access/ssh/README.md.

Веб-доступ

Начиная с версии 0.11.4 образа доступ к шеллу также доступен через веб-браузер (с использованием Butterfly). Для доступа откройте страницу http://192.168.11.1 и выберите на ней ссылку *Open web terminal*:



Далее: Командная строка.

Командная строка

В Linux-системах, к семейству которых принадлежит используемая на Raspberry Pi ОС Raspbian, основным способом взаимодействия пользователя с системой является командная строка. Для работы с командной строкой откройте SSH-соединение с Raspberry Pi.

Базовые команды

> Двойное нажатие клавиши `Tab ↹` позволяет автоматически дополнить вводимую команду или аргумент.

Показать содержимое текущей директории:

```
ls
```

Перейти в директорию:

```
cd catkin_ws/src/clover/clover/launch/
```

Перейти на директорию выше:

```
cd ..
```

Вывести путь к текущей директории:

```
pwd
```

Перейти в домашнюю директорию пользователя:

```
# все три команды равнозначны, где символ тильда (~) это сокращённая запись пут
# к домашней директории, а переменная $HOME хранит этот путь
cd
cd ~
cd $HOME
```

Вывести содержимое файла `file.py`:

```
cat file.py
```

Запустить Python-скрипт `file.py`:

```
python3 file.py
```

Перезагрузить Raspberry Pi:

```
sudo reboot
```

Для завершения работающей программы нажмите комбинацию клавиш `Ctrl` + `C`.

Читайте больше о командах Linux в документации Raspberry Pi: https://www.raspberrypi.org/documentation/linux/usage/commands.md.

Редактирование файлов

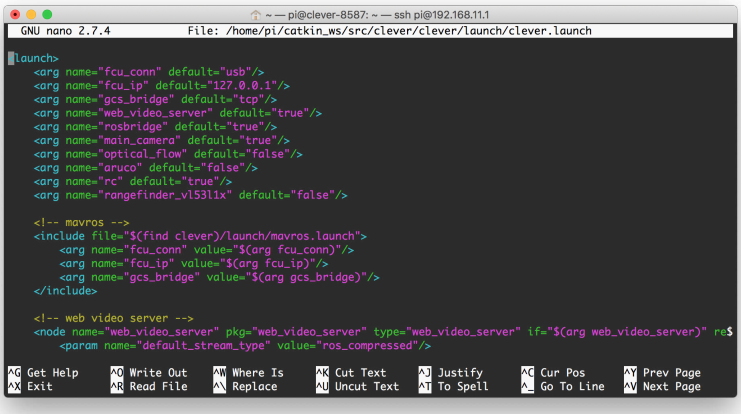
Используйте редактор **nano** для того, чтобы создавать или редактировать файлы на Raspberry Pi. Среди текстовых редакторов, доступных в терминале, он является наиболее простым и интуитивным.

1. Для редактирования файла введите команду:

   ```
   nano путь/к/файлу
   ```

   Например:

   ```
   nano ~/catkin_ws/src/clover/clover/launch/clover.launch
   ```



2. Отредактируйте файл.
3. Для выхода с сохранением нажмите `Ctrl` + `X`, `Y`, `Enter`.
4. При изменении .launch-файлов необходимо перезапустить пакет `clover`:

   ```
   sudo systemctl restart clover
   ```

Для редактирования файлов также можно использовать и другие редакторы, например, **vim**.

Сброс изменений

Для сброса изменений всех файлов, относящихся к пакету Клевера (`launch` -файлы) используйте git:

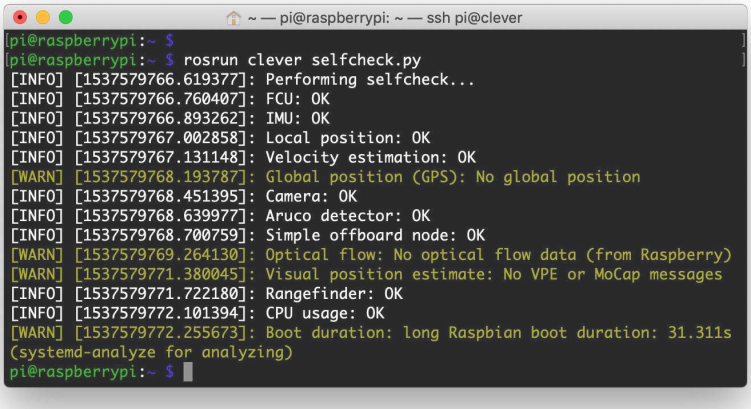
```
cd ~/catkin_ws/src/clover
git checkout .
sudo systemctl restart clover
```

Автоматическая проверка

Образ Клевера включает в себя средство автоматической проверки корректности настроек и работы всех подсистем квадрокоптера Клевер — **selfcheck.py**.

Для запуска наберите в консоли Raspberry Pi:

```
rosrun clover selfcheck.py
```



Описание некоторых проверок:

- FCU – проверка корректности соединения с полетным контроллером;
- IMU – проверка корректности данных с IMU;
- Local position – наличие локальной позиции дрона;
- Velocity estimation – оценка скоростей дрона (**запрещено выполнять автономный взлет при ошибках в этой проверке!**);
- Global position (GPS) – наличие глобальной позиции (требуется GPS);
- Camera – корректная работа камеры Raspberry.
- ArUco – проверка работы распознавания ArUco-маркеров.
- VPE – проверка правильности работы VPE.
- Rangefinder – проверка работы дальномера.
- RPi health – проверка состояния бортового компьютера.
- CPU usage – проверка загруженности процессора бортового компьютера.

> Обращайте пристальное внимание на предупреждения, отмеченные строкой *WARN*. При необходимости, обращайтесь в Техподдержку Copter Express.

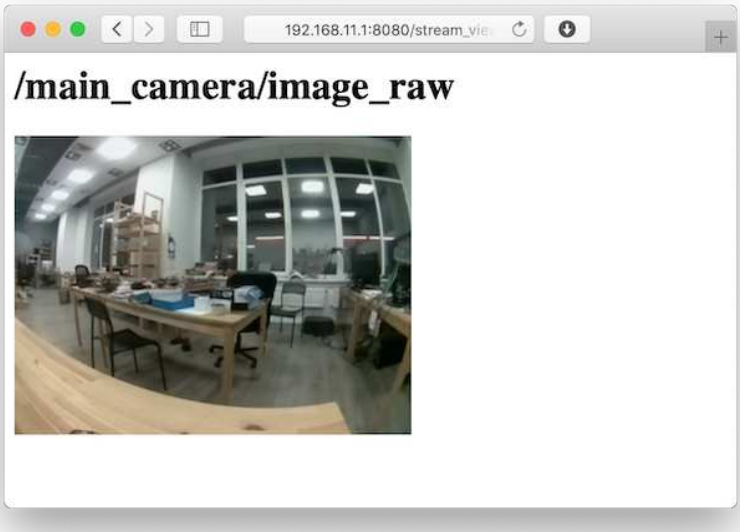
Просмотр изображений с камер

Для просмотра изображений с камер (или других ROS-топиков) можно воспользоваться rviz, rqt, или смотреть их через браузер, используя web_video_server.

См. подробнее про использование rqt.

Просмотр через браузер

Для просмотра видеострима нужно подключиться к Wi-Fi Клевера (`clover-xxxx`), перейти на страницу http://192.168.11.1:8080/ и выбрать топик.



Если передача картинки работает слишком медленно, можно ускорить ее, указав тип передаваемых данных `mjpeg` и меняя GET-параметр `quality` (от 1 до 100), который отвечает за сжатие видеострима, например:

http://192.168.11.1:8080/stream_viewer?topic=/main_camera/image_raw&type=mjpeg&quality=1

По URL выше будет доступен стрим с основной камеры в минимальном возможном качестве.

Также доступны параметры `width` , `height` и другие. Подробнее о `web_video_server` : http://wiki.ros.org/web_video_server.

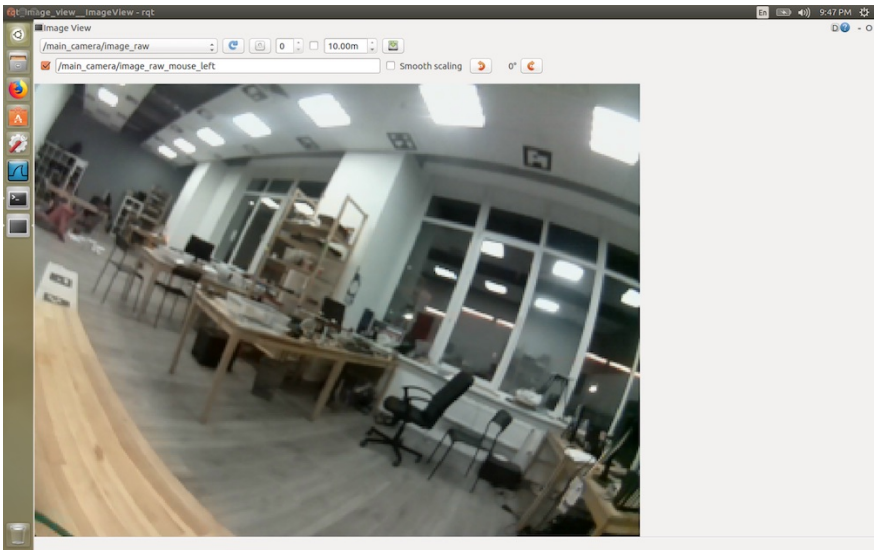
Просмотр через rqt_image_view

Для просмотра изображений через инструменты rqt необходим компьютер с установленной Ubuntu 20.04 и ROS Noetic.

Подключитесь к Wi-Fi сети Клевера и запустите `rqt_image_view` с указанием его IP-адреса:

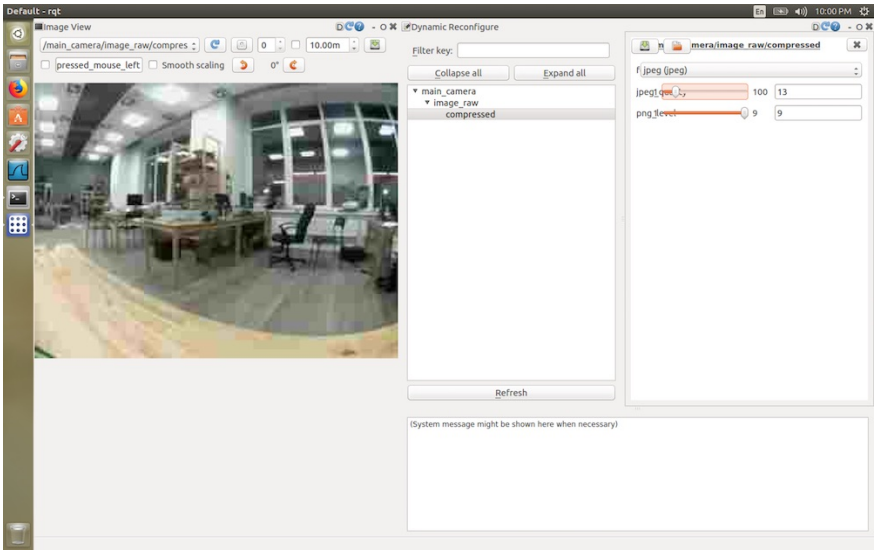
```
ROS_MASTER_URI=http://192.168.11.1:11311 rqt_image_view
```

Выберите топик для просмотра, например, `/main_camera/image_raw`:



Для снижения нагрузки на сеть и уменьшения задержки используйте сжатый вариант топика – `/main_camera/image_raw/compressed`.

Для изменения настроек сжатия используйте rqt-плагин Dynamic Reconfigure:



См. подробнее об rviz и rqt.

Программирование



Платформа Клевера позволяет использовать Raspberry Pi для того, чтобы запрограммировать автономный полет дрона. Чаще всего программа для автономного полета пишется на языке Python. Программа может получать телеметрию (заряд батареи, ориентацию, положение, скорости) и отправлять команды, например: полететь в точку, установить ориентацию, установить угловую скорость.

Платформа основывается на фреймворке ROS, который обеспечивает связь между пользовательской программой и сервисами Клевера, которые запущены в фоне в виде systemd-демона `clover`. Для связи с полетным контроллером используется пакет MAVROS.

Для автономного полета в PX4 используется режим OFFBOARD. API Клевера переводит дрон в этом режим автоматически. В случае необходимости прерывания автономного полета, необходимо перевести дрон в любой другой режим, используя стик переключения режимов на пульте.

Система позиционирования

Для того, чтобы дрон мог зависать на месте или летать между точками, необходимо использование системы позиционирования. Такая система вычисляет и сообщает дрону, где он находится. Клевер предполагает использование нескольких систем позиционирования: optical flow (используется камера и лазерный дальномер), визуальные маркеры (используется камера и маркеры, наклеенные на пол или потолок), GPS и других.

Optical flow

Принцип работы optical flow основан на вычислении сдвигов между соседними кадрами с камеры и передачи этой информации в полетный контроллер для дальнейшего расчета смещения дрона относительно изначальной точки.

Для настройки этой системы позиционирования обращайтесь к соответствующей статье.

ArUco-маркеры

Технология визуальных маркеров позволяет рассчитать позицию дрона относительно распознанных маркеров и передать эту информацию в полетный контроллер.

Читайте цикл статей про ArUco-маркеры для получения подробностей.

GPS (уличный полет)

Использование GPS позволяет также использовать для навигации глобальные координаты – широту и долготу (функция `navigate_global`).

Основная статья: подключение GPS.

Автономный полет

> Для изучения языка программирования Python можно обратиться к самоучителю.

После настройки системы позиционирования становится возможным написание скриптов для автономных полетов. Для выполнения скриптов подключитесь в Raspberry Pi по SSH.

Перед первым полетом рекомендуется проверить конфигурацию Клевера при помощи утилиты selfcheck.py:

```
rosrun clover selfcheck.py
```

Для того, чтобы запустить Python-скрипт, используйте команду `python3`:

```
python3 flight.py
```

Пример программы для полета (взлет, пролет вперед, посадка):

```
import rospy
from clover import srv
from std_srvs.srv import Trigger

rospy.init_node('flight')

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)

# Взлет на высоту 1 м
navigate(x=0, y=0, z=1, frame_id='body', auto_arm=True)

# Ожидание 3 секунды
rospy.sleep(3)

# Пролет вперед 1 метр
navigate(x=1, y=0, z=0, frame_id='body')

# Ожидание 3 секунды
rospy.sleep(3)

# Посадка
land()
```

> Функция navigate не ожидает, пока дрон долетит до целевой точки; скрипт продолжит выполнение сразу. Для блокирующей версии смотрите пример функции `navigate_wait`.

Обратите внимание, что параметр `auto_arm` установлен на `True` только у первого вызова функции `navigate`. Этот параметр армит дрон и переводит его в режим автономного полета (OFFBOARD).

Параметр `frame_id` задает систему координат, относительно которой задаются целевая точка для полета дрона:

- `body` связана с корпусом дрона;
- `navigate_target` связана с предыдущей целевой точкой полета;
- `map` связана с локальной системой координат дрона;
- `aruco_map` связана с картой ArUco-маркеров;
- `aruco_N` связана ArUco-маркером с ID=N.

Подробности описаны в статье "Системы координат".

Полное описания API Клевера приведено в статье "Автономный полет".

> В Клевере также доступна поддержка блочного программирования автономных полетов.

Дополнительное оборудование

Платформа Клевера также имеет API для работы с периферией. Читайте соответствующие статьи для подробностей:

- работа со светодиодной лентой;
- лазерный дальномер;
- GPIO;
- ультразвуковой дальномер;
- камера.

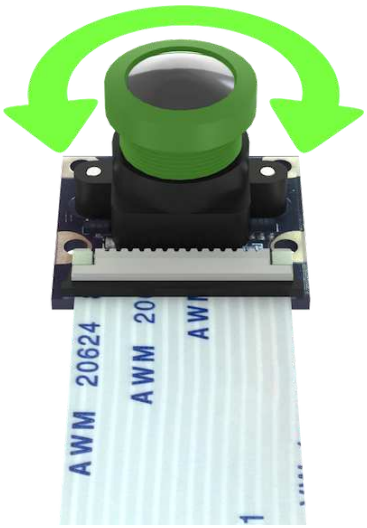
Настройка камеры

> Документация для версий образа, начиная с **0.20**. Для более ранних версий см. документацию для версии **0.19**.

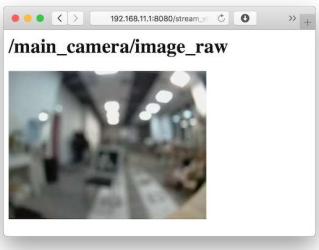
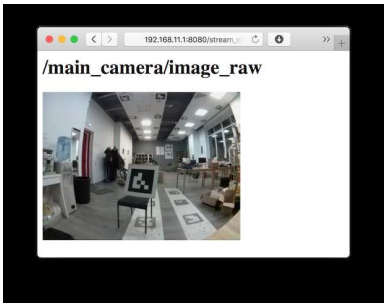
Для корректной работы всех функций, связанных с компьютерным зрением (в том числе полета по ArUco-маркерам и Optical Flow) необходимо сфокусировать основную камеру, а также выставить ее расположение и ориентацию. Улучшить качество работы также может опциональная калибровка камеры.

Настройка фокуса камеры

Для успешного осуществления полетов с использованием камеры, необходимо настроить фокус камеры.



1. Откройте трансляцию изображения с камеры используя web_video_server.
2. С помощью вращения объектива камеры добейтесь максимальной резкости деталей (предпочтительно на расстоянии предполагаемой высоты полета – 2–3 м).

Расфокусированное изображение	Сфокусированное изображение
	

Настройка расположения камеры

Расположение и ориентация камеры задается в файле `~/catkin_ws/src/clover/clover/launch/main_camera.launch`:

```
<arg name="direction_z" default="down"/> <!-- direction the camera points: down
<arg name="direction_y" default="backward"/> <!-- direction the camera cable po
```

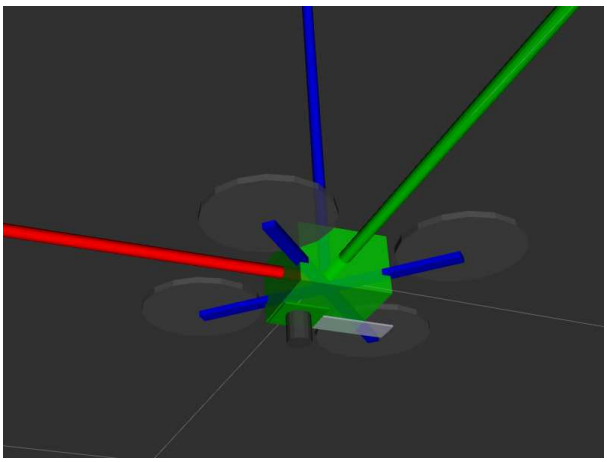
Для того, чтобы задать ориентацию, необходимо установить:

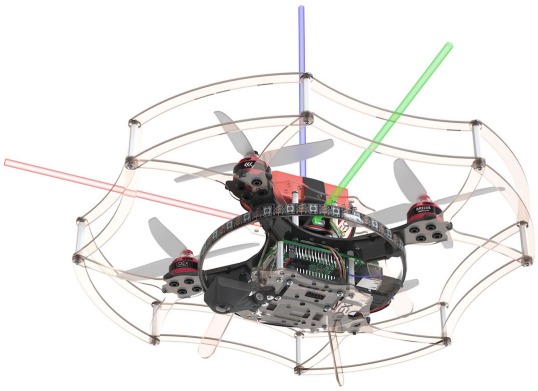
- направление обзора камеры `direction_z`: вниз (`down`) или вверх (`up`);
- направление, в которое указывает шлейф камеры `direction_y`: назад (`backward`) или вперед (`forward`).

Примеры

Камера направлена вниз, шлейф назад

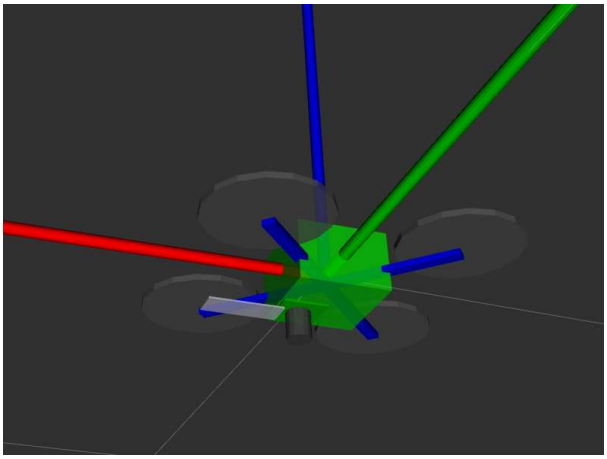
```
<arg name="direction_z" default="down"/>
<arg name="direction_y" default="backward"/>
```

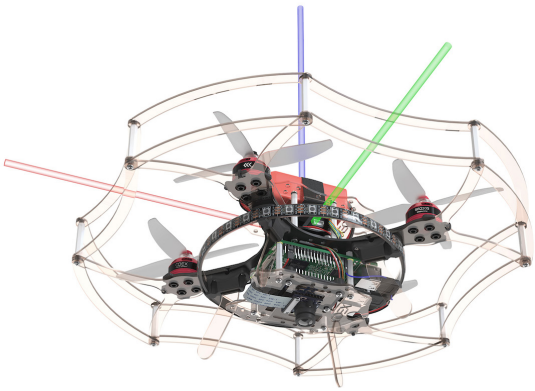




Камера направлена вниз, шлейф вперёд

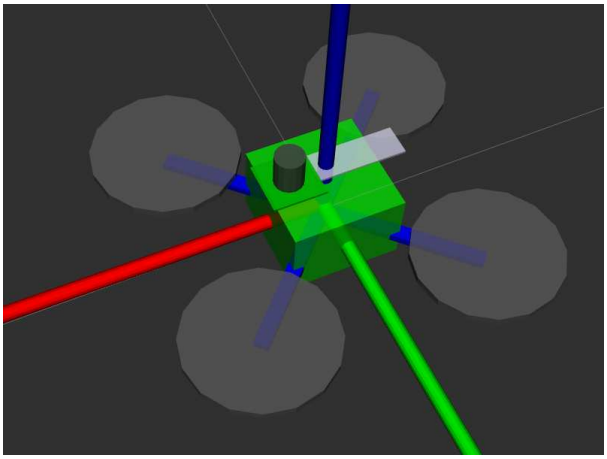
```
<arg name="direction_z" default="down"/>
<arg name="direction_y" default="forward"/>
```

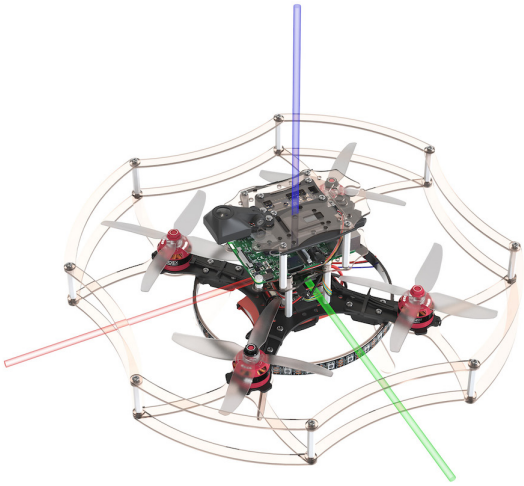




Камера направлена вверх, шлейф назад

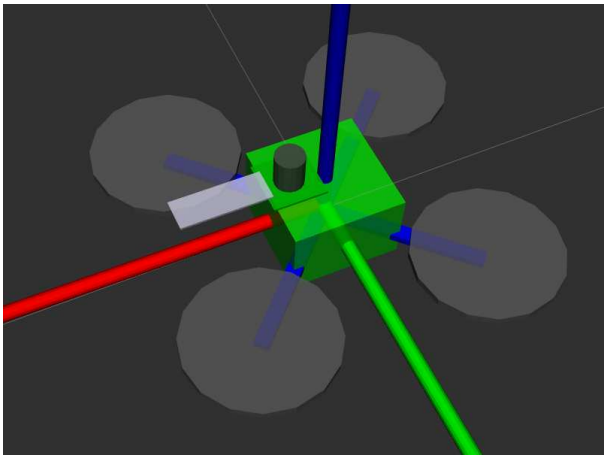
```
<arg name="direction_z" default="up"/>
<arg name="direction_y" default="backward"/>
```

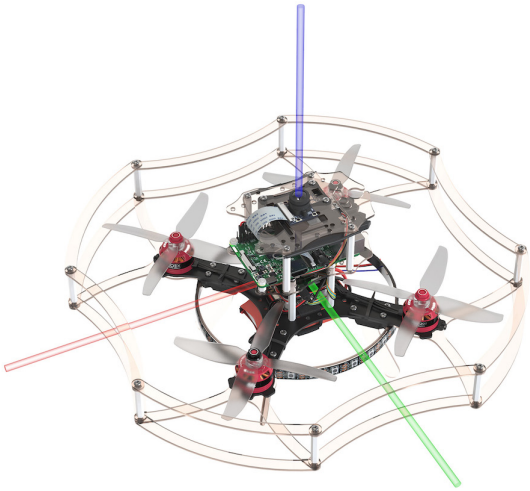




Камера направлена вверх, шлейф вперёд

```
<arg name="direction_z" default="up"/>
<arg name="direction_y" default="forward"/>
```





Утилита `selfcheck.py` выдает словесное описание установленной в данной момент ориентации основной камеры.

Произвольное расположение камеры

Также возможны произвольное расположение и ориентация камеры. Для этого раскомментируйте запуск ноды, подписанной как `Template for custom camera orientation`:

```
<!-- Template for custom camera orientation -->
<!-- Camera position and orientation are represented by base_link -> main_camer
<!-- static_transform_publisher arguments: x y z yaw pitch roll frame_id child_
<node pkg="tf2_ros" type="static_transform_publisher" name="main_camera_frame"
```

Эта строка задает статическую трансформацию между фреймом `base_link` (соответствует корпусу полетного контроллера) и камерой (`main_camera_optical`) в формате:

```
сдвиг_x сдвиг_y сдвиг_z угол_рысканье угол_тангаж угол_крен
```

Фрейм камеры задается таким образом, что:

- **x** указывает направо на изображении;

- **y** указывает вниз на изображении;
- **z** указывает от плоскости матрицы камеры.

Сдвиги задаются в метрах, углы задаются в радианах. Корректность установленной трансформации может быть проверена с использованием rviz.

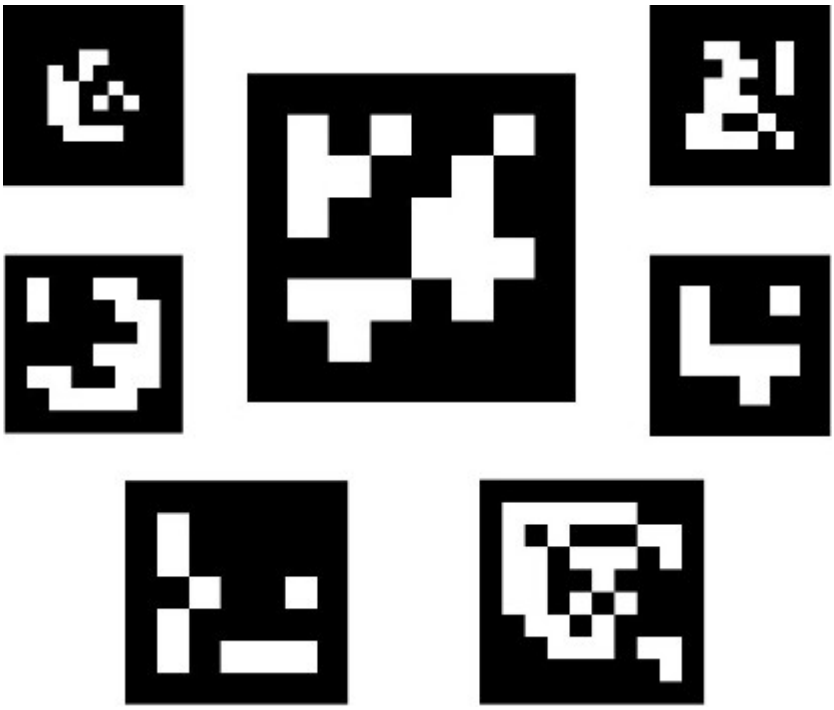
Калибровка

Для улучшения качества работы алгоритмов также рекомендуется произвести калибровку камеры, процесс которой описан в отдельной статье.

ArUco-маркеры

> Документация для версий образа, начиная с версии **0.16**. Для более ранних версий см. документацию для версии **0.15.1**.

ArUco-маркеры — это популярная технология для позиционирования робототехнических систем с использованием компьютерного зрения.



> При печати визуальных маркеров необходимо использовать максимально матовую бумагу. Глянцевая бумага будет бликовать на свету, сильно ухудшая качество распознавания.

Для быстрого генерирования маркеров для печати можно использовать онлайн-инструмент: http://chev.me/arucogen/.

На образе Клевера для RPi предустановлен пакет `aruco_pose`, предназначенный для работы с ArUco-маркерами.

Режимы работы

Клевер имеет несколько преднастроенных режимов работы с ArUco-маркерами:

- распознавание и навигация по отдельным маркерам;
- распознавание и навигация по картам маркеров.

> Исчерпывающую документацию по пакету `aruco_pose` на английском языке можно посмотреть на GitHub.

Распознавание ArUco-маркеров

> Документация для версий образа, начиная с версии **0.22**. Для более ранних версий см. документацию для версии **0.20**.

> Для распознавания маркеров модуль камеры должен быть корректно подключен и сконфигурирован.

Модуль `aruco_detect` распознает ArUco-маркеры и публикует их позиции в ROS-топики и в TF.

Эта функция полезна для применения совместно с какой-либо системой позиционирования для дрона, такой как GPS, Optical Flow, PX4Flow, визуальная одометрия, ультразвуковое (Marvelmind) или UWB-позиционирование (Pozyx).

Также возможно применение совместно с навигацией по карте маркеров.

Настройка

Аргумент `aruco` в файле `~/catkin_ws/src/clover/clover/launch/clover.launch` должен быть в значении `true`:

```
<arg name="aruco" default="true"/>
```

Для включения распознавания маркеров аргумент `aruco_detect` в файле `~/catkin_ws/src/clover/clover/launch/aruco.launch` должен быть в значении `true`:

```
<arg name="aruco_detect" default="true"/>
```

Для правильной работы в этом же файле также должны быть выставлены аргументы:

```
<arg name="placement" default="floor"/> <!-- расположение маркеров, см. далее
<arg name="length" default="0.33"/>     <!-- размер маркеров в метрах (не включ
```

Значение аргумента `placement` следует выставлять следующим образом:

- если *все* маркеры наклеены на полу (земле), выставить значение `floor`;
- если *все* маркеры наклеены на потолке, выставить значение `ceiling`;
- в противном случае удалить строку с параметром.

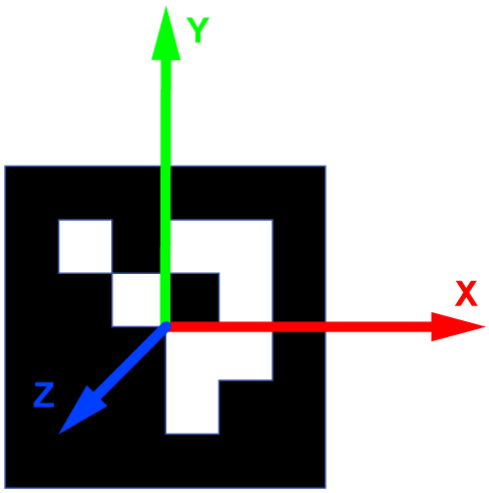
Если некоторые маркеры имеют размер, отличный значения `length`, их размер может быть переопределен с помощью параметра `length_override` ноды `aruco_detect`:

```
<param name="length_override/3" value="0.1"/>    <!-- маркер с id 3 имеет разме
<param name="length_override/17" value="0.25"/>  <!-- маркер с id 17 имеет разм
```

Система координат

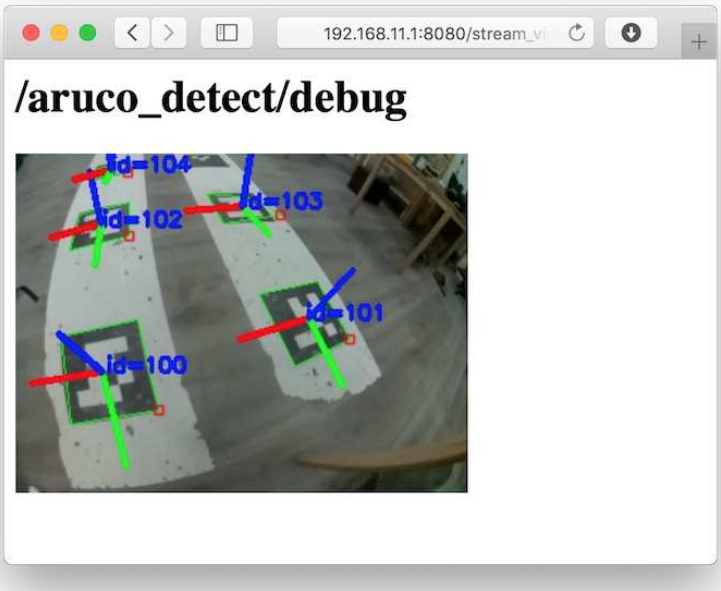
С маркером связана следующая система координат:

- ось **x** указывает на правую сторону маркера;
- ось **y** указывает кверху маркера;
- ось **z** указывает от плоскости маркера.



Работа с распознанными маркерами

Наглядно распознанные маркеры можно видеть в топике `aruco_detect/debug`. Просмотреть его можно с помощью `rqt_image_view` или через web_video_server по ссылке http://192.168.11.1:8080/snapshot?topic=/aruco_pose/debug:



Распознанные маркеры и их позиции публикуются в топик `aruco_detect/markers`. Чтение топика из Bash:

```
rostopic echo /aruco_detect/markers
```

Навигация по маркерам

С использованием модуля `simple_offboard` можно осуществлять навигацию по маркерам используя соответствующие TF-фреймы.

Полет в точку над маркером 5 на высоту 1 метр:

```
navigate(frame_id='aruco_5', x=0, y=0, z=1)
```

Полет в точку на метр левее маркера 7 на высоте 2 метра:

```
navigate(frame_id='aruco_7', x=-1, y=0, z=2)
```

Если необходимый маркер не появится в поле зрения в течение полусекунды, дрон продолжит выполнять предыдущую команду.

Подобные значения `frame_id` можно использовать и в других сервисах, например `get_telemetry`. Получение расположения дрона относительно маркера 3:

```
telem = get_telemetry(frame_id='aruco_3')
```

Если необходимый маркер не появится в поле зрения в течение полусекунды, в полях `telem.x`, `telem.y`, `telem.z`, `telem.yaw` будет значение `NaN`.

Работа с результатом распознавания из Python

Чтение топика `aruco_detect/markers` из Python:

```
import rospy
from aruco_pose.msg import MarkerArray
rospy.init_node('my_node')

# ...

def markers_callback(msg):
    print('Detected markers:'):
    for marker in msg.markers:
        print('Marker: %s' % marker)

# Подписываемся. При получении сообщения в топик aruco_detect/markers будет вы
rospy.Subscriber('aruco_detect/markers', MarkerArray, markers_callback)

# ...

rospy.spin()
```

Сообщения будут содержать ID маркера, его угловые точки на изображении и его позицию (относительно камеры).

См. далее: навигация по картам маркеров.

Навигация по картам ArUco-маркеров

> Документация для версий образа, начиная с версии **0.22**. Для более ранних версий см. документацию для версии **0.20**.

> Для распознавания маркеров модуль камеры должен быть корректно подключен и сконфигурирован.

> Рекомендуется использование специальной сборки PX4 для Клевера.

Модуль `aruco_map` распознает карты ArUco-маркеров, как единое целое. Также возможна навигация по картам ArUco-маркеров с использованием механизма Vision Position Estimate (VPE).

Конфигурирование

Аргумент `aruco` в файле `~/catkin_ws/src/clover/clover/launch/clover.launch` должен быть в значении `true`:

```
<arg name="aruco" default="true"/>
```

Для включения распознавания карт маркеров аргументы `aruco_map` и `aruco_detect` в файле `~/catkin_ws/src/clover/clover/launch/aruco.launch` должны быть в значении `true`:

```
<arg name="aruco_detect" default="true"/>
<arg name="aruco_map" default="true"/>
```

Для включения передачи координат в полетный контроллер по механизму VPE, аргумента `aruco_vpe` должен быть в значении `true`:

```
<arg name="aruco_vpe" default="true"/>
```

Настройка карты маркеров

Карта загружается из текстового файла, каждая строка которого имеет следующий формат:

```
id_маркера размер_маркера x y z угол_z угол_y угол_x
```

Где `угол_N` – это угол поворота маркера вокруг оси N в радианах.

Файлы карт располагаются в каталоге `~/catkin_ws/src/clover/aruco_pose/map`. Название файла с картой задается в аргументе `map`:

```
<arg name="map" default="map.txt"/>
```

Смотрите примеры карт маркеров в `вышеуказанном каталоге`.

Файл карты может быть сгенерирован с помощью инструмента `genmap.py`:

```
rosrun aruco_pose genmap.py length x y dist_x dist_y first -o test_map.txt
```

Где `length` – размер маркера, `x` – количество маркеров по оси *x*, `y` - количество маркеров по оси *y*, `dist_x` – расстояние между центрами маркеров по оси *x*, `y` – расстояние между центрами маркеров по оси *y*, `first` – ID первого (левого нижнего) маркера, `test_map.txt` – название файла с картой. Дополнительный ключ `--bottom-left` позволяет нумеровать маркеры с левого нижнего угла.

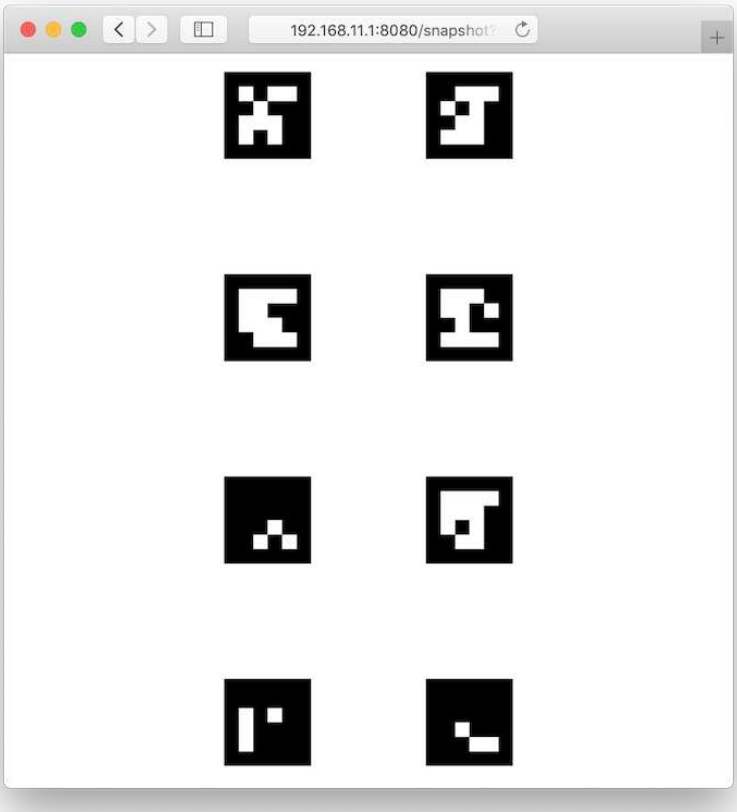
Пример:

```
rosrun aruco_pose genmap.py 0.33 2 4 1 1 0 -o test_map.txt
```

Дополнительную информацию по утилите можно получить по ключу `-h`: `rosrun aruco_pose genmap.py -h`.

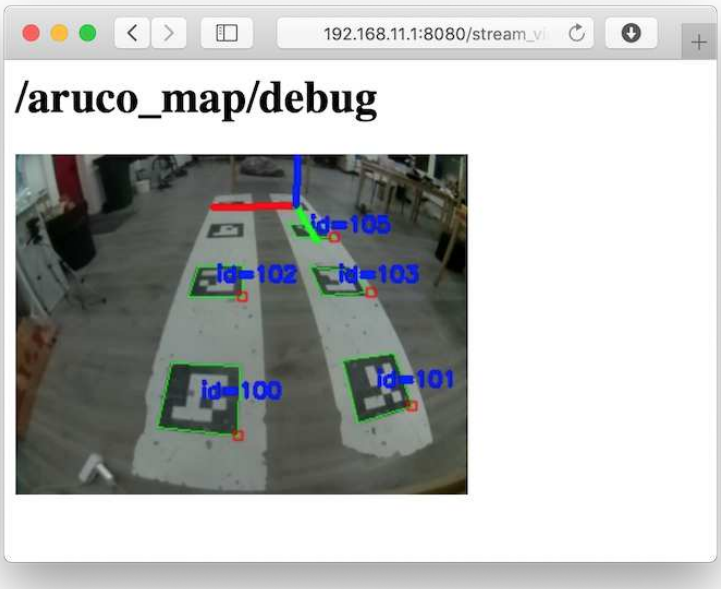
Проверка

Для контроля карты, по которой в данный момент коптер осуществляет навигацию, можно просмотреть содержимое топика `/aruco_map/image`. Через браузер его можно просмотреть при помощи web_video_server по ссылке http://192.168.11.1:8080/snapshot?topic=/aruco_map/image:



Клевер публикует текущую позицию распознанной карты в топик `aruco_map/pose`. Также публикуется TF-фрейм `aruco_map` (VPE выключен) или `aruco_map_detected` (VPE включен). Используя топик `aruco_map/visualization` можно визуализировать текущую карту маркеров в rviz.

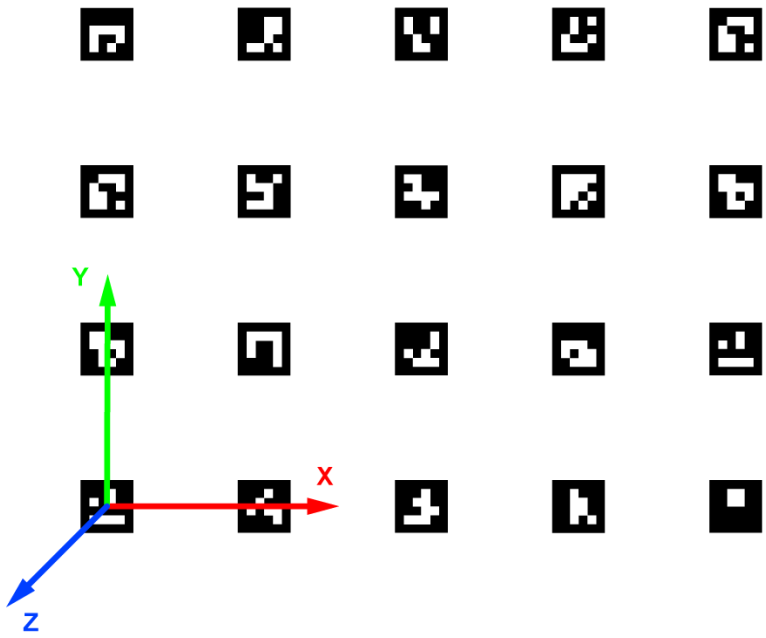
Наглядно позиция распознанной карты отображается в топике `aruco_map/debug` (просмотр доступен по ссылке http://192.168.11.1:8080/stream_viewer?topic=/aruco_map/debug):



Система координат

По соглашению в маркерном поле используется стандартная система координат ENU:

- ось **x** указывает на правую сторону карты маркеров;
- ось **y** указывает кверху карты маркеров;
- ось **z** указывает от плоскости карты маркеров.



Настройка VPE

Для работы механизма Vision Position Estimation необходимы следующие настройки PX4.

При использовании **LPE** (параметр `SYS_MC_EST_GROUP` = `local_position_estimator, attitude_estimator_q`):

- В параметре `LPE_FUSION` включены флажки `vision position`, `land detector`. Флажок `baro` рекомендуется отключить.
- Вес угла по рысканью по зрению: `ATT_W_EXT_HDG` = 0.5
- Включена ориентация по Yaw по зрению: `ATT_EXT_HDG_M` = 1 `Vision`.
- Шумы позиции по зрению: `LPE_VIS_XY` = 0.1 m, `LPE_VIS_Z` = 0.1 m.
- `LPE_VIS_DELAY` = 0 sec.

При использовании **EKF2** (параметр `SYS_MC_EST_GROUP` = `ekf2`):

- В параметре `EKF2_AID_MASK` включены флажки `vision position fusion`, `vision yaw fusion`.
- Шум угла по зрению: `EKF2_EVA_NOISE` = 0.1 rad.
- Шум позиции по зрению: `EKF2_EVP_NOISE` = 0.1 m.
- `EKF2_EV_DELAY` = 0.

> На данный момент для полета по маркерам рекомендуется использование **LPE**.

Для проверки правильности всех настроек можно воспользоваться утилитой `selfcheck.py`.

> Для использования LPE в Pixhawk необходимо скачать прошивку с названием `px4fmu-v2_lpe.px4`.

Полет

При правильной настройке коптер начнет удерживать позицию в режимах `POSCTL` и `OFFBOARD` автоматически.

Для автономных полетов можно будет использовать функции `navigate`, `set_position`, `set_velocity`. Для полета в определенные координаты маркерного поля необходимо использовать фрейм `aruco_map`:

```
# Вначале необходимо взлететь, чтобы коптер увидел карту меток и появился фрейм
navigate(x=0, y=0, z=2, frame_id='body', speed=0.5, auto_arm=True)  # взлет на

time.sleep(5)

# Полет в координату 2:2 маркерного поля, высота 2 метра
navigate(x=2, y=2, z=2, speed=1, frame_id='aruco_map')  # полет в координату 2
```

Полет в координаты по ID маркера

Начиная с версии образа 0.18, доступны также полёты относительно отдельного маркера в карте, даже если дрон его не видит. По аналогии с навигацией по отдельным маркерам при настройке карты маркеров дрон сможет лететь в координаты относительно отдельного маркера, используя фрейм `aruco_ID` с соответствующим ID маркера.

Полет в точку над маркером 5 на высоту 1 метр:

```
navigate(frame_id='aruco_5', x=0, y=0, z=1)
```

Дополнительные настройки

Дополнительную информацию по настройке PID см. в статье Настройка коэффициентов PID.

Если коптер нестабильно удерживает позицию по VPE, попробуйте увеличить коэффициенты *P* PID-регулятора по скорости – параметры `MPC_XY_VEL_P` и `MPC_Z_VEL_P`.

Если коптер нестабильно удерживает высоту, попробуйте увеличить коэффициент `MPC_Z_VEL_P` или лучше подобрать газ висения – `MPC_THR_HOVER`.

Расположение маркеров на потолке



Для навигации по маркерам, расположенным на потолке, необходимо поставить основную камеру так, чтобы она смотрела вверх и установить соответствующий фрейм камеры.

Также в файле `~/catkin_ws/src/clover/clover/launch/aruco.launch` необходимо выставить аргумент `placement` в значение `ceiling`:

```
<arg name="placement" default="ceiling"/>
```

Технология Optical Flow не может нормально работать при таком расположении камеры, поэтому в файле `~/catkin_ws/src/clover/clover/launch/clover.launch` ее следует отключить:

```
<arg name="optical_flow" default="false"/>
```

При такой конфигурации фрейм `aruco_map` также окажется перевернутым. Таким образом, для полета на высоту 2 метра ниже потолка, аргумент `z` нужно устанавливать в 2:

```
navigate(x=1, y=2, z=1.1, speed=0.5, frame_id='aruco_map')
```

Использование Optical Flow

При использовании технологии Optical Flow возможен полет в режиме POSCTL и автономные полеты (режим OFFBOARD) по камере, направленной вниз, за счет измерения сдвигов текстуры поверхности пола.

Включение

> Для работы Optical Flow необходим подключенный и настроенный лазерный дальномер.

Включите Optical Flow в файле `~/catkin_ws/src/clover/clover/launch/clover.launch`:

```
<arg name="optical_flow" default="true"/>
```

Optical Flow публикует данные в топик `/mavros/px4flow/raw/send`. Кроме того, в топик `/optical_flow/debug` публикуется визуализация, которую можно просмотреть с помощью web_video_server.

> Для правильной работы модуль камеры должен быть корректно подключен и сконфигурирован.

Настройка полетного контроллера

> При использовании сборки PX4 для Клевера необходимые параметры PX4 применятся автоматически.

При использовании **LPE** (параметр `SYS_MC_EST_GROUP` = `local_position_estimator, attitude_estimator_q`):

- `LPE_FUSION` – включены флажки fuse optical flow и flow gyro compensation.
- `LPE_FLW_QMIN` – 10.
- `LPE_FLW_SCALE` – 1.0.
- `LPE_FLW_R` – 0.2.
- `LPE_FLW_RR` – 0.0.
- `SENS_FLOW_ROT` – No rotation (отсутствие поворота).
- `SENS_FLOW_MAXHGT` – 4.0 (для дальномера VL53L1X)
- `SENS_FLOW_MINHGT` – 0.0 (для дальномера VL53L1X)
- Опционально: `LPE_FUSION` – включен флажок pub agl as lpos down (см. конфигурирование дальномера.

При использовании **EKF2** (параметр `SYS_MC_EST_GROUP` = `ekf2`):

- `EKF2_AID_MASK` – включен флажок use optical flow.

- `EKF2_OF_DELAY` – 0.
- `EKF2_OF_QMIN` – 10.
- `EKF2_OF_N_MIN` – 0.05.
- `EKF2_OF_N_MAX` - 0.2.
- `SENS_FLOW_ROT` – No rotation (отсутствие поворота).
- `SENS_FLOW_MAXHGT` – 4.0 (для дальномера VL53L1X)
- `SENS_FLOW_MINHGT` – 0.0 (для дальномера VL53L1X)
- Опционально: `EKF2_HGT_MODE` – range sensor (см. конфигурирование дальномера).

Для проверки правильности всех настроек можно воспользоваться утилитой `selfcheck.py`.

Полет в POSCTL

Настройте POSCTL как один из полетных режимов PX4. Переведите в режим POSCTL.

Автономный полет

Автономный полет возможен с использованием модуля simple_offboard.

Пример взлета на высоту 1.5 м и удержание позиции:

```
navigate(z=1.5, frame_id='body', auto_arm=True)
```

Полет вперед на 1 м:

```
navigate(x=1, frame_id='body')
```

Полет назад на 1 м (относительно предыдущей целевой точки):

```
navigate(x=-1, frame_id='navigate_target')
```

При использовании Optical Flow возможна также навигация по ArUco-маркерам, в том числе используя VPE.

Дополнительные настройки

Дополнительную информацию по настройке PID см. в статье Настройка коэффициентов PID.

Если коптер нестабильно удерживает позицию, попробуйте увеличить коэффициенты *P* PID-регулятора по скорости – параметры `MPC_XY_VEL_P` и `MPC_Z_VEL_P`.

Если коптер нестабильно удерживает высоту, попробуйте увеличить коэффициент `MPC_Z_VEL_P` или лучше подобрать газ висения – `MPC_THR_HOVER`.

Если коптер уплывает по рысканью, попробуйте:

- перекалибровать гироскопы;
- перекалибровать магнитометр;
- попробовать разные значения параметра `EKF2_MAG_TYPE`, который определяет, каким образом данные с магнитометра используются в EKF2;
- изменять значения параметров `EKF2_MAG_NOISE`, `EKF2_GYR_NOISE`, `EKF2_GYR_B_NOISE`.

> Для более хороших результатов выполняйте перекалибровку гироскопов непосредственно перед взлетом, используя соответствующий сниппет.

Если коптер уплывает по высоте, попробуйте:

- повысить значение коэффициента `MPC_Z_VEL_P`;
- изменить значение параметра `MPC_THR_HOVER`;
- выставить `MPC_ALT_MODE` = 2 (Terrain following).

При использовании Optical Flow максимальная горизонтальная скорость дополнительно ограничивается. За это косвенно отвечает параметр `SENS_FLOW_MAXR` (максимальная достоверная "угловая скорость" оптического потока). При нормальном полёте горизонтальная скорость будет регулироваться так, чтобы показания Optical Flow не превышали 50% значения данного параметра.

Неисправности

При появлении в QGC ошибок типа `EKF INTERNAL CHECKS` попробуйте перезагрузить EKF2. Для этого наберите в MAVLink-консоли:

```
ekf2 stop
ekf2 start
```

Автономный полет

> Документация для версий образа, начиная с версии **0.24**. Для более ранних версий см. документацию для версии **0.23**.

Модуль `simple_offboard` пакета `clover` предназначен для упрощенного программирования автономного полета дрона (режим `OFFBOARD`). Он позволяет устанавливать желаемые полетные задачи и автоматически трансформирует систему координат.

`simple_offboard` является высокоуровневым способом взаимодействия с полетным контроллером. Для более низкоуровневой работы см. mavros.

Основные сервисы – `get_telemetry` (получение телеметрии), `navigate` (полет в заданную точку по прямой), `navigate_global` (полет в глобальную точку по прямой), `land` (переход в режим посадки).

Использование из языка Python

Для использования сервисов, необходимо создать объекты-прокси к ним. Используйте этот шаблон для вашей программы:

```
import rospy
from clover import srv
from std_srvs.srv import Trigger

rospy.init_node('flight')

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_altitude = rospy.ServiceProxy('set_altitude', srv.SetAltitude)
set_yaw = rospy.ServiceProxy('set_yaw', srv.SetYaw)
set_yaw_rate = rospy.ServiceProxy('set_yaw_rate', srv.SetYawRate)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)
```

Неиспользуемые функции-прокси можно удалить из кода.

Описание API

> Незаполненные числовые параметры устанавливаются в значение 0.

get_telemetry

Получить полную телеметрию коптера.

Параметры:

- `frame_id` – система координат для значений `x`, `y`, `z`, `vx`, `vy`, `vz`. Пример: `map`, `body`, `aruco_map`. Значение по умолчанию: `map`.

Формат ответа:

- `frame_id` – система координат;
- `connected` – есть ли подключение к FCU;
- `armed` – состояние `armed` винтов (винты включены, если true);
- `mode` – текущий полетный режим;
- `x, y, z` – локальная позиция коптера *(м)*;
- `lat, lon` – широта, долгота *(градусы)*, необходимо наличие GPS;
- `alt` – высота в глобальной системе координат (стандарт WGS-84, не AMSL!), необходимо наличие GPS;
- `vx, vy, vz` – скорость коптера *(м/с)*;
- `roll` – угол по крену *(радианы)*;
- `pitch` – угол по тангажу *(радианы)*;
- `yaw` – угол по рысканью *(радианы)*;
- `roll_rate` – угловая скорость по крену *(рад/с)*;
- `pitch_rate` – угловая скорость по тангажу *(рад/с)*;
- `yaw_rate` – угловая скорость по рысканью *(рад/с)*;
- `voltage` – общее напряжение аккумулятора *(В)*;
- `cell_voltage` – напряжение аккумулятора на ячейку *(В)*.

> Недоступные по каким-то причинам поля будут содержать в себе значения `NaN`.

Вывод координат `x`, `y` и `z` коптера в локальной системе координат:

```
telemetry = get_telemetry()
print(telemetry.x, telemetry.y, telemetry.z)
```

Вывод высоты коптера относительно карты ArUco-меток:

```
telemetry = get_telemetry(frame_id='aruco_map')
print(telemetry.z)
```

Проверка доступности глобальной позиции:

```
import math
if not math.isnan(get_telemetry().lat):
    print('Global position is available')
else:
    print('No global position')
```

Вывод текущей телеметрии (командная строка):

```
rosservice call /get_telemetry "{frame_id: ''}"
```

navigate

Прилететь в обозначенную точку по прямой.

Параметры:

- `x`, `y`, `z` – координаты *(м)*;
- `yaw` – угол по рысканью *(радианы)*;
- `speed` – скорость полета (скорость движения setpoint) *(м/с)*;
- `auto_arm` – перевести коптер в `OFFBOARD` и заармить автоматически (**коптер взлетит**);
- `frame_id` – система координат, в которой заданы `x`, `y`, `z` и `yaw` (по умолчанию: `map`).

> Для полета без изменения угла по рысканью достаточно установить `yaw` в `NaN` (значение угловой скорости по умолчанию – 0).

Взлет на высоту 1.5 м со скоростью взлета 0.5 м/с:

```
navigate(x=0, y=0, z=1.5, speed=0.5, frame_id='body', auto_arm=True)
```

Полет по прямой в точку 5:0 (высота 2) в локальной системе координат со скоростью 0.8 м/с (рысканье установится в 0):

```
navigate(x=5, y=0, z=3, speed=0.8)
```

Полет в точку 5:0 без изменения угла по рысканью:

```
navigate(x=5, y=0, z=3, speed=0.8, yaw=float('nan'))
```

Полет вправо относительно коптера на 3 м:

```
navigate(x=0, y=-3, z=0, speed=1, frame_id='body')
```

Полет влево на 2 м относительно последней целевой точки полета дрона:

```
navigate(x=0, y=2, z=0, speed=1, frame_id='navigate_target')
```

Повернуться на 90 градусов по часовой:

```
navigate(yaw=math.radians(-90), frame_id='body')
```

Полет в точку 3:2 (высота 2) в системе координат маркерного поля со скоростью 1 м/с:

```
navigate(x=3, y=2, z=2, speed=1, frame_id='aruco_map')
```

Взлет на высоту 2 м (командная строка):

```
rosservice call /navigate "{x: 0.0, y: 0.0, z: 2, yaw: 0.0, speed: 0.5, frame_
```

> При программировании миссии дрона в терминах "вперед-назад-влево-вправо" рекомендуется использовать систему координат `navigate_target` вместо `body`, чтобы не учитывать неточность прилета дрона в предыдущую целевую точку при вычислении следующей.

navigate_global

Полет по прямой в точку в глобальной системе координат (широта/долгота).

Параметры:

- `lat`, `lon` – широта и долгота *(градусы)*;
- `z` – высота *(м)*;
- `yaw` – угол по рысканью *(радианы)*;
- `speed` – скорость полета (скорость движения setpoint) *(м/с)*;
- `auto_arm` – перевести коптер в `OFFBOARD` и заармить автоматически (**коптер взлетит**);
- `frame_id` – система координат, в которой заданы `z` и `yaw` (по умолчанию: `map`).

> Для полета без изменения угла по рысканью достаточно установить `yaw` в `NaN`.

Полет в глобальную точку со скоростью 5 м/с, оставаясь на текущей высоте (`yaw` установится в 0, коптер сориентируется передом на восток):

```
navigate_global(lat=55.707033, lon=37.725010, z=0, speed=5, frame_id='body')
```

Полет в глобальную точку без изменения угла по рысканью:

```
navigate_global(lat=55.707033, lon=37.725010, z=0, speed=5, yaw=float('nan'), f
```

Полет в глобальную точку (командная строка):

```
rosservice call /navigate_global "{lat: 55.707033, lon: 37.725010, z: 0.0, yaw
```

set_altitude

Изменить целевую высоту полета. Сервис используется для независимой установки высоты (и системы координат для расчета высота) в режимах полета `navigate` и `set_position`.

Параметры:

- `z` – высота полета *(м)*;
- `frame_id` – система координат для расчета высоты полета.

Установить высоту полета в 2 м относительно пола:

```
set_altitude(z=2, frame_id='terrain')
```

Установить высоту полета в 1 м относительно маркерного поля:

```
set_altitude(z=1, frame_id='aruco_map')
```

set_yaw

Изменить целевой угол по рысканью (и систему координат для его расчета), оставив предыдущую команду в силе.

Параметры:

- yaw – угол по рысканью *(радианы)*;
- frame_id – система координат для расчета угла по рысканью.

Повернуться на 90 градусов по часовой (продолжая выполнять предыдущую команду):

```
set_yaw(yaw=math.radians(-90), frame_id='body')
```

Установить угол по рысканью в ноль в системе координат маркерного поля:

```
set_yaw(yaw=0, frame_id='aruco_map')
```

Остановить вращение по рысканью (при использовании `set_yaw_rate`):

```
set_yaw(yaw=float('nan'))
```

set_yaw_rate

Изменить целевую угловую скорость по рысканью, оставив предыдущую команду в силе.

Параметры:

- yaw_rate – угловая скорость по рысканью *(рад/с)*.

Положительное направление вращения (при виде сверху) – против часовой.

Начать вращение на месте со скоростью 0.5 рад/с против часовой (продолжая выполнять предыдущую команду):

```
set_yaw_rate(yaw_rate=0.5)
```

set_position

Установить цель по позиции и рысканью. Данный сервис следует использовать при необходимости задания продолжающегося потока целевых точек, например, для полета по сложным траекториям (круговой, дугообразной и т. д.).

> Для полета на точку по прямой или взлета используйте более высокоуровневый сервис `navigate`.

Параметры:

- `x`, `y`, `z` – координаты точки *(м)*;
- `yaw` – угол по рысканью *(радианы)*;
- `auto_arm` – перевести коптер в `OFFBOARD` и заармить автоматически (**коптер взлетит**);
- `frame_id` – система координат, в которой заданы `x`, `y`, `z` и `yaw` (по умолчанию: `map`).

Зависнуть на месте:

```
set_position(frame_id='body')
```

Назначить целевую точку на 3 м выше текущей позиции:

```
set_position(x=0, y=0, z=3, frame_id='body')
```

Назначить целевую точку на 1 м впереди текущей позиции:

```
set_position(x=1, y=0, z=0, frame_id='body')
```

set_velocity

Установить скорости и рысканье.

- `vx`, `vy`, `vz` – требуемая скорость полета *(м/с)*;
- `yaw` – угол по рысканью *(радианы)*;
- `auto_arm` – перевести коптер в `OFFBOARD` и заармить автоматически (**коптер взлетит**);
- `frame_id` – система координат, в которой заданы `vx`, `vy`, `vz` и `yaw` (по умолчанию: `map`).

> Параметр `frame_id` определяет только ориентацию результирующего вектора скорости, но не его длину.

Полет вперед (относительно коптера) со скоростью 1 м/с:

```
set_velocity(vx=1, vy=0.0, vz=0, frame_id='body')
```

set_attitude

Установить тангаж, крен, рысканье и уровень газа (примерный аналог управления в режиме `STABILIZED`). Данный сервис может быть использован для более низкоуровневого контроля поведения коптера либо для управления коптером при отсутствии источника достоверных данных о его позиции.

Параметры:

- `roll`, `pitch`, `yaw` – необходимый угол по тангажу, крену и рысканью *(радианы)*;
- `thrust` – уровень газа от 0 (нет газа, пропеллеры остановлены) до 1 (полный газ);
- `auto_arm` – перевести коптер в `OFFBOARD` и заармить автоматически (**коптер взлетит**);
- `frame_id` – система координат, в которой задан `yaw` (по умолчанию: `map`).

set_rates

Установить угловые скорости по тангажу, крену и рысканью и уровень газа (примерный аналог управления в режиме `ACRO`). Это самый низкий уровень управления коптером (исключая непосредственный контроль оборотов моторов). Данный сервис может быть использован для автоматического выполнения акробатических трюков (например, флипа).

Параметры:

- `roll_rate`, `pitch_rate`, `yaw_rate` – угловая скорость по тангажу, крену и рыканью *(рад/с)*;
- `thrust` – уровень газа от 0 (нет газа, пропеллеры остановлены) до 1 (полный газ).
- `auto_arm` – перевести коптер в `OFFBOARD` и заармить автоматически (**коптер взлетит**);

Положительное направление вращения `yaw_rate` (при виде сверху) – против часовой, `pitch_rate` – вперед, `roll_rate` – влево.

land

Перевести коптер в режим посадки (`AUTO.LAND` или аналогичный).

> Для автоматического отключения винтов после посадки параметр PX4 `COM_DISARM_LAND` должен быть установлен в значение > 0.

Посадка коптера:

```
res = land()

if res.success:
    print('Copter is landing')
```

Посадка коптера (командная строка):

```
rosservice call /land "{}"
```

> В более новых версиях PX4 коптер выйдет из режима LAND в ручной режим, если сильно перемещать стики.

release

В случае необходимости приостановки отправки setpoint-сообщений, используйте сервис `simple_offboard/release`:

```
release = rospy.ServiceProxy('simple_offboard/release', Trigger)

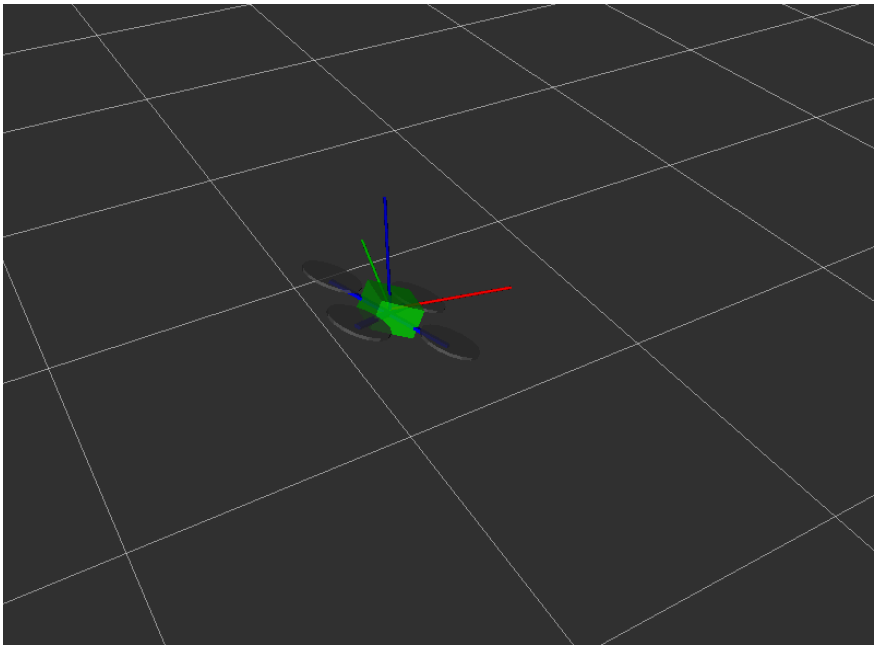
release()
```

Дополнительные материалы

- Полеты в поле ArUco-маркеров.
- Примеры программ и сниппеты.

Системы координат (фреймы)

> Документация для версий образа, начиная с **0.15**. Для более ранних версий см. документацию для версии **0.14**.



Основные фреймы в пакете `clover`:

- `map` — координаты относительно точки инициализации полетного контроллера: белая сетка на иллюстрации;
- `base_link` — координаты относительно квадрокоптера: схематичное изображение квадрокоптера на иллюстрации;
- `body` — координаты относительно квадрокоптера без учета наклонов по тангажу и крену: красная, синяя и зеленая линии на иллюстрации;
- `navigate_target` – координаты точки, в которую сейчас летит дрон (с использованием navigate);
- `terrain` – координаты относительно пола в текущей позиции коптера (см. сервис set_altitude)
- `setpoint` – текущий setpoint по позиции;
- `main_camera_optical` – система координат, связанная с основной камерой.

При использовании системы позиционирования по ArUco-маркерам появляются дополнительные фреймы:

- `aruco_map` – координаты относительно карты ArUco-маркеров;
- `aruco_N` – координаты относительно маркера с ID=N.

> В соответствии с соглашением, для фреймов, связанных с коптером, ось X направлена вперед, Y – налево и Z – вверх.

Более наглядно 3D визуализацию систем координат можно наблюдать, используя rviz.

tf2

Основная документация: http://wiki.ros.org/tf2

Для работы с системами координат в Клевере используется ROS-пакет tf2. tf2 – это набор библиотек для языков программирования C++, Python и других, которые помогают работать с системами координат. ROS-ноды публикуют в топик `/tf` сообщения формата `TransformStamped`, которые содержат в себе трансформации между заданными системами координат в определенные моменты времени.

С помощью `simple_offboard` можно запросить расположение коптера в любой системе координат, используя аргумент `frame_id` сервиса `get_telemetry`.

Из Python можно использовать библиотеку tf2 для преобразования геометрических объектов (например, PoseStamped, PointStamped) из одной системы координат в другую.

Примеры кода

Python

При использовании кириллических символов в кодировке UTF-8 необходимо добавить в начало программы указание кодировки:

```
# -*- coding: utf-8 -*-
```

#

Функция для полета в точку и ожидание окончания полета:

```
import math

def navigate_wait(x=0, y=0, z=0, yaw=float('nan'), speed=0.5, frame_id='', auto
    navigate(x=x, y=y, z=z, yaw=yaw, speed=speed, frame_id=frame_id, auto_arm=a

    while not rospy.is_shutdown():
        telem = get_telemetry(frame_id='navigate_target')
        if math.sqrt(telem.x ** 2 + telem.y ** 2 + telem.z ** 2) < tolerance:
            break
        rospy.sleep(0.2)
```

Для того, чтобы определить расстояние до целевой точки, функция использует фрейм `navigate_target`.

Использование функции для полета в точку x=3, y=2, z=1 относительно карты маркеров:

```
navigate_wait(x=3, y=2, z=1, frame_id='aruco_map')
```

Эту функцию можно использовать и для взлета:

```
navigate_wait(z=1, frame_id='body', auto_arm=True)
```

#

Посадка и ожидание окончания посадки:

```
def land_wait():
    land()
    while get_telemetry().armed:
        rospy.sleep(0.2)
```

Использование:

```
land_wait()
```

#

Ожидание окончания прилета в navigate-точку:

```
import math

def wait_arrival(tolerance=0.2):
    while not rospy.is_shutdown():
        telem = get_telemetry(frame_id='navigate_target')
        if math.sqrt(telem.x ** 2 + telem.y ** 2 + telem.z ** 2) < tolerance:
            break
        rospy.sleep(0.2)
```

#

Функция определения расстояния между двумя точками (**важно**: точки должны быть в одной системе координат):

```
import math

def get_distance(x1, y1, z1, x2, y2, z2):
    return math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2)
```

#

Функция для приблизительного определения расстояния (в метрах) между двумя глобальными координатами (широта/долгота):

```
import math

def get_distance_global(lat1, lon1, lat2, lon2):
    return math.hypot(lat1 - lat2, lon1 - lon2) * 1.113195e5
```

#

Дизарм коптера (выключение винтов, **коптер упадет**):

```
# Объявление прокси:
from mavros_msgs.srv import CommandBool
arming = rospy.ServiceProxy('mavros/cmd/arming', CommandBool)

# ...

arming(False)  # дизарм
```

#

Трансформировать позицию (`PoseStamped`) из одной системы координат (фрейма) в другую, используя tf2:

```
import tf2_ros
import tf2_geometry_msgs
from geometry_msgs.msg import PoseStamped

tf_buffer = tf2_ros.Buffer()
tf_listener = tf2_ros.TransformListener(tf_buffer)

# ...

# Создаем объект PoseStamped (либо получаем из топика):
pose = PoseStamped()
pose.header.frame_id = 'map'  # фрейм, в котором задана позиция
pose.header.stamp = rospy.get_rostime()  # момент времени, для которого задана
pose.pose.position.x = 1
pose.pose.position.y = 2
pose.pose.position.z = 3
pose.pose.orientation.w = 1

frame_id = 'base_link'  # целевой фрейм
transform_timeout = rospy.Duration(0.2)  # таймаут ожидания трансформации

# Преобразовываем позицию из старого фрейма в новый:
new_pose = tf_buffer.transform(pose, frame_id, transform_timeout)
```

#

Определение, перевернут ли коптер:

```
PI_2 = math.pi / 2
telem = get_telemetry()

flipped = abs(telem.roll) > PI_2 or abs(telem.pitch) > PI_2
```

#

Расчет общего угла коптера к горизонту:

```
PI_2 = math.pi / 2
telem = get_telemetry()

flipped = not -PI_2 <= telem.roll <= PI_2 or not -PI_2 <= telem.pitch <= PI_2
angle_to_horizon = math.atan(math.hypot(math.tan(telem.pitch), math.tan(telem.r
if flipped:
    angle_to_horizon = math.pi - angle_to_horizon
```

#

Полет по круговой траектории:

```
RADIUS = 0.6  # m
SPEED = 0.3  # rad / s

start = get_telemetry()
start_stamp = rospy.get_rostime()

r = rospy.Rate(10)

while not rospy.is_shutdown():
    angle = (rospy.get_rostime() - start_stamp).to_sec() * SPEED
    x = start.x + math.sin(angle) * RADIUS
    y = start.y + math.cos(angle) * RADIUS
    set_position(x=x, y=y, z=start.z)

    r.sleep()
```

#

Повторять действие с частотой 10 Гц:

```
r = rospy.Rate(10)
while not rospy.is_shutdown():
    # Do anything
    r.sleep()
```

#

Пример подписки на топики из MAVROS:

```
from geometry_msgs.msg import PoseStamped, TwistStamped
from sensor_msgs.msg import BatteryState
from mavros_msgs.msg import RCIn

def pose_update(pose):
    # Обработка новых данных о позиции коптера
    pass

rospy.Subscriber('mavros/local_position/pose', PoseStamped, pose_update)
rospy.Subscriber('mavros/local_position/velocity', TwistStamped, velocity_upda
rospy.Subscriber('mavros/battery', BatteryState, battery_update)
rospy.Subscriber('mavros/rc/in', RCIn, rc_callback)

rospy.spin()
```

Информацию по топикам MAVROS см. по ссылке.

#

Пример отправки произвольного MAVLink-сообщения коптеру:

```
from mavros_msgs.msg import Mavlink
from mavros import mavlink
from pymavlink import mavutil

mavlink_pub = rospy.Publisher('mavlink/to', Mavlink, queue_size=1)

# Отправка сообщения HEARTBEAT:

msg = mavutil.mavlink.MAVLink_heartbeat_message(mavutil.mavlink.MAV_TYPE_GCS, (
msg.pack(mavutil.mavlink.MAVLink('', 2, 1))
ros_msg = mavlink.convert_to_rosmsg(msg)

mavlink_pub.publish(ros_msg)
```

#

Подписка на все MAVLink-сообщения от полетного контроллера и их декодирование:

```
from mavros_msgs.msg import Mavlink
from mavros import mavlink
from pymavlink import mavutil

link = mavutil.mavlink.MAVLink('', 255, 1)

def mavlink_cb(msg):
    mav_msg = link.decode(mavlink.convert_to_bytes(msg))
    print('msgid =', msg.msgid, mav_msg) # print message id and parsed message

mavlink_sub = rospy.Subscriber('mavlink/from', Mavlink, mavlink_cb)

rospy.spin()
```

#

Реакция на переключение режима на пульте радиоуправления (может быть использовано для запуска автономного полета, см. пример):

```
from mavros_msgs.msg import RCIn

# Вызывается при получении новых данных с пульта
def rc_callback(data):
    # Произвольная реакция на переключение тумблера на пульте
    if data.channels[5] < 1100:
        # ...
        pass
    elif data.channels[5] > 1900:
        # ...
        pass
    else:
        # ...
        pass

# Создаем подписчик на топик с данными с пульта
rospy.Subscriber('mavros/rc/in', RCIn, rc_callback)

rospy.spin()
```

#

Сменить режим полета на произвольный:

```
from mavros_msgs.srv import SetMode

set_mode = rospy.ServiceProxy('mavros/set_mode', SetMode)

# ...

set_mode(custom_mode='STABILIZED')
```

#

Флип:

```
import math

PI_2 = math.pi / 2

def flip():
    start = get_telemetry()  # memorize starting position

    set_rates(thrust=1)  # bump up
    rospy.sleep(0.2)

    set_rates(pitch_rate=30, thrust=0.2)  # pitch flip
    # set_rates(roll_rate=30, thrust=0.2)  # roll flip

    while True:
        telem = get_telemetry()
        flipped = abs(telem.roll) > PI_2 or abs(telem.pitch) > PI_2
        if flipped:
            break

    rospy.loginfo('finish flip')
    set_position(x=start.x, y=start.y, z=start.z, yaw=start.yaw)  # finish fli

print(navigate(z=2, speed=1, frame_id='body', auto_arm=True))  # take off
rospy.sleep(10)

rospy.loginfo('flip')
flip()
```

Необходимо использование специальной сборки PX4 для Клевера. Перед выполнением флипа необходимо принять все меры безопасности.

#

Произвести калибровку гироскопа:

```
from pymavlink import mavutil
from mavros_msgs.srv import CommandLong
from mavros_msgs.msg import State

send_command = rospy.ServiceProxy('mavros/cmd/command', CommandLong)

def calibrate_gyro():
    rospy.loginfo('Calibrate gyro')
    if not send_command(command=mavutil.mavlink.MAV_CMD_PREFLIGHT_CALIBRATION,
        return False

    calibrating = False
    while not rospy.is_shutdown():
        state = rospy.wait_for_message('mavros/state', State)
        if state.system_status == mavutil.mavlink.MAV_STATE_CALIBRATING or stat
            calibrating = True
        elif calibrating and state.system_status == mavutil.mavlink.MAV_STATE_S
            rospy.loginfo('Calibrating finished')
            return True

calibrate_gyro()
```

> В процессе калибровки гироскопов дрон нельзя двигать.

#

Динамически включать и отключать распознавание ArUco-маркеров (например, для экономии ресурсов процессора):

```
import rospy
import dynamic_reconfigure.client

rospy.init_node('flight')
aruco_client = dynamic_reconfigure.client.Client('aruco_detect')

# Выключить распознавание маркеров
aruco_client.update_configuration({'enabled': False})

rospy.sleep(5)

# Включить распознавание маркеров
aruco_client.update_configuration({'enabled': True})
```

#

Динамически включать и отключать Optical Flow:

```
import rospy
import dynamic_reconfigure.client

rospy.init_node('flight')
flow_client = dynamic_reconfigure.client.Client('optical_flow')

# Выключить Optical Flow
flow_client.update_configuration({'enabled': False})

rospy.sleep(5)

# Включить Optical Flow
flow_client.update_configuration({'enabled': True})
```

#

> Для образа версии > 0.23.

Динамически изменить используемый файл с картой ArUco-маркеров:

```
import rospy
import dynamic_reconfigure.client

rospy.init_node('flight')
map_client = dynamic_reconfigure.client.Client('aruco_map')

map_client.update_configuration({'map': '/home/pi/catkin_ws/src/clover/aruco_p
```

#

Ожидать появления глобальной позиции (окончания инициализации GPS-приемника):

```
import math

while not rospy.is_shutdown():
    if math.isfinite(get_telemetry().lat):
        break
    rospy.sleep(0.2)
```

#

Считать параметр полетного контроллера:

```
from mavros_msgs.srv import ParamGet
from mavros_msgs.msg import ParamValue

param_get = rospy.ServiceProxy('mavros/param/get', ParamGet)

# Считать параметр типа INT
value = param_get(param_id='COM_FLTMODE1').value.integer

# Считать параметр типа FLOAT
value = param_get(param_id='MPC_Z_P').value.float
```

#

Изменить параметр полетного контроллера:

```
from mavros_msgs.srv import ParamSet
from mavros_msgs.msg import ParamValue

param_set = rospy.ServiceProxy('mavros/param/set', ParamSet)

# Изменить параметр типа INT:
param_set(param_id='COM_FLTMODE1', value=ParamValue(integer=8))

# Изменить параметр типа FLOAT:
param_set(param_id='MPC_Z_P', value=ParamValue(real=1.5))
```

#

Проверить, что код запущен в симуляции Gazebo:

```
is_simulation = rospy.get_param('/use_sim_time', False)
```

#

Переместить физический объект (линк) в Gazebo (а также поменять его скорости) можно при помощи сервиса `gazebo/set_link_state` (тип `SetLinkState`). Например, если добавить в мир объект куб (линк `unit_box::link`), то так можно переместить его в точку (1, 2, 3):

```
import rospy
from geometry_msgs.msg import Point, Pose, Quaternion
from gazebo_msgs.srv import SetLinkState
from gazebo_msgs.msg import LinkState

rospy.init_node('flight')

set_link_state = rospy.ServiceProxy('gazebo/set_link_state', SetLinkState)

# Переместить линк в Gazebo
set_link_state(LinkState(link_name='unit_box::link', pose=Pose(position=Point(1
```

> Простую анимацию объектов в Gazebo можно реализовать с помощью акторов.

Работа с лазерным дальномером

> Документация для версий образа, начиная с **0.18**. Для более ранних версий см. документацию для версии **0.17**.

Дальномер VL53L1X

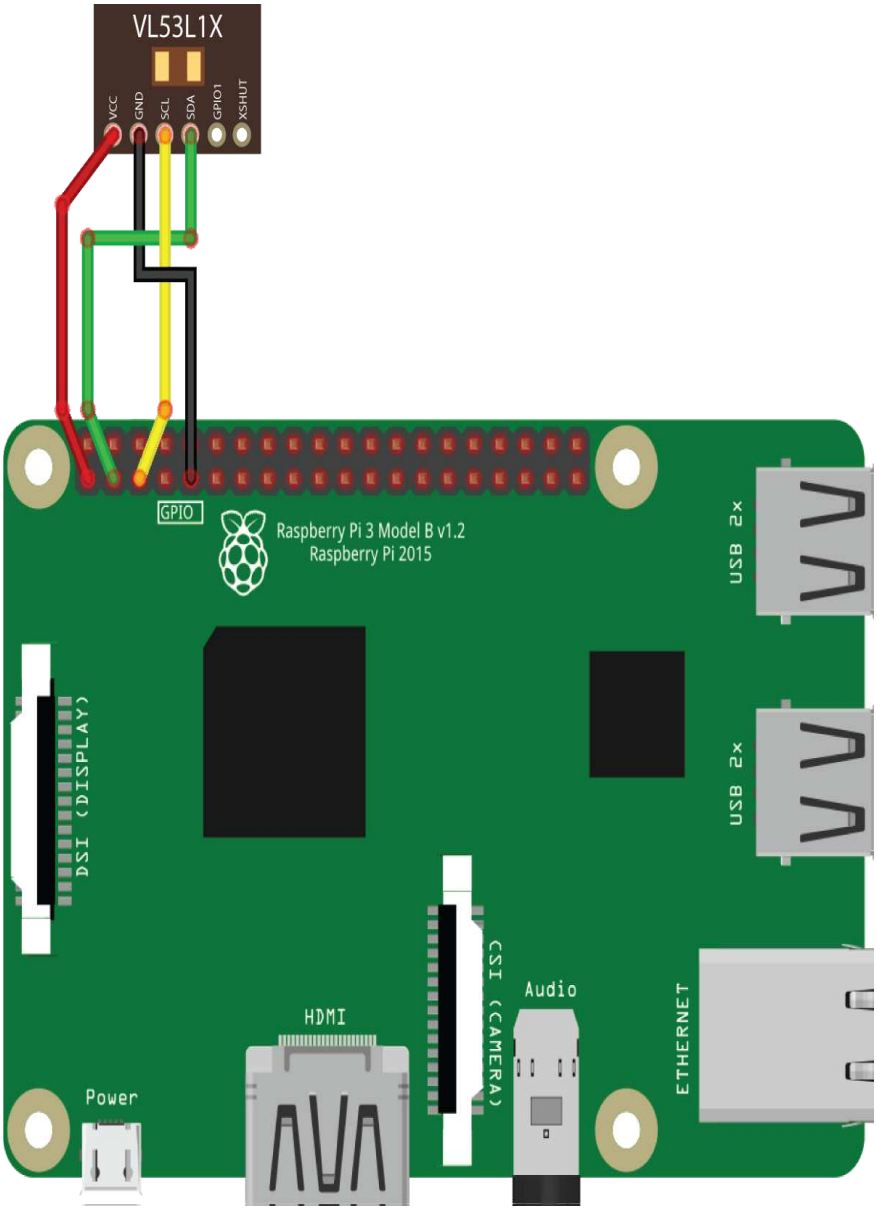
Рекомендуемая для Клевера модель дальномера – STM VL53L1X. Это дальномер может измерять расстояния от 0 до 4 м, при этом обеспечивая высокую точность измерений.

На образе для Raspberry Pi предустановлен соответствующий ROS-драйвер.

Подключение к Raspberry Pi

> Перед включением дальномера необходимо снять с него защитную пленку.

Подключите дальномер по интерфейсу I²C к пинам 3V, GND, SCL и SDA:



Если обозначенный пин GND занят, можно использовать другой свободный, используя распиновку.

> По интерфейсу I²C возможно подключать несколько периферийных устройств одновременно. Используйте для этого параллельное подключение.

Включение

Подключитесь по SSH и отредактируйте файл `~/catkin_ws/src/clover/clover/launch/clover.launch` так, чтобы драйвер VL53L1X был включен:

```
<arg name="rangefinder_vl53l1x" default="true"/>
```

По умолчания драйвер дальномера передает данные в Pixhawk (через топик `/rangefinder/range`). Для просмотра данных из топика используйте команду:

```
rostopic echo /rangefinder/range
```

Настройки PX4

> Для корректной работы лазерного дальномера с полетным контроллером рекомендуется использование специальной сборки PX4 для Клевера.

Для использования данных с дальномера в PX4 должен быть сконфигурирован.

При использовании EKF2 (`SYS_MC_EST_GROUP` = `ekf2`):

- `EKF2_HGT_MODE` = `2` (Range sensor) – при полете над горизонтальным полом;
- `EKF2_RNG_AID` = `1` (Range aid enabled) – в остальных случаях.

При использовании LPE (`SYS_MC_EST_GROUP` = `local_position_estimator, attitude_estimator_q`):

- В параметре `LPE_FUSION` включен флажок "pub agl as lpos down" – при полете над горизонтальным полом.

Получение данных из Python

Для получения данных из топика создайте подписчика:

```
import rospy
from sensor_msgs.msg import Range

rospy.init_node('flight')

def range_callback(msg):
    # Обработка новых данных с дальномера
    print('Rangefinder distance:', msg.range)

rospy.Subscriber('rangefinder/range', Range, range_callback)

rospy.spin() # дальнейший код программы
```

Также существует возможность однократного получения данных с дальномера:

```
from sensor_msgs.msg import Range

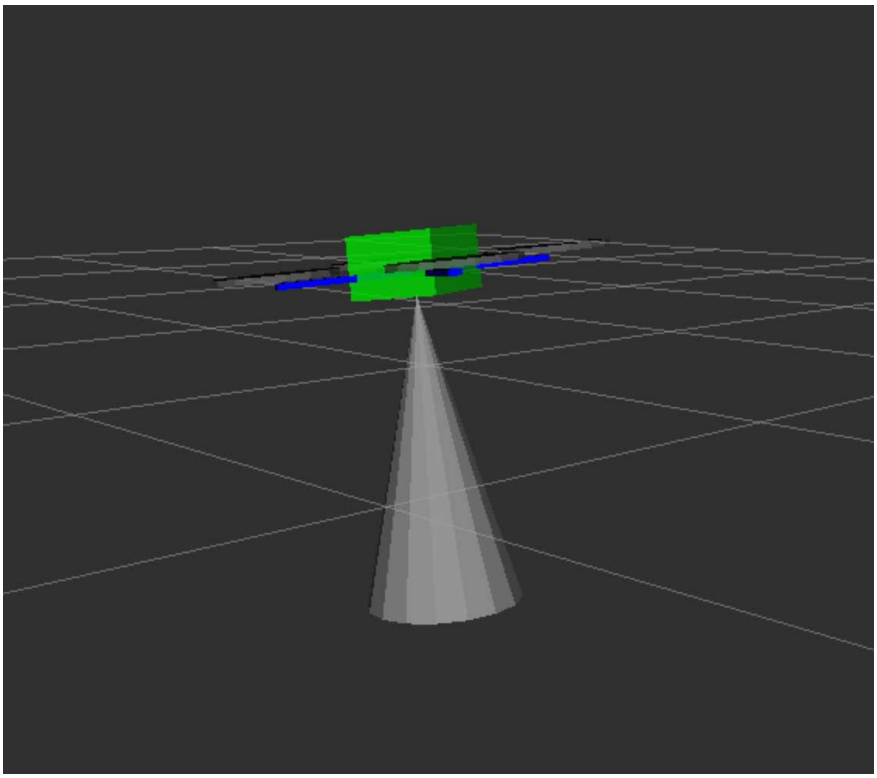
# ...

dist = rospy.wait_for_message('rangefinder/range', Range).range
```

Визуализация данных

Для построения графика по данным с дальномера может быть использован rqt_multiplot.

Для визуализации данных может быть использован rviz. Для этого необходимо добавить топик типа `sensor_msgs/Range` в визуализацию:



См. подробнее об rviz и rqt.

Работа со светодиодной лентой

> Документация для версий образа, начиная с **0.21**. Для более ранних версий см. документацию для версии **0.20**.

Адресуемая RGB-светодиодная лента типа *ws281x*, которая входит в наборы "Клевер", позволяет выставлять произвольные 24-битные цвета на каждый из отдельных светодиодов. Это позволяет сделать полет Клевера более ярким, а также визуально получать информацию о полетных режимах, этапе выполнения пользовательской программы и других событиях.



На образе для RPi предустановлены необходимые модули для работы с лентой. Они позволяют:

- управлять эффектами/анимациями на ленте;
- управлять лентой на низком уровне (переключением цветов отдельных светодиодов);
- настраивать реакцию ленты на полетные события.

> Обратите внимание, что светодиодную ленту нужно питать от стабильного источника энергии. Если вы подключите питание напрямую к Raspberry, то это создаст слишком большую нагрузку на ваш микрокомпьютер. Для снятия нагрузки с Raspberry можно подключить питание к преобразователю BEC.

Высокоуровневое управление лентой

1. Для работы с лентой подключите ее к питанию +5v – 5v, земле GND – GND и сигнальному порту DIN – GPIO21. Обратитесь к инструкции по сборке для подробностей.

2. Включите поддержку LED-ленты в файле `~/catkin_ws/src/clover/clover/launch/clover.launch`:

```
<arg name="led" default="true"/>
```

3. Настройте параметры подключения ленты *ws281x* в файле `~/catkin_ws/src/clover/clover/launch/led.launch`. Необходимо ввести верное количество светодиодов в ленте и GPIO-пин, использованный для подключения (если он отличается от *GPIO21*):

```
<arg name="led_count" default="58"/> <!-- количество светодиодов в ленте
<arg name="gpio_pin" default="21"/> <!-- GPIO-пин для подключения -->
```

Высокоуровневое управления лентой позволяет управлять текущим эффектом (анимацией) на ленте. Для этого используется ROS-сервис `/led/set_effect`. Параметры сервиса:

- `effect` – название необходимого эффекта.
- `r`, `g`, `b` – цвет эффекта в формате RGB. Значения изменяются от 0 до 255.

Список доступных эффектов:

- `fill` (или пустая строка) – залить всю ленту цветом;
- `blink` – мигание цветом;
- `blink_fast` – ускоренное мигание цветом;
- `fade` – плавное перетекание в цвет;
- `wipe` – "надвигание" нового цвета;
- `flash` – быстро мигнуть цветом 2 раза и вернуться к предыдущему эффекту;
- `rainbow` – переливание ленты цветами радуги;
- `rainbow_fill` – переливать заливку по цветам радуги.

Пример работы с сервисом из Python:

```
import rospy
from clover.srv import SetLEDEffect

rospy.init_node('flight')

set_effect = rospy.ServiceProxy('led/set_effect', SetLEDEffect)  # define proxy

set_effect(r=255, g=0, b=0)  # fill strip with red color
rospy.sleep(2)

set_effect(r=0, g=100, b=0)  # fill strip with green color
rospy.sleep(2)

set_effect(effect='fade', r=0, g=0, b=255)  # fade to blue color
rospy.sleep(2)

set_effect(effect='flash', r=255, g=0, b=0)  # flash twice with red color
rospy.sleep(5)

set_effect(effect='blink', r=255, g=255, b=255)  # blink with white color
rospy.sleep(5)

set_effect(effect='rainbow')  # show rainbow
```

Также лентой можно управлять из командной сроки (Bash):

```
rosservice call /led/set_effect "{effect: 'fade', r: 0, g: 0, b: 255}"
```

```
rosservice call /led/set_effect "{effect: 'rainbow'}"
```

Настройка реакции ленты на события

Клевер умеет показывать LED-лентой текущее состояние полетного контроллера и сигнализировать о событиях. Данная функция настраивается в файле `~/catkin_ws/src/clover/clover/launch/led.launch` в разделе *events effects table*. Пример настройки:

```
startup: { r: 255, g: 255, b: 255 }
connected: { effect: rainbow }
disconnected: { effect: blink, r: 255, g: 50, b: 50 }
<!-- ... -->
```

В левой части таблицы указывается событие, на которая лента должна среагировать. В правой части указывается эффект (анимация), который необходимо включить при возникновении события.

Список поддерживаемых событий:

Событие	Описание	Эффект по умолчанию
`startup`	Запуск всех систем Клевера	Белый
`connected`	Успешное подключение к полетному контроллеру	Эффект радуги
`disconnected`	Разрыв связи с полетным контроллером	Мигание красным
`armed`	Переход в состояние Armed	
`disarmed`	Переход в состояние Disarmed	
`acro`	Режим Acro	Оранжевый
`stabilized`	Режим Stabilized	Зеленый
`altctl`	Режим Altitude	Желтый
`posctl`	Режим Position	Синий
`offboard`	Режим Offboard	Фиолетовый
`rattitude`, `mission`, `rtl`, `land`	Переход в соответствующие режимы	
`error`	Возникновение ошибки в ROS-нодах или полетном контроллере (*ERROR*-сообщение в топике `/rosout`)	Мигнуть красным
`low_battery`	Низкий заряд батареи (порог настраивается в параметре `threshold`)	Быстрое мигание красным

> Для корректной работы сигнализации LED-лентой о низком заряде батареи необходимо корректная калибровка электропитания.

Для того, чтобы отключить реакцию светодиодной ленты на события, установите аргумент `led_notify` в файле `~/catkin_ws/src/clover/clover/launch/led.launch` в значение `false`:

```
<arg name="led_notify" default="false"/>
```

Низкоуровневое управление лентой

Для управления отдельными светодиодами используется ROS-сервис `/led/set_leds`. В параметрах задается массив номеров и RGB-цветов светодиодов, которые необходимо переключить.

Пример работы с сервисом из Python:

```
import rospy
from led_msgs.srv import SetLEDs
from led_msgs.msg import LEDStateArray, LEDState

rospy.init_node('flight')

set_leds = rospy.ServiceProxy('led/set_leds', SetLEDs)  # define proxy to ROS s

# switch LEDs number 0, 1 and 2 to red, green and blue color:
set_leds([LEDState(0, 255, 0, 0), LEDState(1, 0, 255, 0), LEDState(2, 0, 0, 255
```

Сервис можно использовать из командной строки:

```
rosservice call /led/set_leds "leds:
- index: 0
  r: 50
  g: 100
  b: 200"
```

При использовании ленты в ROS-топике `/led/state` публикуется текущие цвета светодиодов. Просмотр топика из командной строки:

```
rostopic echo /led/state
```

Используя этот же топик можно получить общее выставленное в настройках количество светодиодов:

```
led_count = len(rospy.wait_for_message('led/state', LEDStateArray, timeout=10)
```

Работа с GPIO

GPIO (General-Purpose Input/Output) – это тип пинов на Raspberry Pi, напряжение на которых можно программно подавать и измерять. Также на некоторых пинах реализован аппаратный ШИМ (PWM). Интерфейс GPIO может быть использован для управления различной периферией: светодиодами, электромагнитами, электромоторами, сервоприводами и т. д.

> Используйте распиновку, чтобы понять, какие из пинов на Raspberry Pi поддерживают GPIO и ШИМ.

> Для того, чтобы не создавалось конфликтов при использовании портов *GPIO* в образе закрыт доступ для портов 0, 1, 2, 3, 14, 15, на которые выведены интерфейсы подключения I2C и UART.

Для работы с GPIO на образе для RPi предустановлена библиотека `pigpio`. Чтобы взаимодействовать с этой библиотекой, запустите соответствующий демон:

```
sudo systemctl start pigpiod.service
```

Для включение автозапуска демона `pigpiod` используйте команду:

```
sudo systemctl enable pigpiod.service
```

> При одновременном использовании `pigpiod` и LED-ленты возможны конфликты. Для подключения ленты используйте пин GPIO21. На версиях образа ниже 0.17 измените в файле `/lib/systemd/system/pigpiod.service` строку запуска сервиса на `ExecStart=/usr/bin/pigpiod -l -t 0 -x 0x0FFF3FF0`.

Пример работы с библиотекой:

```
import time
import pigpio

# инициализируем подключение к pigpiod
pi = pigpio.pi()

# устанавливаем режим 11 пина на вывод
pi.set_mode(11, pigpio.OUTPUT)

# включаем сигнал на 11 пине
pi.write(11, 1)

time.sleep(2)

# отключаем сигнал на 11 пине
pi.write(11, 0)

# ...

# устанавливаем режим 12 пина на ввод
pi.set_mode(12, pigpio.INPUT)

# считываем состояние 12 пина
level = pi.read(12)
```

Для определения номера пина используйте распиновку Raspberry Pi.

Подключение сервоприводов

Большинство сервоприводов управляются с помощью ШИМ-сигнала, причем крайним положениям привода соответствуют сигналы шириной приблизительно 1000 и 2000 мкс. Значения для конкретного сервопривода могут быть определены экспериментально.

Подключите сигнальный провод сервопривода к одному из GPIO-пинов Raspberry. Для управления сервоприводом, подключенного к 13 пину, используйте такой код:

```
import time
import pigpio

pi = pigpio.pi()

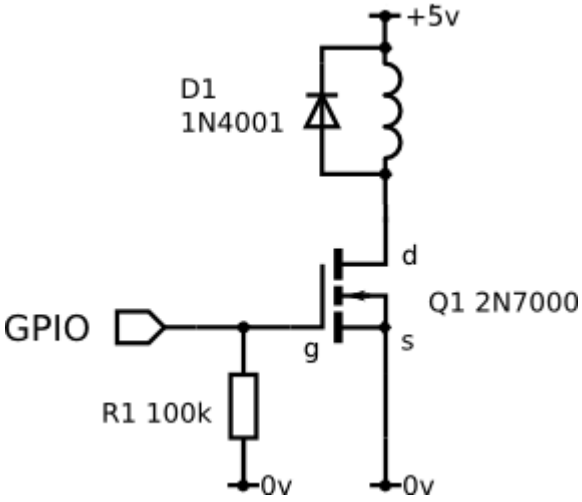
# устанавливаем режим 13 пина на вывод
pi.set_mode(13, pigpio.OUTPUT)

# устанавливаем на 13 пине ШИМ сигнал в 1000 мкс
pi.set_servo_pulsewidth(13, 1000)

time.sleep(2)

# устанавливаем на 13 пине ШИМ сигнал в 2000 мкс
pi.set_servo_pulsewidth(13, 2000)
```

Подключение электромагнита



Для подключения электромагнита используйте полевой транзистор (MOSFET). Подключите транзистор к одному из GPIO-пинов Raspberry Pi. Для управления магнитом, подключенным к 18 пину, используйте такой код:

```
import time
import pigpio

pi = pigpio.pi()

# устанавливаем режим 18 пина на вывод
pi.set_mode(18, pigpio.OUTPUT)

# включаем электромагнит
pi.write(18, 1)

time.sleep(2)

# отключаем электромагнит
pi.write(18, 0)
```

Работа с ультразвуковым дальномером

Ультразвуковой дальномер (*«сонар»*) — это датчик расстояния, принцип действия которого основан на измерении времени распространения звуковой волны (с частотой около 40 кГц) до препятствия и обратно. Сонар может измерять расстояние до 1,5–3 м с точностью до нескольких сантиметров.

Дальномер HC-SR04

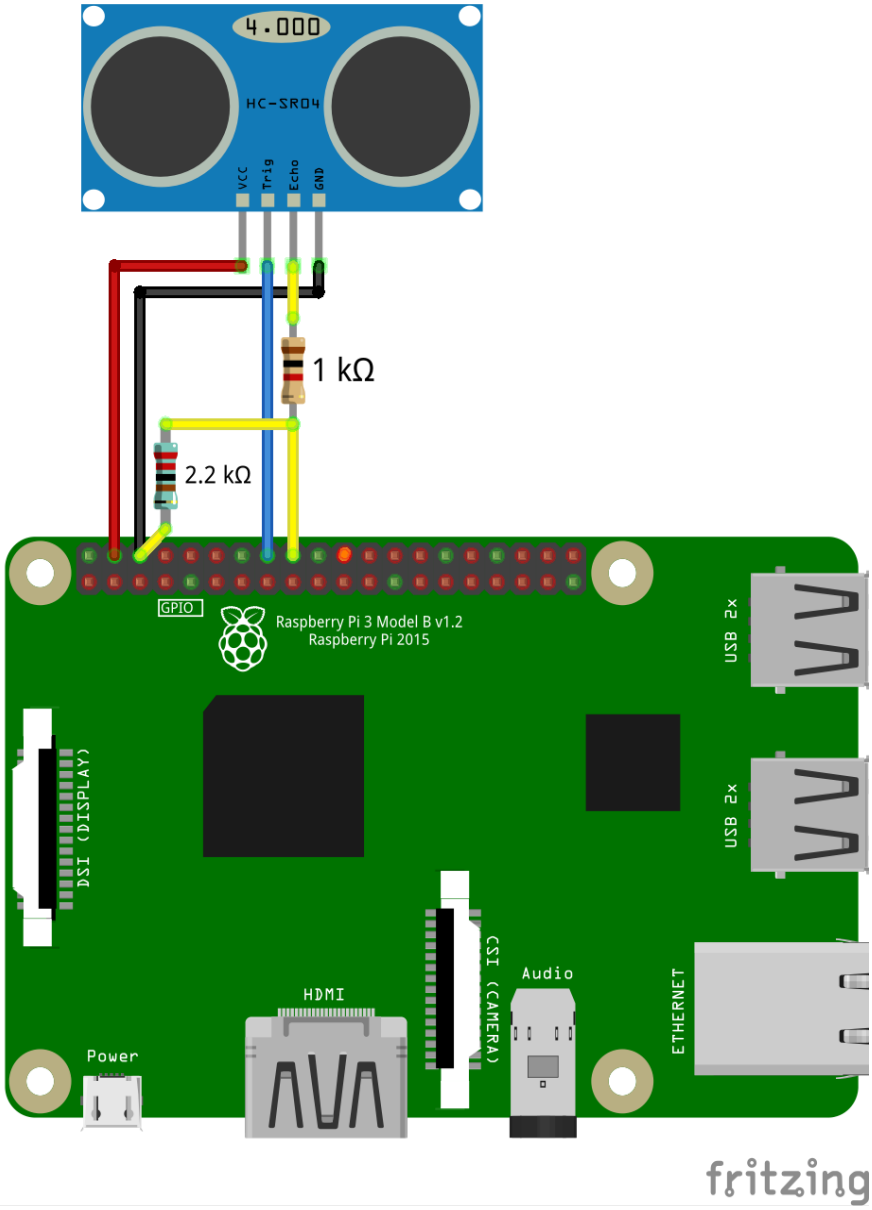


Установка

Дальномер закрепляется к корпусу с помощью двухстороннего скотча. Для получения приемлемых результатов необходимо использование виброразвязки. В качестве виброразвязки можно использовать кусок поролона.

Подключение

Подключите HC-SR04 к Raspberry Pi согласно схеме подключения. Используйте резисторы на 1,0 и 2,2 кОм и любые свободные GPIO-пины, например 23 и 24:



> Вместо резистора на 2,2 кОм можно использовать два резистора на 1 кОм, соединенные последовательно.

> На Raspberry Pi есть несколько взаимозаменяемых пинов **GND** и **VCC 5V**. Используйте распиновку, чтобы найти их.

Чтение данных

Чтобы считать данные с дальномера HC-SR04, используется библиотека для работы с GPIO – `pigpio`. Эта библиотека предустановлена на образе Клевера, начиная с версии **v0.14**. Для более старых версий образа используйте инструкцию по установке.

Для работы с `pigpio` необходимо запустить соответствующий демон:

```
sudo systemctl start pigpiod.service
```

Вы также можете включить автоматический запуск `pigpiod` при старте системы:

```
sudo systemctl enable pigpiod.service
```

Таким образом становится возможным взаимодействие с демоном `pigpiod` из языка Python:

```
import pigpio
pi = pigpio.pi()
```

> См. подробное описание Python API в документации `pigpio`.

Пример кода для чтения данных с HC-SR04:

```
import time
import threading
import pigpio

TRIG = 23  # пин, к которому подключен контакт Trig дальномера
ECHO = 24  # пин, к которому подключен контакт Echo дальномера

pi = pigpio.pi()
done = threading.Event()

def rise(gpio, level, tick):
    global high
    high = tick

def fall(gpio, level, tick):
    global low
    low = tick - high
    done.set()

def read_distance():
    global low
    done.clear()
    pi.gpio_trigger(TRIG, 50, 1)
    if done.wait(timeout=5):
        return low / 58.0 / 100.0

pi.set_mode(TRIG, pigpio.OUTPUT)
pi.set_mode(ECHO, pigpio.INPUT)
pi.callback(ECHO, pigpio.RISING_EDGE, rise)
pi.callback(ECHO, pigpio.FALLING_EDGE, fall)

while True:
    # Читаем дистанцию:
    print(read_distance())
```

Фильтрация данных

Для фильтрации (сглаживания) данных и удаления выбросов может быть использован фильтр Калмана или более простой медианный фильтр. Пример реализации медианной фильтрации:

```
import collections
import numpy

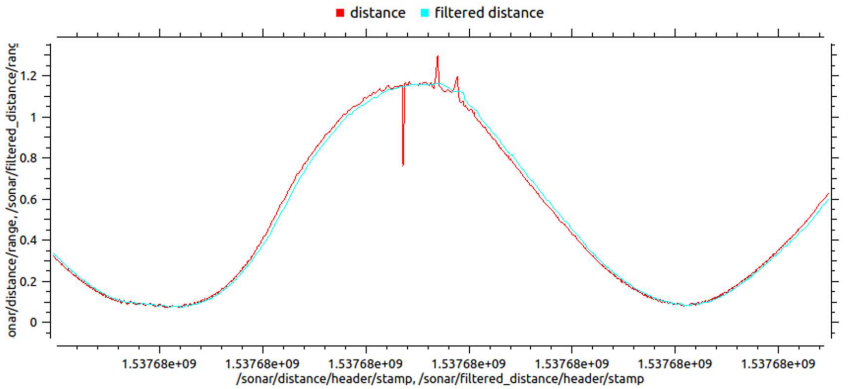
# ...

history = collections.deque(maxlen=10)  # 10 - количество сэмплов для усреднени

def read_distance_filtered():
    history.append(read_distance())
    return numpy.median(history)

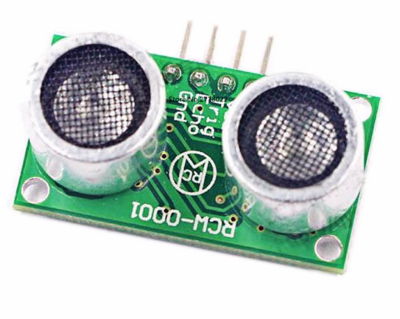
while True:
    print(read_distance_filtered())
```

Пример графиков исходных и отфильтрованных данных:



Исходный код ROS-ноды, использовавшейся для построения графика можно найти на Gist.

Дальномер RCW-0001



Ультразвуковой дальномер RCW-0001 совместим с дальномером HC-SR04. Используйте инструкцию выше для подключения и работы с ним.

Полет

Пример полетной программы с использованием simple_offboard, которая заставляет коптер лететь вперед, пока подключенный ультразвуковой дальномер не задетектирует препятствие:

```
set_velocity(vx=0.5, frame_id='body', auto_arm=True)  # полет вперед со скорост

while True:
    if read_distance_filtered() < 1:
        # если препятствие ближе, чем в 1 м, зависаем в точке
        set_position(x=0, y=0, z=0, frame_id='body')
    rospy.sleep(0.1)
```

Работа с камерой

> Эта статья описывает работу с образом версии **0.24**, который пока находится в стадии тестирования. Для версии **0.23** доступна более старая документация.

Для работы с основной камерой необходимо убедиться что она включена в файле `~/catkin_ws/src/clover/clover/launch/clover.launch`:

```
<arg name="main_camera" default="true"/>
```

Также нужно убедиться, что камера сфокусирована и для нее указано корректное расположение и ориентация.

При изменении launch-файла необходимо перезапустить пакет `clover`:

```
sudo systemctl restart clover
```

Для мониторинга изображения с камеры можно использовать rqt или web_video_server.

Неисправности

Если изображение с камеры отсутствует, попробуйте проверить ее с помощью утилиты `raspistill`.

Остановите сервисы Клевера:

```
sudo systemctl stop clover
```

Получите картинку с камеры утилитой `raspistill`:

```
raspistill -o test.jpg
```

Если команда завершается с ошибкой, проверьте качество подключения шлейфа камеры к Raspberry Pi или замените его.

Настройки камеры

Ряд параметров камеры - размер изображения, максимальную частоту кадров, экспозицию - можно настроить в файле `main_camera.launch`. Список настраиваемых параметров можно посмотреть в репозитории cv_camera.

Параметры, не указанные в этом списке, можно указывать через код параметра OpenCV. Например, для установки фиксированной экспозиции добавьте следующие параметры в ноду камеры:

```
<param name="property_0_code" value="21"/> <!-- property code 21 is CAP_PROP_AU
<param name="property_0_value" value="0.25"/> <!-- property values are normaliz
<param name="cv_cap_prop_exposure" value="0.3"/> <!-- set exposure to 30% of ma
```

Компьютерное зрение

Для реализации алгоритмов компьютерного зрения рекомендуется использовать предустановленную на образ SD-карты библиотеку OpenCV.

Python

Пример создания подписчика на топик с изображением с основной камеры для обработки с использованием OpenCV:

```
import rospy
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
from clover import long_callback

rospy.init_node('cv')
bridge = CvBridge()

@long_callback
def image_callback(data):
    img = bridge.imgmsg_to_cv2(data, 'bgr8')  # OpenCV image
    # Do any image processing with cv2...

image_sub = rospy.Subscriber('main_camera/image_raw', Image, image_callback)

rospy.spin()
```

> Обработка изображения может занимать значительное время. Это может вызвать проблему в библиотеке rospy, которая приведет к обработке устаревших кадров с камеры. Для решения этой проблемы необходимо использовать декоратор `long_callback` из библиотеки `clover`, как в примере выше.

Ограничение использования CPU

При использовании топика `main_camera/image_raw` скрипт будет обрабатывать максимальное количество кадров с камеры, активно используя CPU (вплоть до 100%). В задачах, где обработка каждого кадра не критична, можно использовать топик, где кадры публикуются с частотой 5 Гц: `main_camera/image_raw_throttled`:

```
image_sub = rospy.Subscriber('main_camera/image_raw_throttled', Image, image_ca
```

Публикация изображений

Для отладки обработки изображения можно публиковать отдельный топик с обработанным изображением:

```
image_pub = rospy.Publisher('~debug', Image)
```

Публикация обработанного изображения:

```
image_pub.publish(bridge.cv2_to_imgmsg(img, 'bgr8'))
```

Получаемые изображения можно просматривать используя web_video_server или rqt.

Получение одного кадра

Существует возможность единоразового получения кадра с камеры. Этот способ работает медленнее, чем подписка на топик; его не следует применять в случае необходимости постоянной обработки изображений.

```
import rospy
from sensor_msgs.msg import Image
from cv_bridge import CvBridge

rospy.init_node('cv')
bridge = CvBridge()

# ...

# Retrieve a frame:
img = bridge.imgmsg_to_cv2(rospy.wait_for_message('main_camera/image_raw', Ima
```

Примеры

Работа с QR-кодами

> Для высокоскоростного распознавания и позиционирования лучше использовать ArUco-маркеры.

Для программирования различных действий коптера при детектировании нужных QR-кодов можно использовать библиотеку pyZBar. Она уже установлена в последнем образе для Raspberry Pi.

Распознавание QR-кодов на Python:

```
import rospy
from pyzbar import pyzbar
import cv2
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
from clover import long_callback

rospy.init_node('cv')
bridge = CvBridge()

@long_callback
def image_callback(msg):
    img = bridge.imgmsg_to_cv2(msg, 'bgr8')
    barcodes = pyzbar.decode(img)
    for barcode in barcodes:
        b_data = barcode.data.decode('utf-8')
        b_type = barcode.type
        (x, y, w, h) = barcode.rect
        xc = x + w/2
        yc = y + h/2
        print('Found {} with data {} with center at x={}, y={}'.format(b_type,

image_sub = rospy.Subscriber('main_camera/image_raw_throttled', Image, image_ca

rospy.spin()
```

Смотрите другие примеры по работе с компьютерным зрением в каталоге `~/examples` образа для RPi.

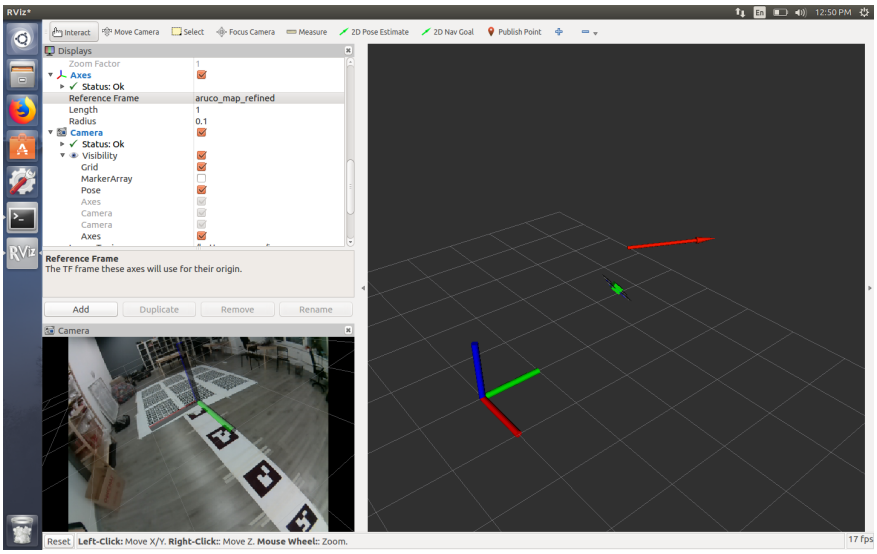
Запись видео

Для записи видео может использована нода `video_recorder` из пакета `image_view`:

```
rosrun image_view video_recorder image:=/main_camera/image_raw
```

Видео будет сохранено в файл `output.avi`. В аргументе `image` указывается название топика для записи видео.

Использование rviz и rqt



Инструмент rviz позволяет в реальном времени визуализировать на 3D-сцене все компоненты робототехнической системы — системы координат, движущиеся части, показания датчиков, изображения с камер.

rqt – это набор GUI для анализа и контроля ROS-систем. Например, `rqt_image_view` позволяет просматривать топики с изображениями, `rqt_multiplot` – строить графики по значениям в топиках и т. д.

Для использования rviz и rqt необходим компьютер с ОС Ubuntu Linux (либо виртуальная машина, например Parallels Desktop Lite или VirtualBox).

> Вы можете можете использовать готовый образ виртуальной машины с инструментами для Клевера.

На него необходимо установить пакет `ros-noetic-desktop-full` или `ros-noetic-desktop`, используя документацию по установке.

Запуск rviz

Для запуска визуализация состояния Клевера в реальном времени, необходимо подключиться к нему по Wi-Fi (`clover-xxxx`) и запустить rviz, указав соответствующий ROS_MASTER_URI:

```
ROS_MASTER_URI=http://192.168.11.1:11311 rviz
```

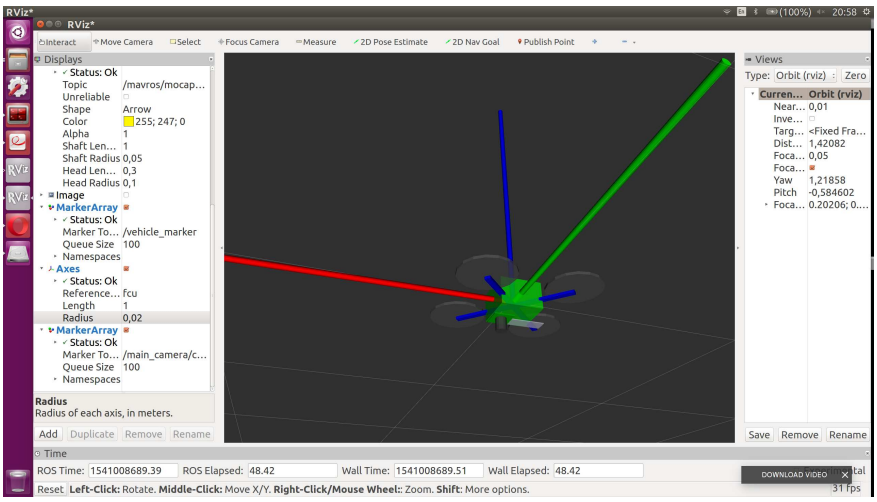
> В случае использования виртуальной машины для использования rviz и других инструментов может быть необходимо поменять ее сетевую конфигурацию на режим *bridge* (см. подробности для VMware).

Использование rviz

Визуализация положения коптера

В качестве reference frame рекомендуется установить фрейм `map`. Для визуализации коптера добавьте визуализационные маркеры из топика `/vehicle_markers`. Для визуализации камеры коптера добавьте визуализационные маркеры из топика `/main_camera/camera_markers`.

Результат визуализации коптера и камеры представлен ниже:



Визуализация окружения

Можно просмотреть картинку с дополненной реальностью из топика основной камеры `/main_camera/image_raw`.

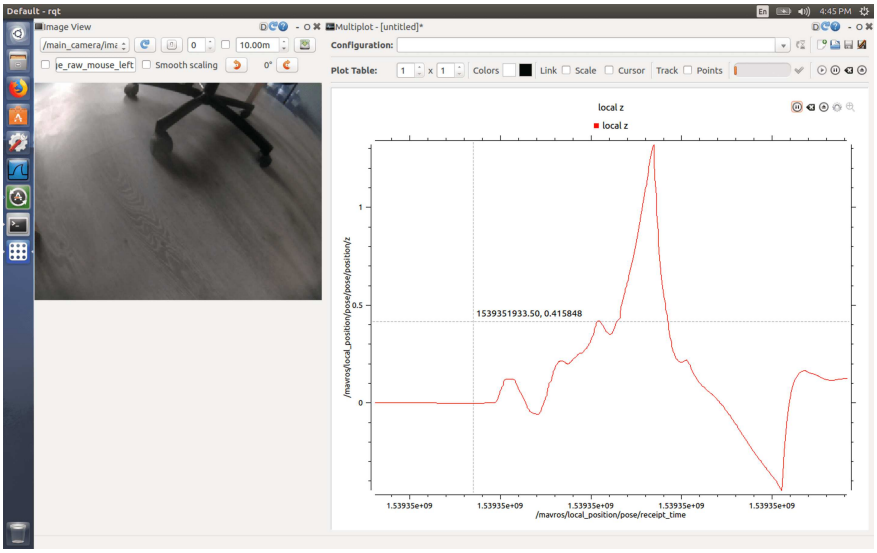
Axis или Grid настроенный на фрейм `aruco_map` будут визуализировать расположение карты ArUco-меток.

jsk_rviz_plugins

Рекомендуется также установка набора дополнительных полезных плагинов для rviz jsk_rviz_plugins. Это набор позволяет визуализировать топики типа `TwistStamped` (скорость), `CameraInfo`, `PolygonArray` и многое другое. Для установки используйте команду:

```
sudo apt-get install ros-melodic-jsk-visualization
```

Запуск инструментов rqt



Для запуска rqt для мониторинга состояния Клевера используйте команду:

```
ROS_MASTER_URI=http://192.168.11.1:11311 rqt
```

Пример запуск конкретного плагина (`rqt_image_view`):

```
ROS_MASTER_URI=http://192.168.11.1:11311 rqt_image_view
```

Краткое описание полезных rqt-плагинов:

- `rqt_image_view` – просмотр изображений из топиков типа `sensor_msgs/Image` ;
- `rqt_multiplot` – построение графиков по данным из произвольным топиков (установка: `sudo apt-get install ros-melodic-rqt-multiplot`);
- Bag – работа с Bag-файлами.

Автозапуск ПО

> В версии образа **0.20** пакет и сервис `clever` был переименован в `clover`. Для более ранних версий см. документацию для версии **0.19**.

systemd

Основная документация: https://wiki.archlinux.org/index.php/Systemd_(Русский).

Все автоматически стартуемое ПО Клевера запускается в виде systemd-сервиса `clover.service`.

Сервис может быть перезапущен командой `systemctl`:

```
sudo systemctl restart clover
```

Текстовый вывод ПО можно просмотреть с помощью команды `journalctl`:

```
journalctl -u clover
```

Для того чтобы запустить ПО Клевера непосредственно в текущей консольной сессии, вы можете использовать `roslaunch`:

```
sudo systemctl stop clover
roslaunch clover clover.launch
```

Вы можете выключить автозапуск ПО Клевера с помощью команды `disable`:

```
sudo systemctl disable clover
```

roslaunch

Основная документация: http://wiki.ros.org/roslaunch.

Список объявленных для запуска нод / программ указывается в файле `/home/pi/catkin_ws/src/clover/clover/launch/clover.launch`.

Вы можете добавить собственную ноду в список автозапускаемых. Для этого разместите ваш запускаемый файл (например, `my_program.py`) в каталог `/home/pi/catkin_ws/src/clover/clover`. Затем добавьте запуск вашей ноды в `clover.launch`, например:

```
<node name="my_program" pkg="clover" type="my_program.py" output="screen"/>
```

Запускаемый файл должен иметь *permission* на запуск:

```
chmod +x my_program.py
```

При использовании скриптовых языков вначале файла должен стоять shebang, например:

```
#!/usr/bin/env python3
```

Работа с ROS из браузера

С помощью библиотеки `roslibjs` возможна работа со всеми ресурсами ROS (топики, сервисы, параметры) из JavaScript-кода внутри браузера, что позволяет создавать различные интерактивные браузерные приложения для коптера.

Все необходимое для работы с `roslibjs` предустановлено и настроено на образе для RPi для Клевера.

Пример

Пример HTML-кода страницы, работающей с `roslib.js`:

```
<html>
    <script src="js/roslib.js"></script>
    <script type="text/javascript">
        // Establish roslibjs connection
        var ros = new ROSLIB.Ros({ url: 'ws://' + location.hostname + ':9090' }

        ros.on('connection', function () {
            // Connection callback
            alert('Connected');
        });

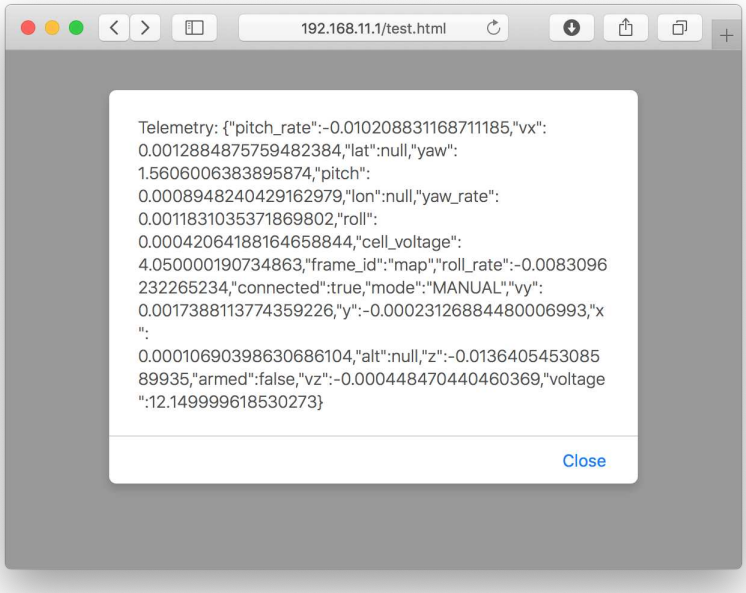
        // Declare get_telemetry service client
        var getTelemetry = new ROSLIB.Service({ ros: ros, name : '/get_telemetr

        // Call get_telemetry
        getTelemetry.callService(new ROSLIB.ServiceRequest({ frame_id: 'map' }),
            // Service respond callback
            alert('Telemetry: ' + JSON.stringify(result));
        });

        // Subscribe to `/mavros/state` topic
        var stateSub = new ROSLIB.Topic({ ros : ros, name : '/mavros/state', me
        stateSub.subscribe(function(msg) {
            // Topic message callback
            console.log('State: ', msg);
        });
    </script>
</html>
```

Взлет, посадка и все остальные операции могут быть осуществлены аналогичным образом.

Страница должна быть помещена в каталог `/home/pi/catkin_ws/src/clover/clover/www/`. После этого она станет доступна по адресу `http://192.168.11.1/clover/<имя_страницы>.html`. При открытии страницы браузер должен показать окно с телеметрией дрона, а также постоянно выводить состояние полетного контроллера в консоль.



Более подробную информацию смотрите в туториале по `roslibjs`.

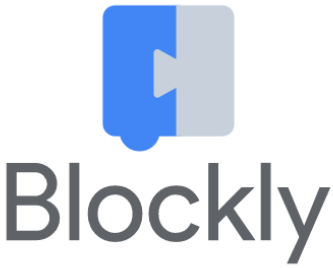
Браузерная GCS

Смотрите также пример реализации (`gcs.html`, `gcs.js`) упрощенной браузерной наземной станции (GCS) на Клевере по адресу http://192.168.11.1/clover/gcs.html.



Блочное программирование Клевера

Возможность блочного визуального программирования автономных полетов Клевера добавлена в образ для RPi, начиная с версии **0.21**. Реализация блочного программирования основана на Google Blockly. Интеграция Blockly в Клевер позволяет понизить входной порог в программирование автономных полетов до минимального уровня.



Конфигурация

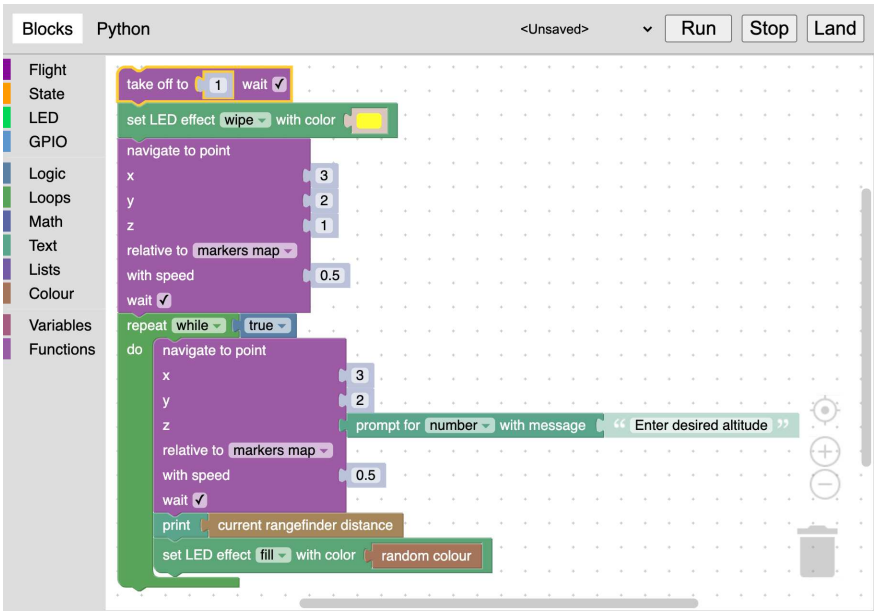
Для корректной работы работы блочного программирования аргумент `blocks` в launch-файле Клевера (`~/catkin_ws/src/clover/clover/launch/clover.launch`) должен быть в значении `true`:

```
<arg name="blocks" default="true"/>
```

Запуск

Для того, чтобы открыть интерфейс блочного программирования в Клевере, подключитесь к Клеверу по Wi-Fi и перейдите на страницу http://192.168.11.1/clover_blocks/ либо нажмите ссылку *Blocks programming* на основной веб-странице Клевера.

Интерфейс выглядит следующим образом:



Соберите необходимую программу из блоков в меню слева а затем нажмите кнопку *Run* для ее запуска. Также вы можете просмотреть сгенерированный код на языке Python, переключившись во вкладку *Python*.

Кнопка *Stop* позволяет остановить программу. Нажатие кнопки *Land* также останавливает программу и сажает дрон.

Сохранение и загрузка



Для сохранения программы откройте меню справа сверху, выберите пункт меню *Save* и введите название программы. Название программы может содержать только латинские буквы, дефис, подчеркивание и точку. Все ранее сохраненные программы будут доступны в этом же меню.

На карте памяти сохраненные XML-файлы программ хранятся в каталоге `/catkin_ws/src/clover/clover_blocks/programs/`.

В этом же меню доступны примеры программ (подкаталог `examples`).

Блоки

Набор блоков приблизительно аналогичен набору ROS-сервисов API автономных полетов Клевера. В этом разделе приведено описание некоторых из них.

Блоки Клевера поделены на 4 категории:

- Flight – команды, имеющие отношение к полету.
- State – блоки, позволяющие получить те или иные параметры текущего состояния коптера.

- LED – блоки для управления LED-лентой.
- GPIO – блоки для работы с GPIO-пинами.

В остальных категориях находятся стандартные блоки Google Blockly.

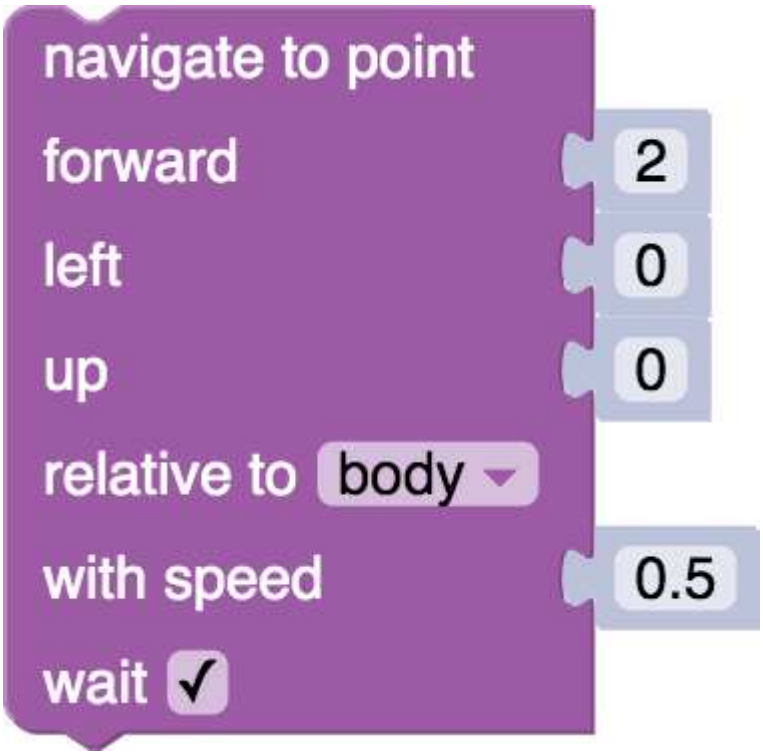
take_off



Взлететь на указанную высоту в метрах. Высота может быть произвольным блоком, возвращающим числовое значение.

Флаг `wait` определяет, должен ли дрон ожидать окончания взлета перед выполнением следующего блока.

navigate



Прилететь в заданную точку. Координаты точки задаются в метрах.

Флаг `wait` определяет, должен ли дрон ожидать завершения полета в точку перед выполнением следующего блока.

Поле *relative to*

В блоке может быть выбрана система координат, в которой задана целевая точка:

- *body* – координаты относительно коптера: вперед (*forward*), влево (*left*), вверх (*up*).
- *markers map* – система координат, связанная с картой ArUco-маркеров.
- *marker* – система координат, связанная с ArUco-маркером; появляется поле для ввода ID маркеа.
- *last navigate target* – координаты относительно последней заданной точки для навигации.
- *global* – глобальная система координат (широта и долгота) и относительная высота.
- *global, WGS 84 alt.* – глобальная система координат и высота в системе WGS 84.

land



Произвести посадку.

Флаг `wait` определяет, должен ли дрон ожидать окончания посадки перед выполнением следующего блока.

wait



Ожидать заданное время в секундах. Время ожидания может быть произвольным блоком, возвращающим числовое значение.

wait_arrival



Ожидать, пока дрон долетит до целевой точки (заданной в navigate-блоке).

get_position



Блок позволяет получить позицию, скорость и угол по рысканью дрона в заданной системе координат.

set_effect



Блок позволяет устанавливать различные анимации на LED-ленту аналогично ROS-сервису `set_effect`.

Пример использования блока для установки случайного цвета (блоки, связанные с цветами находятся в категории *Colour*):



Работа с GPIO

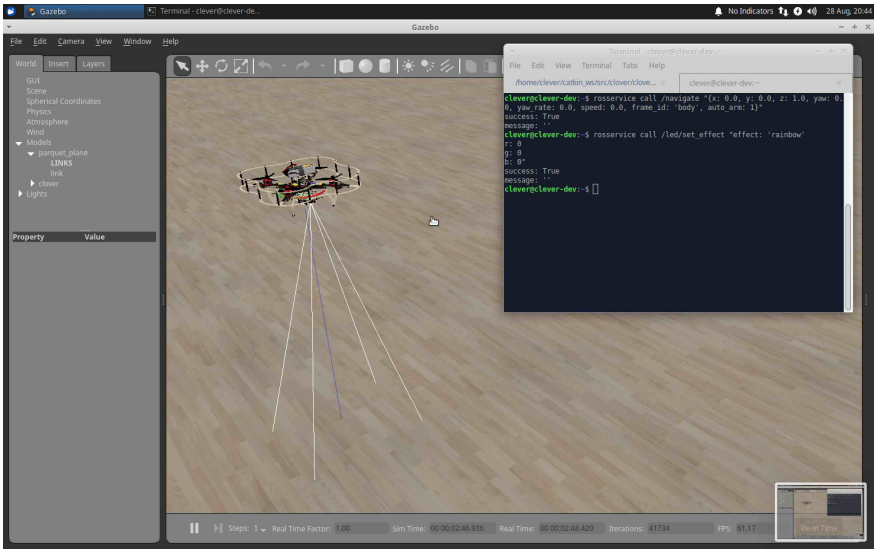
Категория GPIO содержит блоки для работы с GPIO. Обратите внимание, что для корректной работы этих блоков демон для работы с GPIO `pigpiod` должен быть включен:

```
sudo systemctl enable pigpiod.service
sudo systemctl start pigpiod.service
```

Более подробную информацию о GPIO читайте в соответствующей статье.

Симулятор

Среда симуляции Клевера позволяет пользователям запускать и отлаживать свой код в симуляторе, используя большинство функций, доступных на реальном дроне. Симулятор использует режим PX4 SITL и тот же код, использующий ROS, что и настоящий дрон. Большинство железа также симулируется.



Особенности

Устанавливаемая пользователем среда включает в себя:

- высококачественную модель Клевера 4;
- плагины Gazebo для железа Клевера (например, для светодиодной ленты);
- легко изменяемые файлы описания дрона в формате `xacro`;
- примеры моделей и миров;
- `roslaunch` файлы для быстрого запуска и настройки.

Кроме того, предоставляется образ виртуальной машины, который максимально точно имитирует реальный дрон.

Особенности:

- легкий доступ к симулятору;
- установлен и настроен для работы с ROS Visual Studio Code;
- веб-сервер (Monkey) для плагинов Клевера, работающих в браузере;
- постоянно работающий сервис `roscore`;
- средства визуализации (`rviz`, `rqt`).

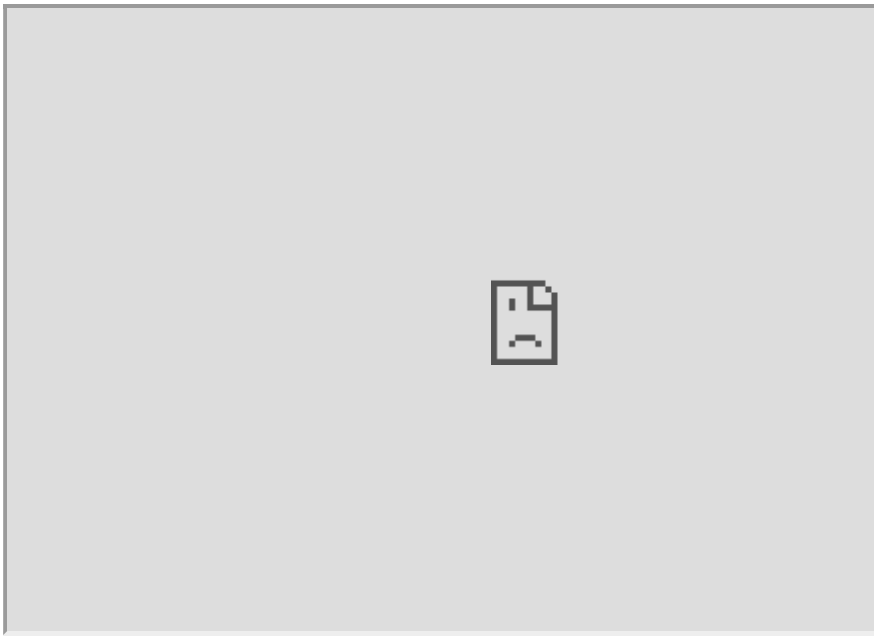
Состав симулятора

Симулятор основан на следующих элементах:

- Gazebo, универсальная среда симуляции для любых типов роботов;
- PX4, в частности, его компонент SITL (software-in-the-loop);
- `sitl_gazebo` пакет, содержащий плагины Gazebo для PX4;
- пакеты ROS и плагины Gazebo;

Видео

Краткий видеообзор симулятора:



Сборка на собственной машине

Настройка среды для симуляции с нуля требует некоторых усилий, однако это приведет к улучшению производительности и к уменьшению вероятности появления проблем с драйверами.

Требования для сборки: **Ubuntu 20.04**.

Установка ROS

Установите ROS Noetic используя официальную документацию по установке (Desktop или Full установка).

Добавьте выполнение инициализирующего скрипта ROS `setup.bash` в ваш файл `.bashrc`:

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Установите необходимые инструменты, которые понадобятся для дальнейшей установки:

```
sudo apt install build-essential git python3-pip python3-rosdep
```

Создание рабочего пространства для симулятора

Создайте рабочее пространство:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin_make
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Склонируйте исходный код пакетов Clover:

```
cd ~/catkin_ws/src
git clone --depth 1 https://github.com/CopterExpress/clover
git clone --depth 1 https://github.com/CopterExpress/ros_led
git clone --depth 1 https://github.com/ethz-asl/mav_comm
```

Установите все зависимости, используя `rosdep`:

```
cd ~/catkin_ws
sudo rosdep init
rosdep update
rosdep install --from-paths src --ignore-src -y
```

Установите Python-зависимости:

```
sudo /usr/bin/python3 -m pip install -r ~/catkin_ws/src/clover/clover/requireme
```

Загрузка исходного кода PX4

Сборка PX4 будет осуществлена вместе с другими пакетами в нашем рабочем пространстве. Вы можете загрузить его прямо в рабочее пространство или поместить куда-нибудь и создать симлинк к `~/catkin_ws/src`. Нам также нужно будет поместить его подмодули `sitl_gazebo` и `mavlink` в `~/catkin_ws/src`.

Склонируйте исходный код PX4 и создайте необходимые симлинки:

```
git clone --recursive --depth 1 --branch v1.12.3 https://github.com/PX4/PX4-Aut
ln -s ~/PX4-Autopilot ~/catkin_ws/src/
ln -s ~/PX4-Autopilot/Tools/sitl_gazebo ~/catkin_ws/src/
ln -s ~/PX4-Autopilot/mavlink ~/catkin_ws/src/
```

> Симулятор пока не поддерживает более поздние версии PX4, и при их использовании могут возникнуть ошибки.

> Если процесс клонирования завершится с ошибкой сети (`fatal: fetch-pack: invalid index-pack output`), используйте версию HTTP 1.1 `git config --global http.version HTTP/1.1` (после клонирования верните 2 версию командой `git config --global http.version HTTP/2`). Альтернативным решением будет принудительное клонирование репозитория и субмодулей через SSH командой `git config --global url."git@github.com:".insteadOf https://github.com/` (требует генерации и установки SSH ключа в настройках профиля GitHub).

Установка зависимостей PX4

PX4 имеет свой собственный скрипт для установки зависимостей. Воспользуемся им:

```
cd ~/catkin_ws/src/PX4-Autopilot/Tools/setup
sudo ./ubuntu.sh
```

Он установит все, что нужно для сборки PX4 и SITL.

> Также вы можете пропустить установку ARM тулчейна, если вы не планируете компилировать PX4 для вашего полетного контроллера. Для этого воспользуйтесь флагом `--no-nuttx`: `sudo ./ubuntu.sh --no-nuttx`.

Установите дополнительные необходимые Python-пакеты:

```
pip3 install --user toml
```

Добавление рамы Клевера

Добавьте в PX4 раму Клевера с помощью следующей команды:

```
ln -s ~/catkin_ws/src/clover/clover_simulation/airframes/* ~/PX4-Autopilot/ROMF
```

Установка датасетов geographiclib

Для `mavros` нужны датасеты geographiclib:

```
sudo /opt/ros/noetic/lib/mavros/install_geographiclib_datasets.sh
```

Сборка симулятора

После установки всех зависимостей можно начинать сборку рабочего пространства:

```
cd ~/catkin_ws
catkin_make -j1
```

> Флаг `-j1` означает, что сборка не будет использовать параллельные процессы, так как при сборке с параллельными процессами на виртуальной машине может не хватить оперативной памяти. Если у вас достаточно памяти, вы можете не использовать этот флаг.

Запуск симулятора

Чтобы удостовериться в том, что все было собрано корректно, попробуйте запустить симулятор:

```
roslaunch clover_simulation simulator.launch
```

Вы можете проверить автономный полет используя скрипты в директории `~/catkin_ws/src/clover/clover/examples`.

Дополнительные шаги

Для того, чтобы возможно было запускать среду симуляции Gazebo отдельно (команда `gazebo`), добавьте в `.bashrc` вызов соответствующего скрипта инициализации:

```
echo "source /usr/share/gazebo/setup.sh" >> ~/.bashrc
```

Опционально вы можете установить systemd-сервис для roscore для того, чтобы roscore был постоянно запущен в фоне:

```
sed -i "s/pi/$USER/g" ~/catkin_ws/src/clover/builder/assets/roscore.service
sudo cp ~/catkin_ws/src/clover/builder/assets/roscore.service /etc/systemd/syst
sudo systemctl enable roscore
sudo systemctl start roscore
```

Конфигурация веб-инструментов

Установите любой веб-сервер, чтобы раздавать веб-инструменты Клевера (директория `~/.ros/www`), например, Monkey:

```
wget https://github.com/CopterExpress/clover_vm/raw/master/assets/packages/monk
sudo dpkg -i /tmp/monkey_*.deb
sed "s/pi/$USER/g" ~/catkin_ws/src/clover/builder/assets/monkey | sudo tee /etc
sudo sed -i 's/SymLink Off/SymLink On/' /etc/monkey/monkey.conf
sudo cp ~/catkin_ws/src/clover/builder/assets/monkey.service /etc/systemd/syste
sudo systemctl enable monkey
sudo systemctl start monkey
```

Создайте директорию `~/.ros/www` следующей командой:

```
rosrun clover www
```

При обновлении набора пакетов, содержащих веб-часть (через каталог `www`), также необходимо выполнение данной команды.

Установка виртуальной машины

Для работы с платформой Клевер рекомендуется иметь установленное окружение ROS на своём компьютере. К сожалению, установка ROS и симулятора сопряжена с рядом трудностей: требуется использовать операционную систему Ubuntu 20.04, процесс установки длительный и требует выполнения большого количества команд в терминале.

Для облегчения процесса настройки окружения мы предлагаем использовать виртуальную машину со всем необходимым для работы с платформой Клевер. В состав виртуальной машины входят:

- операционная система Ubuntu 20.04 с легковесной графической оболочкой XFCE;
- предустановленные пакеты ROS для работы с Клевером;
- QGroundControl;
- предварительно настроенный симулятор Gazebo;
- среда разработки Visual Studio Code с плагинами для разработки на Python и C++.

> Имя пользователя по умолчанию на виртуальной машине - `clover`, пароль - `clover`.

Виртуальная машина может использоваться как для запуска симуляторов, так и для работы с настоящим дроном.

Скачивание

Скачать текущую версию виртуальной машины можно в релизах репозитория виртуальной машины.

Установка виртуальной машины

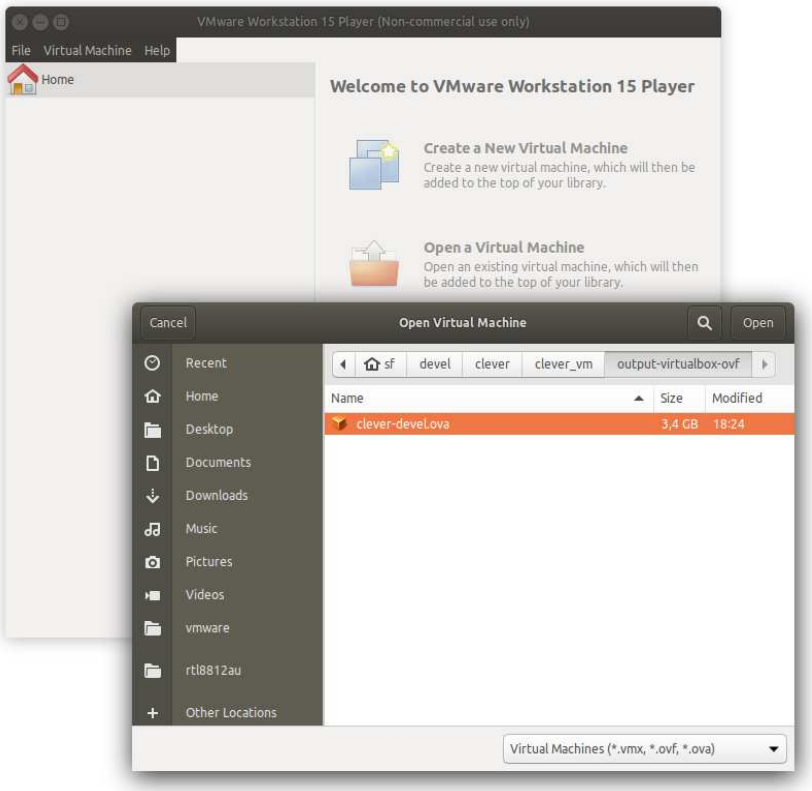
Для запуска виртуальной машины разработчика требуется использовать одну из совместимых сред виртуализации: VirtualBox, VMware Player, VMware Workstation.

> На момент написания данной статьи VirtualBox не обеспечивал достаточный уровень совместимости с виртуальной машиной. Рекомендуется по возможности использовать VMware Player или VMware Workstation; дальнейшая инструкция будет преимущественно написана для VMware Player.

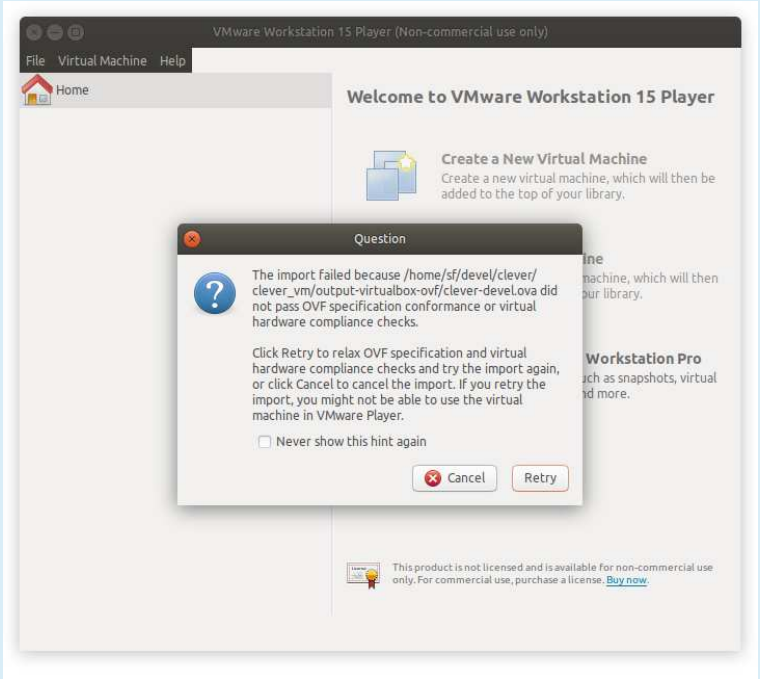
Убедитесь, что поддержка аппаратной виртуализации включена в настройках BIOS/UEFI вашего компьютера. Шаги для включения аппаратной виртуализации, как правило, описаны в руководстве пользователя

компьютера. Проконсультируйтесь с производителем компьютера, если включить виртуализацию не получается.

1. Импортируйте архив виртуальной машины в среду виртуализации. Для VMware Player используйте опцию **Open a Virtual Machine**:



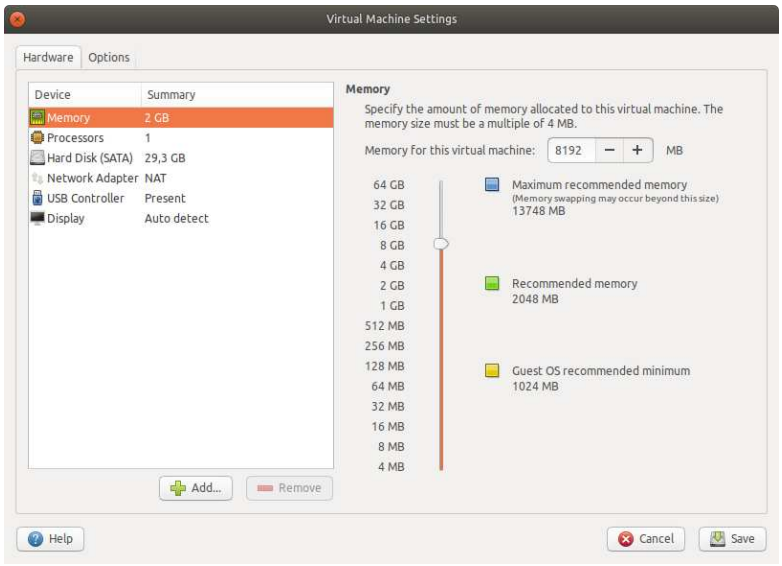
При импорте архива, скорее всего, появится окно с предупреждением о формате виртуальной машины:



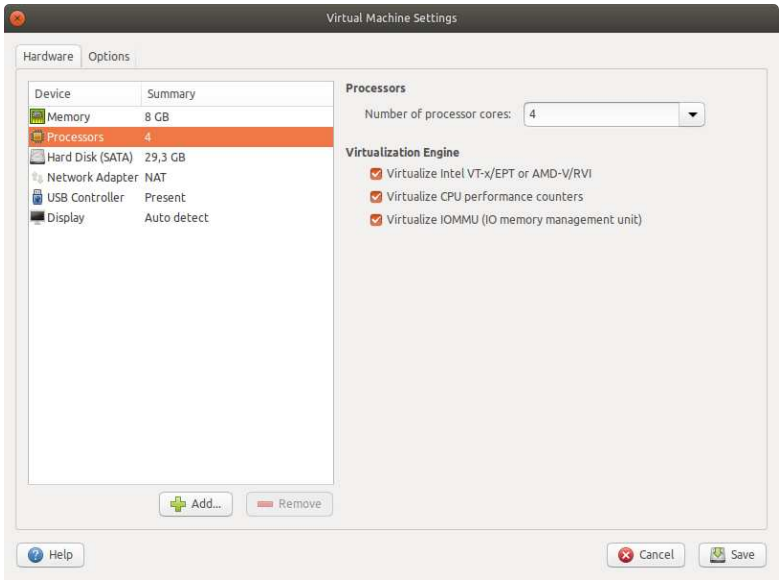
Это предупреждение можно игнорировать и нажать кнопку **Retry**.

2. Откройте окно настроек виртуальной машины и измените параметры для наилучшего соответствия основной системе:

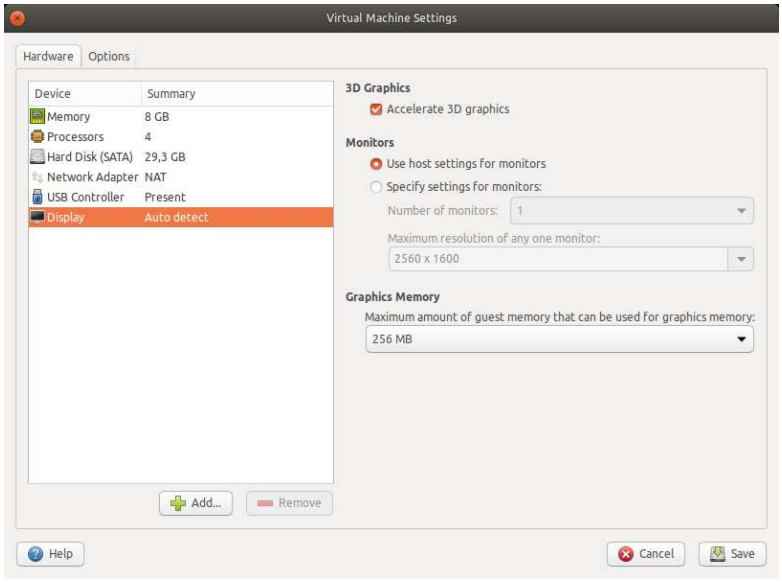
   - увеличьте объём оперативной памяти, отводимый для виртуальной машины:



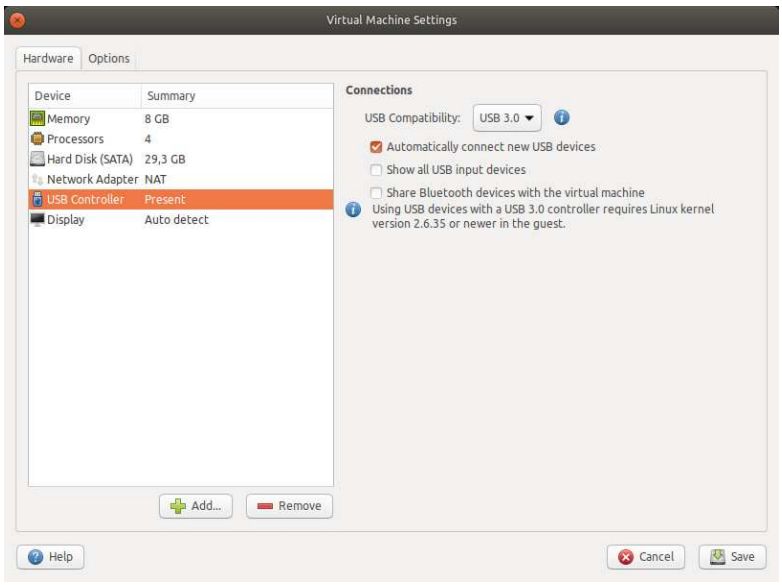
   - увеличьте количество доступных процессорных ядер:



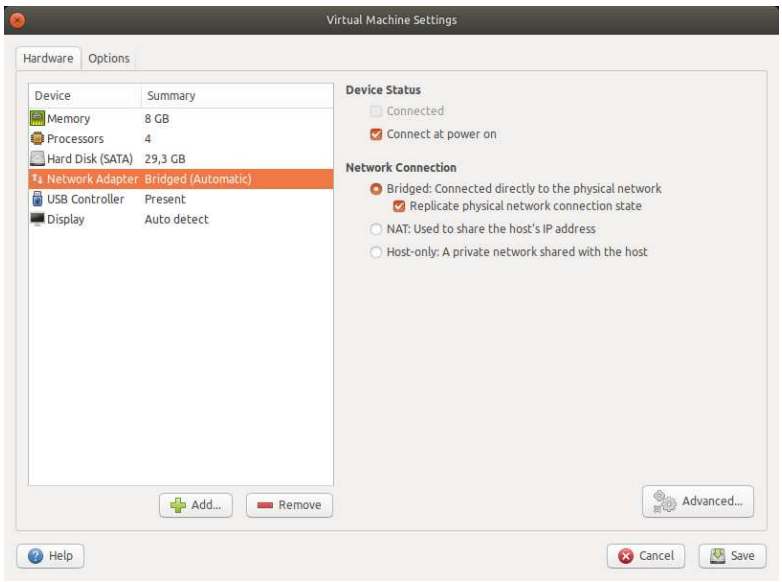
- включите 3D-ускорение:



- включите использование USB 2.0/3.0:

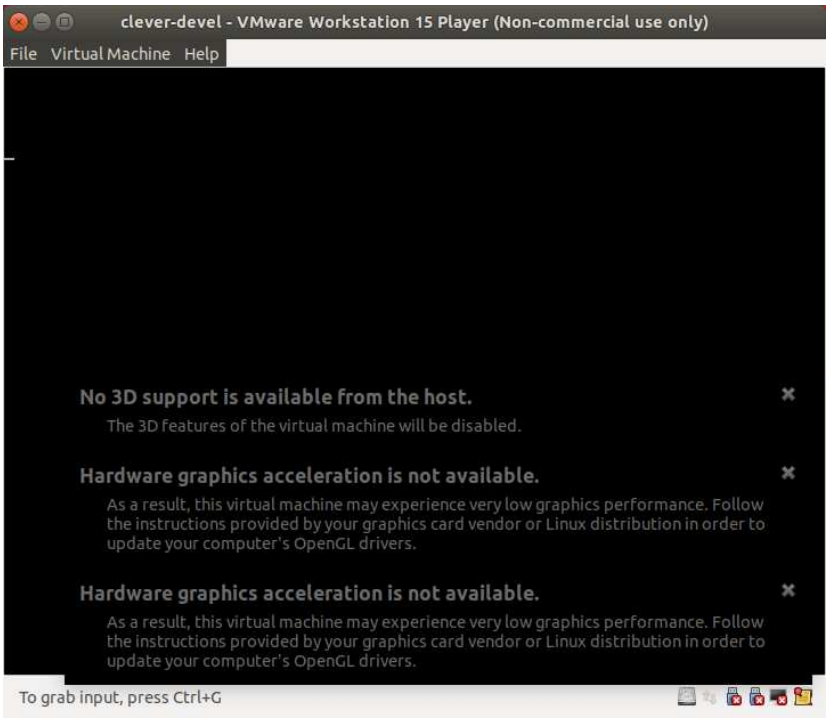


- опционально включите режим "мост" для виртуального сетевого адаптера:



> Режим "мост" может некорректно работать с некоторыми сетевыми адаптерами. Если в режиме "мост" вы не можете подключиться к дрону, используйте USB Wi-Fi-адаптеры, "проброшенные" в виртуальную машину.

3. Запустите виртуальную машину. Возможно, при первом запуске справа появятся сообщения об отсутствии поддержки 3D-ускорения со стороны основной системы:



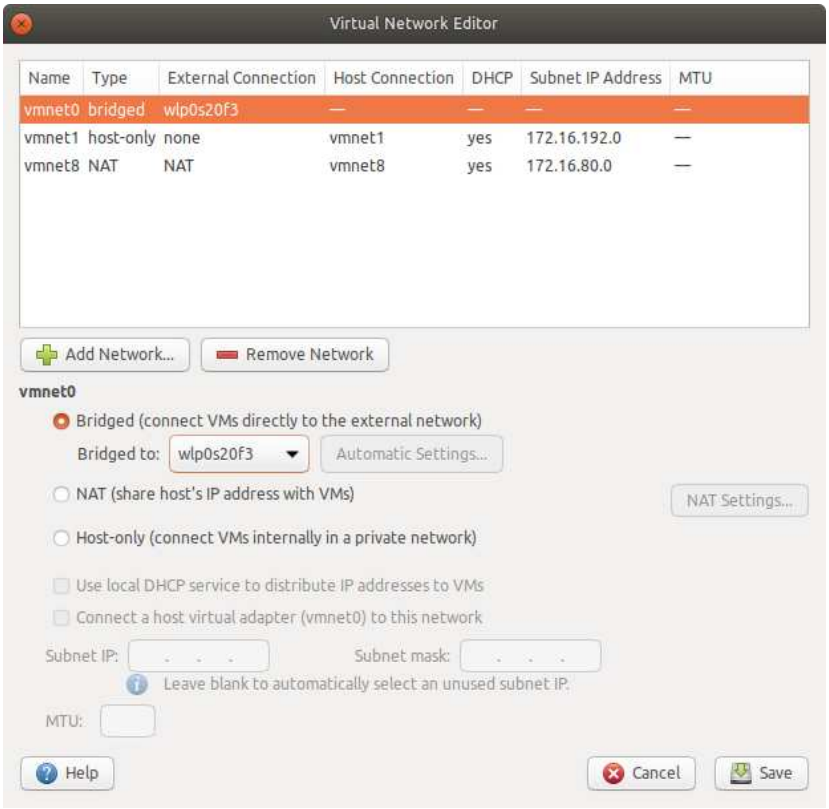
В этом случае убедитесь, что у вас установлены самые последние драйверы для видеокарты в основной системе. Если сообщения появляются при повторных запусках виртуальной машины, добавьте

строку

```
mks.gl.allowBlacklistedDrivers = "TRUE"
```

в файл `clever-devel.vmx`, находящийся в папке, в которую был импортирован архив в п. 1.

4. Настройте режим моста через настройки виртуальной машины (если используется VMware Player для Windows) или с помощью утилиты `vmware-netcfg` (если используется версия для Linux-дистрибутивов):



В списке сетей выберите `vmnet0`, ниже - режим *Bridged*, в выпадающем списке *Bridged to* - название беспроводного адаптера, с помощью которого будет производиться подключение к дрону.

Использование симулятора

Среда симуляции Клевера позволяет пользователям тестировать свой код без какого-либо риска повреждения оборудования. Кроме того, в виртуальной машине по умолчанию запущены дополнительные (не относящиеся к ROS) сервисы, которые присутствуют на реальном дроне, например, веб-сервер Monkey.

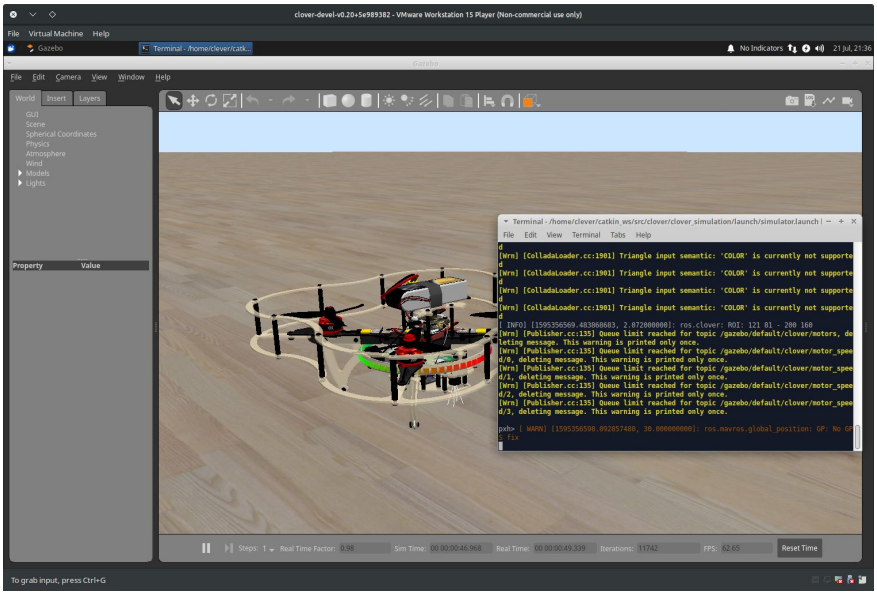
Запуск симулятора

После сборки симулятора с нуля или запуска виртуальной машины, вы можете использовать `roslaunch` для запуска симулятора Gazebo:

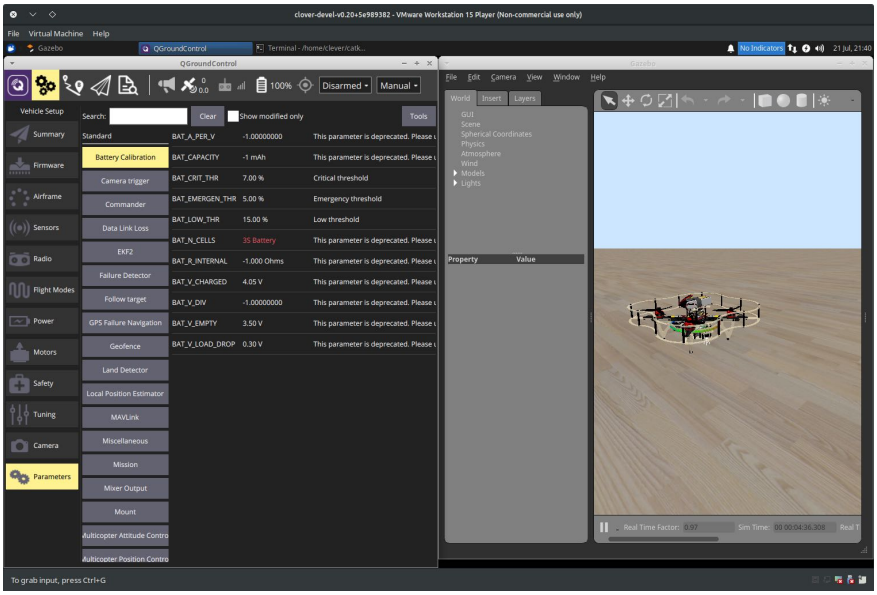
```
# Не забудьте сначала активировать ваше рабочее пространство
source ~/catkin_ws/devel/setup.bash
roslaunch clover_simulation simulator.launch
```

> Кроме того, если вы используете виртуальную машину, просто дважды щелкните `Gazebo PX4` на рабочем столе.

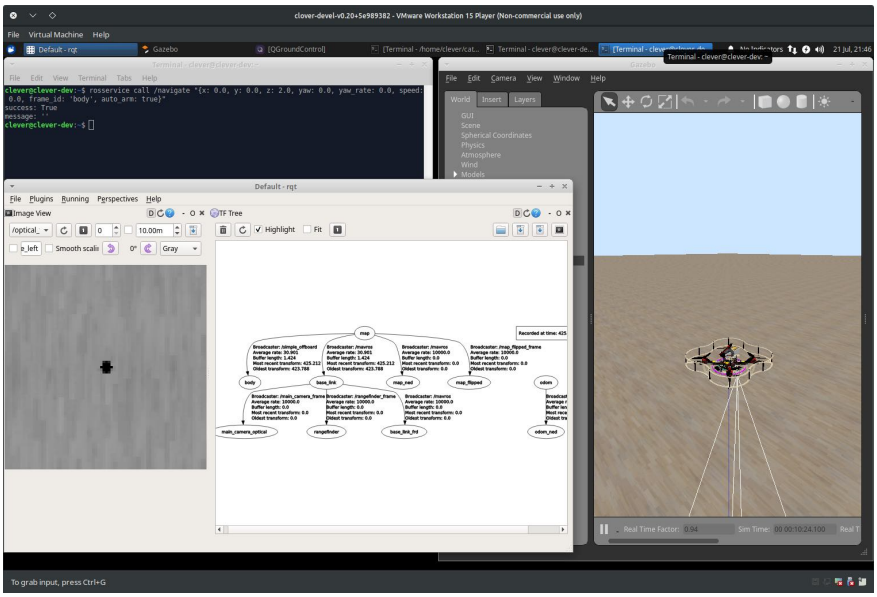
Это запустит Gazebo сервер и клиент, бинарные файлы PX4 и ноды Клевера. Терминал, в котором была запущена команда, будет отображать отладочные сообщения от нод и PX4, а также принимать входные данные для интерпретатора команд PX4:



Вы также можете использовать QGroundControl для настройки параметров дрона в симуляторе, планирование миссий полета (если GPS также симулируется) и управление дроном с пульта:

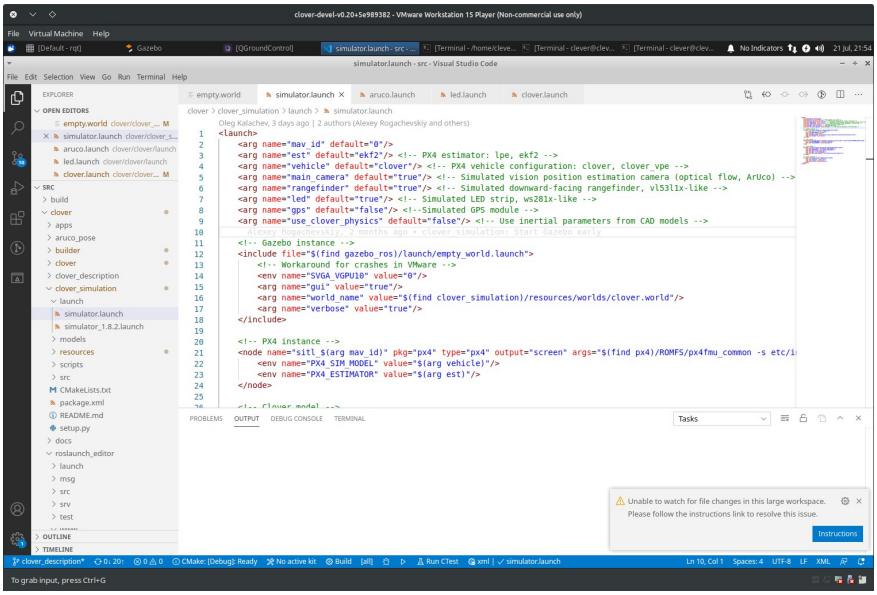


Также вы можете использовать simplified OFFBOARD для управления дроном, и средства визуализации ROS, например, rviz and rqt для анализа состояния дрона:



Конфигурация симулятора

Симулятор можно настроить, передав дополнительные аргументы команде `roslaunch` или изменив файл `~/catkin_ws/src/clover/clover_simulation/launch/simulator.launch`. Ноды, обеспечивающие распознавание ArUco, расчет optical flow и другие сервисы могут быть настроены изменением соответствующих `.launch` файлов, как на реальном дроне.



Включение GPS

GPS датчик необходим полетов с использованием глобальных координат, а также для использования полетных миссий.

Чтобы включить GPS, установите аргумент `gps` в файле `simulator.launch` в значение `true`:

```
<arg name="gps" value="true"/>
```

Включите также флажок *use GPS* в параметре PX4 `EKF2_AID_MASK` (используя QGroundControl).

Камера

При полете по GPS камера может быть не нужна, ее также можно отключить в файле `simulator.launch`:

```
<arg name="main_camera" default="false"/>
```

Другие датчики

Для того, чтобы добавить датчики или изменить их расположение, необходимо изменить файл описания дрона. Этот файл находится в `~/catkin_ws/src/clover/clover_description/urdf/clover/clover4.xacro` и использует формат xacro для сборки описания URDF.

Изменение мира по умолчанию

Плагины Gazebo для дрона на данный момент требуют, чтобы значение параметра мира `real_time_update_rate` было равно 250, а значение `max_step_size` было равно 0.004. С другими параметрами симулятор не

заработает. Используйте файл `~/catkin_ws/src/clover/clover_simulation/resources/worlds/clover.world` как шаблон.

Повышение производительности

Симуляция с использованием Gazebo требовательна к ресурсам, и для нормальной скорости работы требуются мощный процессор и видеокарта. При этом симуляции можно запускать и на менее мощном оборудовании, жертвуя при этом скоростью работы. Ниже приведены рекомендации для компьютеров, на которых симуляция не может работать в реальном времени.

Использование плагина `throttling_camera`

По умолчанию Gazebo не замедляет симуляцию для достижения требуемой частоты работы визуальных сенсоров. Это можно исправить с помощью плагина `throttling_camera` из пакета `clover_simulation`.

Включение этого плагина происходит путём выставления параметра `maintain_camera_rate` в значение `true` в файле `clover_description/launch/spawn_drone.launch`:

```
    <!-- Slow simulation down to maintain camera rate -->
    <arg name="maintain_camera_rate" default="true"/>
```

Этот плагин будет собирать статистику по частоте публикации изображений, и будет замедлять симуляцию до тех пор, пока частота публикации не станет соответствовать или превышать требуемую. При этом значительные замедления симуляции могут негативно сказаться на программном обеспечении, которое подключается к PX4 (например, QGroundControl). Если скорость симуляции опускается ниже, чем 0.5 от реального времени, следует воспользоваться следующей рекомендацией.

Выставление скорости симуляции

PX4, начиная с версии 1.9, поддерживает принудительную установку скорости симуляции с помощью переменной окружения `PX4_SIM_SPEED_FACTOR`. Выставление этой переменной подготавливает все компоненты симулятора к соответствующему ускорению/замедлению.

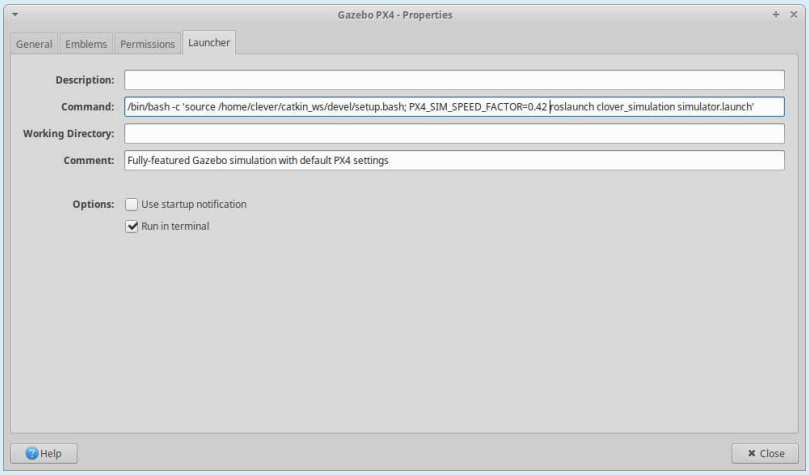
Значение этой переменной должно соответствовать величине Real Time Factor (скорости симуляции по отношению к реальному времени), получаемой при использовании `throttling_camera`. Величина Real Time Factor отображается в окне Gazebo на нижней панели:



В данном случае `PX4_SIM_SPEED_FACTOR` следует выставить в значение `0.42` перед запуском симулятора:

```
PX4_SIM_SPEED_FACTOR=0.42 roslaunch clover_simulation simulator.launch
```

При использовании виртуальной машины удобнее добавить эту переменную в ярлык запуска Gazebo на рабочем столе. Нажмите правой кнопкой на значок Gazebo, выберите "Properties..." и добавьте `PX4_SIM_SPEED_FACTOR=0.42` в поле Command, как показано на иллюстрации:



Выделение ресурсов для виртуальной машины

Выделение нескольких процессорных ядер для виртуальной машины может значительно повысить производительность симуляции. В наших испытаниях виртуальная машина, для которой было выделено одно ядро, не позволяла работать в симуляторе: окно Gazebo не реагировало на пользовательский ввод, сообщения ROS терялись. После выделения четырёх ядер для этой же виртуальной машины симуляция стала работать со скоростью 0.25 от реального времени.

При этом не следует пытаться выделить для виртуальной машины больше ресурсов, чем доступно на основной системе.

Изменение карты ArUco-меток в симуляторе

Для того, чтобы изменить карту ArUco-меток в симуляторе, можно использовать следующую команду:

```
rosrun clover_simulation aruco_gen --single-model --source-world=$(catkin_find
```

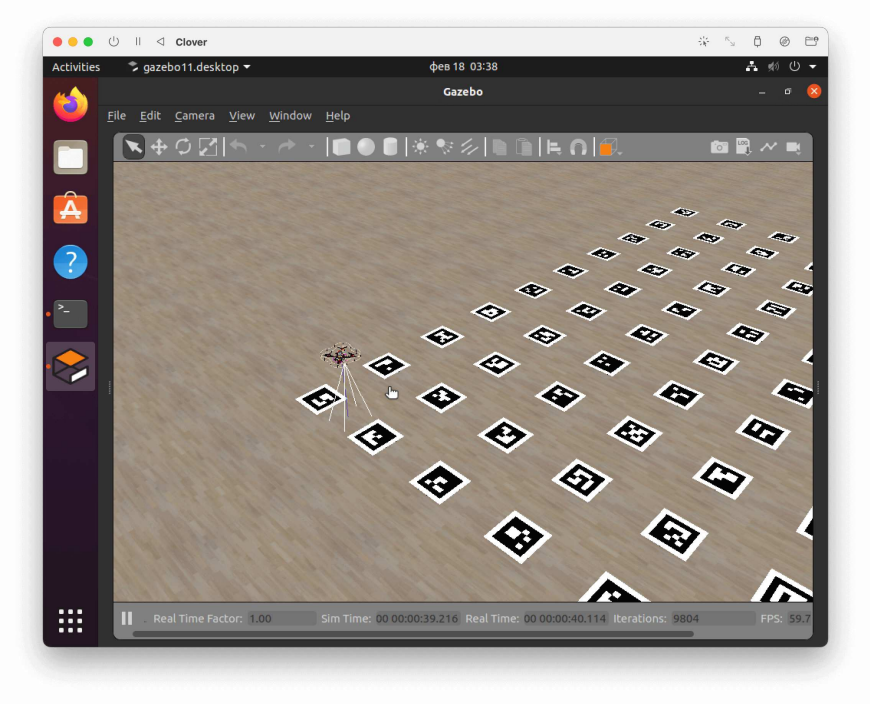
В данном примере `map.txt` – имя карты меток.

Симулятор на компьютерах с чипом M1

Для архитектуры ARM64, которую используют компьютеры с чипом M1 (Apple Silicon), готовый образ с симулятором не выпускается, поэтому возможна только ручная установка симулятора.

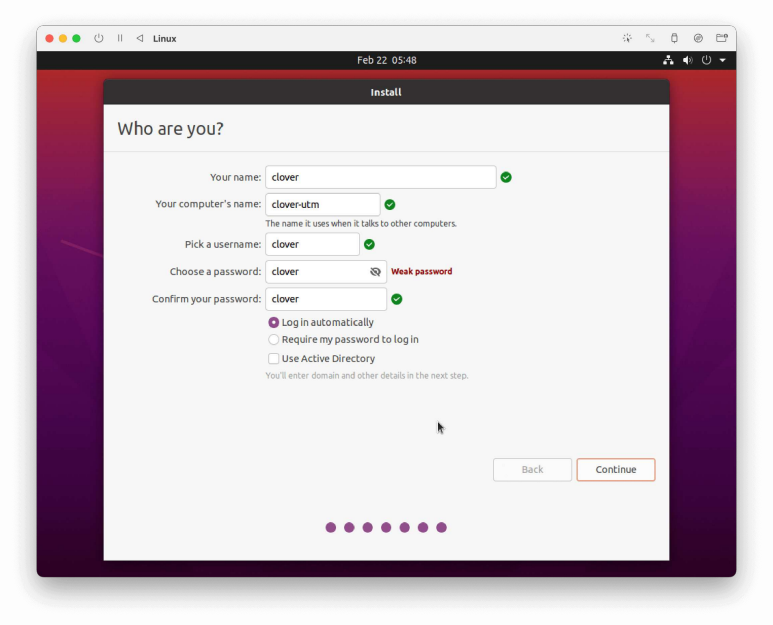
В качестве виртуальной машины рекомендуется использовать бесплатное приложение **UTM**. Также возможно использование **VMware Fusion Public Tech Preview** с поддержкой M1.

Установка симулятора с UTM



1. Скачайте UTM с официального сайта mac.getutm.app и установите его.
2. Скачайте исходный образ установщика Ubuntu 20.04 для архитектуры ARM64 по ссылке: https://clovervm.ams3.digitaloceanspaces.com/focal-desktop-arm64.iso.
3. Создайте новую виртуальную машину в UTM, выбирая следующие настройки:
   - **Тип**: Virtualize.
   - **Operating System** (ОС): Linux.
   - **Boot ISO Image** (образ для загрузки): выберите скаченный образ `focal-desktop-arm64.iso`.
   - **Memory** (память): 4096 MB или более.
   - **CPU Cores** (ядра процессора): 4 или более.
   - Включите поддержку OpenGL: *Enable hardware OpenGL acceleration*.

   - **Storage** (размер хранилища): 20 GB или более.
4. Запустите созданную виртуальную машину.
5. Выберите пункт *Install Ubuntu* и установите Ubuntu с помощью мастера установки.
   - Рекомендуемый набор ПО: *Minimal installation*.
   - Тип установки: *Erase disk and install Ubuntu*.
   - Введите параметры учетной записи по желанию, например:



6. Завершите установку и запустите установленную систему (для этого потребуется извлечь виртуальный CD-диск или выбрать *Boot from next volume* в меню загрузки).
7. Установите симулятор согласно инструкции по сборке симулятора на собственной машине.

Возможные проблемы при установке

Отсутствие картинки

Если при запуске виртуальной машины вместо изображения вы видите черный фон, попробуйте запустить машину с отключенной поддержкой GPU.

В настройках виртуальной машины выберите *Display*, в пункте *Emulated Display Card* выберите *virtio-ramfb*. Запустите машину. При успешном запуске поменяйте настройку обратно на *virtio-ramfb-gl (GPU Supported)* и снова запустите машину.

Проблема с `git clone`

При осуществлении команды `git clone` может возникнуть подобная ошибка:

```
on git clone if error: RPC failed; curl 56 GnuTLS recv error (-54): Error in th
fatal: the remote end hung up unexpectedly
fatal: early EOF
fatal: index-pack failed
```

В этом случае поменяйте типа сетевой карты на Bridged. В настройках виртуальной машины выберите *Network*, в пункте *Network Mode* выберите *Bridged (Advanced)*.

В дальнейшем, при возникновении проблем с сетью измените тип сети обратно на *Shared Network*.

ROS



Основная документация: https://wiki.ros.org.

ROS – это широко используемый фреймворк для создания сложных, распределенных робототехнических систем. На ROS основана программная платформа Клевера.

Установка

ROS уже установлен на образе для RPi для Клевера.

Для установки инструментов ROS на компьютере вы можете обратиться к официальной документации по установке. Для быстрого старта рекомендуется воспользоваться образом виртуальной машины с ROS и симулятором Клевера.

Концепции

Ноды

Основная статья: https://wiki.ros.org/Nodes.

ROS-нода[1] – это специальная программа (обычно написанная на Python или C++), которая взаимодействует с другими нодами посредством ROS-топиков и ROS-сервисов. Разделение сложных робототехнических систем на изолированные ноды дает определенные преимущества: понижается связанность кода, повышается переиспользуемость и надежность.

Очень многие робототехнические библиотеки и драйвера выполнены именно в виде ROS-нод.

Для того, чтобы превратить обычную программу в ROS-ноду, необходимо подключить к ней библиотеку `rospy` (Python) или `roscpp` (C++) и добавить инициализирующий код.

Пример ROS-ноды на языке Python:

```
import rospy

rospy.init_node('my_ros_node')  # имя ROS-ноды

rospy.spin()  # входим в бесконечный цикл...
```

> Любая программа для автономного полета Клевера является ROS-нодой.

Топики

Основная статья: https://wiki.ros.org/Topics.

Топик – это именованная шина данных, по которой ноды обмениваются сообщениями. Любая нода может *опубликовать* сообщение в произвольный топик, а также *подписаться* на произвольный топик.

Для каждого созданного топика должен быть задан тип сообщений, которые по нему передаются. ROS включает в себя большое количество стандартных типов сообщений, покрывающих различные аспекты робототехники, но при необходимости возможно создание собственных типов сообщений. Примеры стандартных типов сообщений:

Тип сообщения	Описание
`std_msgs/Int64`	Целое число.
`std_msgs/Float64`	Число с плавающей точкой (дробное) двойной точности.
`std_msgs/String`	Строка.
`geometry_msgs/PoseStamped`	Позиция и ориентация объекта с заданной системой координат и временной меткой (широко используется для передачи текущей позиции робота и его частей).
`geometry_msgs/TwistStamped`	Линейная и угловая скорость объекта с заданной системой координат и временной меткой.
`sensor_msgs/Image`	Изображение (см. статью о работе с камерой)

> Смотрите остальные стандартные типы сообщений в пакетах `common_msgs`, `std_msgs`, `geometry_msgs`, `sensor_msgs` и других.

Пример публикации сообщения типа `std_msgs/String` (строка) в топик `/foo` на языке Python:

```
import rospy
from std_msgs.msg import String

rospy.init_node('my_ros_node')

foo_pub = rospy.Publisher('/foo', String, queue_size=1)  # создаем Publisher

foo_pub.publish(data='Hello, world!')  # публикуем сообщение
```

Пример подписки на топик `/foo`:

```
import rospy
from std_msgs.msg import String

rospy.init_node('my_ros_node')

def foo_callback(msg):
    print(msg.data)

# Подписываемся. При получении сообщения в топик /foo будет вызвана функция foo
rospy.Subscriber('/foo', String, foo_callback)

rospy.spin()  # входим в бесконечный цикл, чтобы программа не завершила работу
```

Вы можете прочитать данные из топика однократно, используя функцию `wait_for_message`:

```
msg = rospy.wait_for_message('/foo', String, timeout=3)  # ждать сообщения в то
```

Также существует возможность работы с топиками с помощью утилиты `rostopic`. Например, с помощью следующей команды можно просматривать сообщения, публикуемые в топик `/mavros/state`:

```
rostopic echo /mavros/state
```

Команда `rostopic info` позволяет узнать тип сообщений в топике, команда `rostopic hz` — частоту публикуемых в топике сообщений.

Также данные в топиках можно визуализировать и в графических инструментах ROS.

Сервисы

Основная статья: https://wiki.ros.org/Services.

Сервис – это некоторый аналог функции, которая может быть вызвана из одной ноды, а обработана в другой. У сервиса есть имя, аналогичное имени топика, и 2 типа сообщений: тип запроса и тип ответа.

Таким образом, сервисы реализуют паттерн *удаленного вызова процедур*.

Пример вызова ROS-сервиса из языка Python:

```
import rospy
from clover.srv import GetTelemetry

rospy.init_node('my_ros_node')

# Создаем обертку над сервисом get_telemetry пакета clover с типом GetTelemetry
get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)

# Вызываем сервис и получаем телеметрию квадрокоптера:
telemetry = get_telemetry()
```

С сервисами можно также работать при помощи утилиты `rosservice`. Так можно вызвать сервис `/get_telemetry` из командной строки:

```
rosservice call /get_telemetry "{frame_id: ''}"
```

Больше примеров использования сервисов для автономных полетов квадрокоптера Клевер можно посмотреть в документации ноды simple_offboard.

Имена

Основная статья: https://wiki.ros.org/Names.

Любой топик, сервис или параметр идентифицируется с помощью уникального имени. ROS-имя представляет собой иерархическую структуру с символом `/` в качестве разделителя (сходно с именами в файловой системе).

Примеры ROS-имен:

- `/` (глобальное пространство имен)
- `/foo`
- `/stanford/robot/name`
- `/wg/node1`

Эти имена является глобальными (аналогично полному пути в файлу в файловой системе). На практике рекомендуется использование *приватных* или *относительных* имен.

Приватное имя

Каждая нода может использовать собственное приватное пространство имен (соответствующее имени ноды) для своих ресурсов. Например, нода `aruco_detect` может публиковать такие топики:

- `/aruco_detect/markers`
- `/aruco_detect/visualization`
- `/aruco_detect/debug`

Когда нода ссылается на свой приватный ресурс, вместо пространства имен (`/aruco_detect/`) используется символ `~`, например:

- `~markers`
- `~visualization`
- `~debug`

Таким образом, создание топика `foo` в приватном пространство имен из Python будет выглядеть так:

```
private_foo_pub = rospy.Publisher('~foo', String, queue_size=1)
```

Относительное имя

Несколько нод также могут объединяться в общее пространство имен (например, при одновременной работе нескольких роботов). Для того, чтобы ссылаться на топики с учетом общего пространства имен, в названии ресурса опускается начальный символ `/`.

Пример создание топика `foo` с учетом общего пространства имен:

```
relative_foo_pub = rospy.Publisher('foo', String, queue_size=1)
```

> В общем случае всегда рекомендуется использовать приватные или относительные имена ресурсов и никогда не использовать глобальные.

Работа на нескольких машинах

Основная статья: https://wiki.ros.org/ROS/Tutorials/MultipleMachines.

Преимуществом использования ROS является возможность распределения нод на несколько машин в сети. Например, ноду, осуществляющую распознавание образом на изображении можно запустить на более мощном компьютере; ноду, управляющую коптером можно запустить непосредственно на Raspberry Pi, подключенном к полетному контроллеру и т. д.

Дополнительные материалы

- Учебник по ROS от Voltbro - http://docs.voltbro.ru/starting-ros/.
- Другие книги по ROS - https://wiki.ros.org/Books.

[1]. Также встречается перевод "узел". ↩

MAVROS

Основная документация: http://wiki.ros.org/mavros

MAVROS (MAVLink + ROS) — это пакет для ROS, предоставляющий возможность управлять беспилотниками по протоколу MAVLink. MAVROS поддерживает полетные стеки PX4 и APM. Связь организовывается по UART, USB, TCP или UDP.

MAVROS подписывается на определенные ROS-топики в ожидании команд, публикует в другие топики телеметрию, и предоставляет сервисы.

Нода MAVROS автоматически запускается в launch-файле Клевера. Для настройки типа подключения см. аргумент `fcu_conn`.

> Упрощенное взаимодействие с коптером возможно с использованием пакета `simple_offboard`.

> В пакете `clover` некоторые плагины MAVROS отключены (в целях сохранения ресурсов). Подробнее см. параметр `plugin_blacklist` в файле `/home/pi/catkin_ws/src/clover/clover/launch/mavros.launch`.

Основные сервисы

`/mavros/set_mode` — установить полетный режим контроллера. Обычно устанавливается режим OFFBOARD (для управления с Raspberry Pi).

`/mavros/cmd/arming` — включить или выключить моторы беспилотника (изменить armed-статус).

Основные публикуемые топики

`/mavros/state` — статус подключения к полетному контроллеру. Режим полетного контроллера.

`/mavros/local_position/pose` — локальная позиция коптера в системе координат ENU и его ориентация.

`/mavros/local_position/velocity` — текущая скорость в локальных координатах. Угловые скорости.

`/mavros/global_position/global` — текущая глобальная позиция (широта, долгота, высота).

`/mavros/global_position/local` — глобальная позиция в системе координат UTM.

`/mavros/global_position/rel_alt` — относительная высота (относительно высоты включения моторов).

Просмотр сообщений, публикуемых в топики возможен с помощью утилиты `rostopic`, например `rostopic echo /mavros/state`. Подробнее см. работа с ROS.

Основные топики для публикации

`/mavros/setpoint_position/local` — установить целевую позицию и рысканье (yaw) беспилотника (в системе координат ENU).

`/mavros/setpoint_position/global` – установить целевую позицию в глобальных координатах (ширина, долгота и высота) и рысканье беспилотника.

`/mavros/setpoint_velocity/cmd_vel` — установить целевую линейную скорость беспилотника.

`/mavros/setpoint_attitude/attitude` и `/mavros/setpoint_attitude/att_throttle` — установить целевую ориентацию (Attitude) и уровень газа.

`/mavros/setpoint_attitude/cmd_vel` и `/mavros/setpoint_attitude/att_throttle` — установить целевые угловые скорости и уровень газа.

Топики для посылки raw-пакетов

`/mavros/setpoint_raw/local` — отправка пакета SET_POSITION_TARGET_LOCAL_NED. Позволяет установить целевую позицию /целевую скорость и целевое рысканье/угловую скорость по рысканью. Выбор устанавливаемых величин осуществляется с помощью поля `type_mask`.

`/mavros/setpoint_raw/attitude` — отправка пакета SET_ATTITUDE_TARGET. Позволяет установить целевую ориентацию / угловые скорости и уровень газа. Выбор устанавливаемых величин осуществляется с помощью поля `type_mask`

`/mavros/setpoint_raw/global` — отправка пакета SET_POSITION_TARGET_GLOBAL_INT. Позволяет установить целевую позицию в глобальных координатах (ширина, долгота, высота), а также скорости полета.

Дополнительные материалы

Этот раздел содержит статьи, не вошедшие в основные разделы, а так же статьи пользователей по той или иной проблематике, связанной с БПЛА.

Чтобы узнать больше о публикации текста в этом разделе, смотрите статью "Вклад в Клевер".

COEX Pix

Полетный контроллер **COEX Pix** является модифицированным аналогом полетного контроллера Pixracer. Этот полетный контроллер поставляется с наборами **Клевер 4** и далее.

> Исходные файлы полетного контроллера COEX Pix выложены в открытый доступ под лицензией CC BY-NC-SA.

Ревизия 1.1

Характеристики

- Размеры платы – 35x35 мм.
- Диаметр монтажных отверстий – 3.2 мм.
- Расстояние между центрами монтажных отверстий – 30.5 мм.
- Масса платы (без проводов) – 9 г.
- Диапазон рабочих температур – -5...+65 ºC.
- Диапазон входного напряжения – 4.8...5.5 В.

Основные элементы

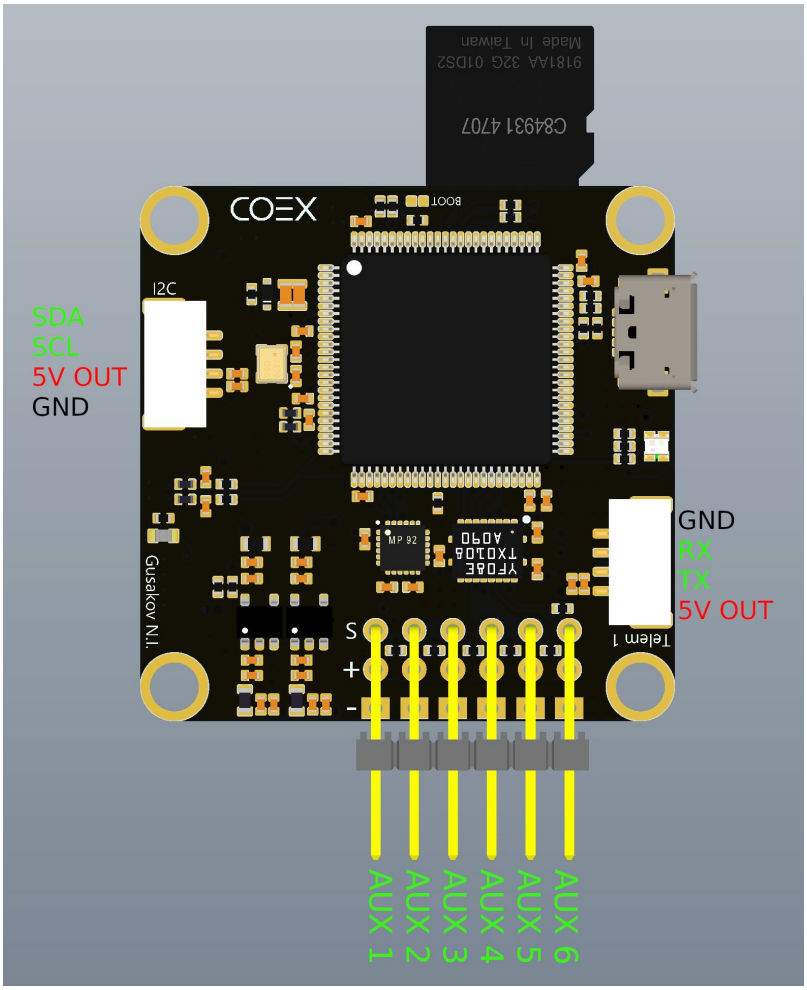
- Основной SOC – *STM32F427VIT6*.
- Память FRAM – *FM25V02A*.
- Датчики:
  - *MPU9250* (9 DOF) – 3-х осевой магнитометр, 3-х осевой гироскоп, 3-х осевой акселерометр.
  - *MS5607* – датчик атмосферного давления.

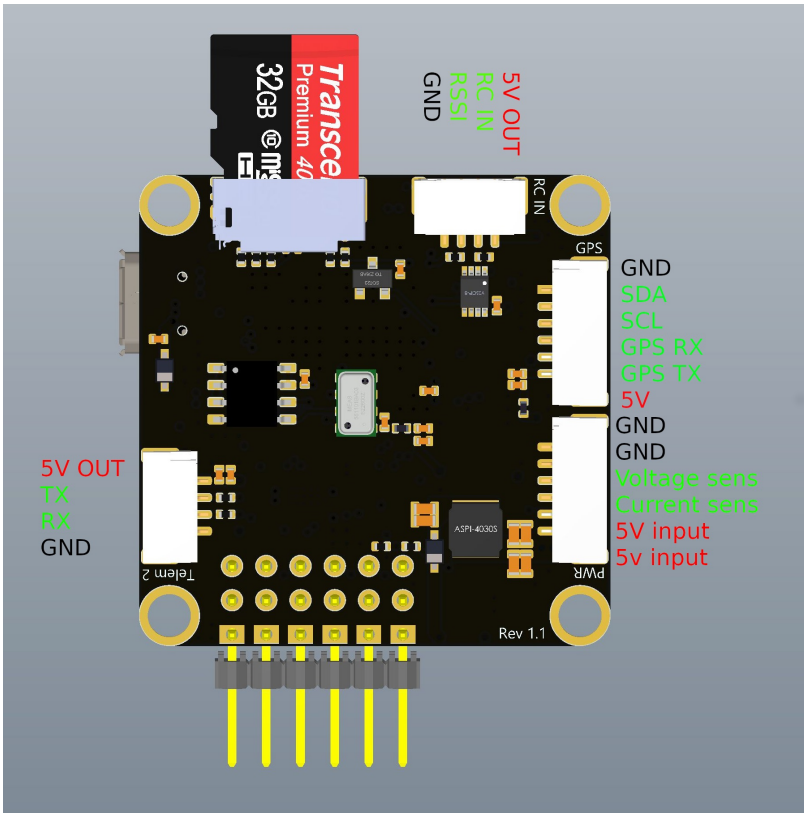
Разъемы

- *TELEM 1* (JST-GH 4 pin) – разъем для подключения телеметрии, порт 1, протокол UART.
- *TELEM 2* (JST-GH 4 pin) – разъем для подключения телеметрии, порт 2, протокол UART.
- *GPS* (JST-GH 6 pin) – разъем для подключения ГПС модуля (UART) с компасом (I2C).
- *I2C* (JST-GH 4 pin) – разъем для подключения поддерживаемых I2C устройств.
- *PWR* (JST-GH 6 pin) – разъем для подключения питания с платы COEX PDB или аналогичной, датчиков напряжения и тока.
- *RC IN* (JST-GH 4 pin) – разъем для подключения радиоприемника аппаратуры радиоуправления, канала для * снятия показаний RSSI. Поддерживаемые RC протоколы – PPM и SBUS.
- Разъем Micro USB – для подключения к ПК для настройки и коммуникации по протоколу USB 2.0/1.1

- Слот для карты памяти MicroSD, до 32 ГБ.
- Серворазъемы – для подключения контроллеров моторов и других устройств.

Схемы расположения контактов





На плате ревизии 1.0 RC IN разъем располагался на месте разъема Micro SD. Распиновка самого разъема осталась такой же.

Установка на Клевере

Важно: плата спроектирована для удобной установки на Клевере с поворотом на 180º по крену и 90º по рысканью (стрелка на плате находится снизу и указывает направо). Таким образом, параметр ориентации автопилота PX4 устанавливается в значение `SENS_BOARD_ROT` = `ROLL 180, YAW 90`.

Рекомендации

Во время установки полетного контроллера, учитывайте возможные влияния магнитных полей от силовых проводов и платы распределения питания на магнитометр. В случае установки данной платы над платой распределения питания, рекомендуется (в случае использования внутреннего магнитометра) поднять плату на высоту не менее 15 мм от платы распределения питания и силовых проводов. Силовой провод от аккумуляторной батареи старайтесь зафиксировать соответственно.

В случае использования внешнего GPS модуля со встроенным магнитометром, внутренний магнитометр рекомендуется отключать.

В случае, если на дроне не предусмотрен защитный кожух, рекомендуется изолировать барометр поролоновой губкой (достаточно подложить губку между полетным контроллером и нижней частью корпуса (платой

распределения питания), либо зафиксировать иным способом.

При подключении питания в разъем PWR, на разъеме **+** будет напряжение 5 вольт, его можно использовать для питания сервомашинок. Не рекомендуется подключать дополнительные источники питания в разъем **+**, если питание подается в разъем PWR. Питать полетный контроллер одновременно от USB и PWR\AUX разъемов допускается.

Особенности платы

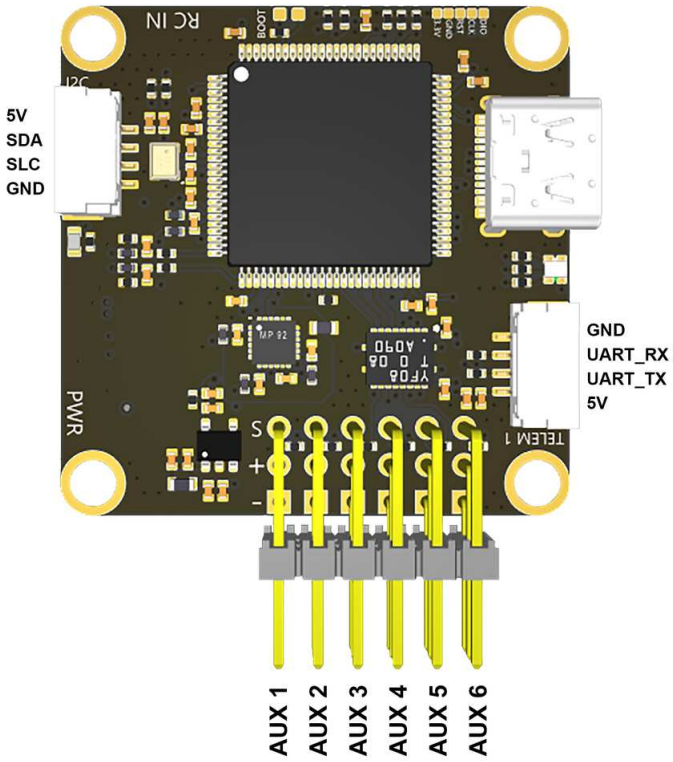
Для повышения надежности и стабильности, плата оснащена низкошумящими понижающими преобразователями. Установлен входной LC фильтр, а также ферритовые фильтры в цепях питания.

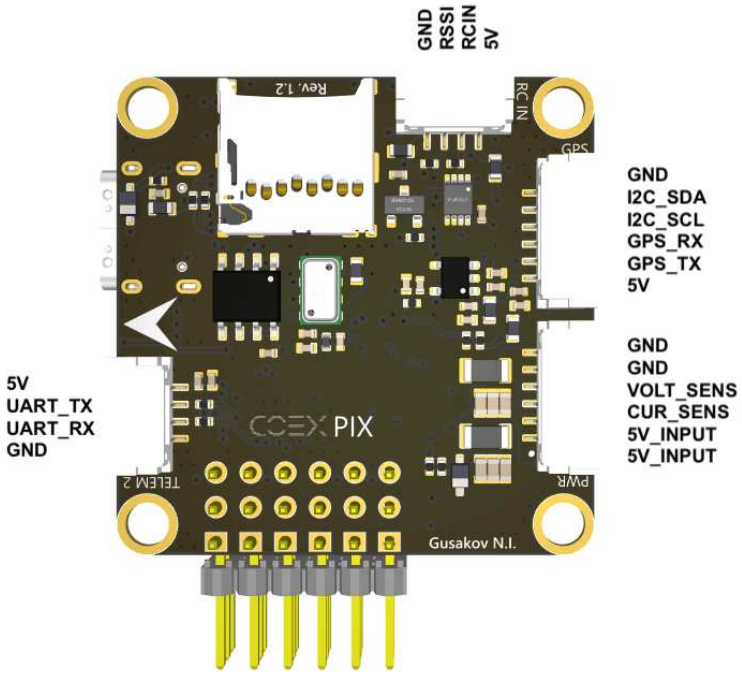
Ревизия 1.2

Нововведения

- Заменен разъем USB Micro-B на разъем USB Type-C.
- Изменен слот для MicroSD карт, на более глубокий.
- Изменены назначения пинов на разъеме I2C.
- Добавлены ферритовые фильтры в цепи питания.

Схемы расположения контактов




GND
RSSI
RCIN
5V
Rev. 1.2
RC IN
GPS
GND
I2C_SDA
I2C_SCL
GPS_RX
GPS_TX
5V
GND
GND
VOLT_SENS
CUR_SENS
5V_INPUT
5V_INPUT
5V
UART_TX
UART_RX
GND
COEX PIX
TELEM 2
PWR
Gusakov N.I.

COEX PDB

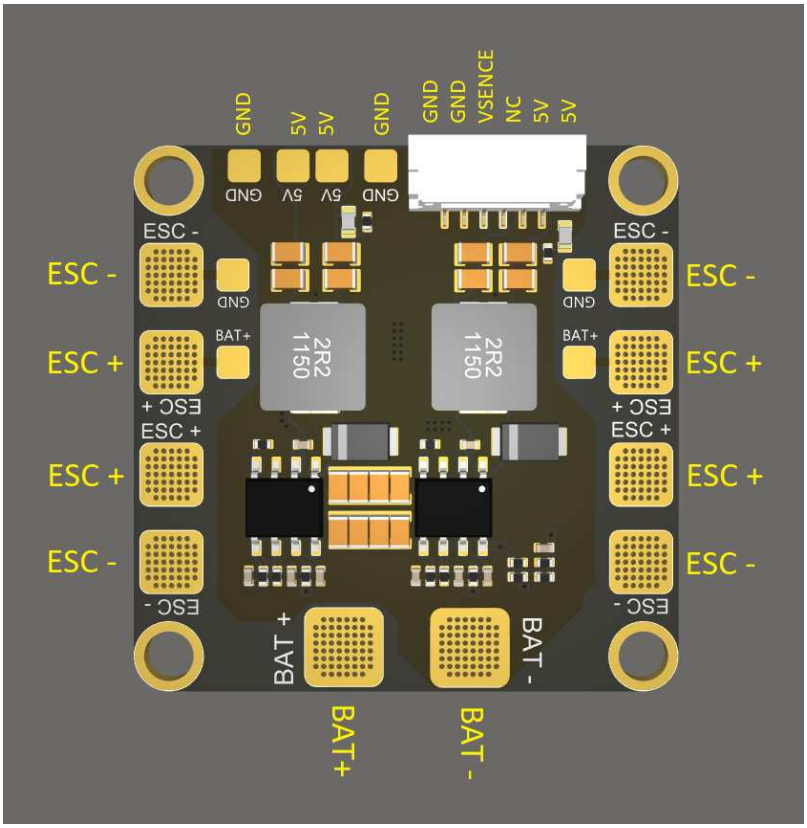
COEX PDB (Power Distribution Board) – плата распределения питания для квадрокоптеров Клевер 4.

Габаритные размеры платы: 35x35 мм.

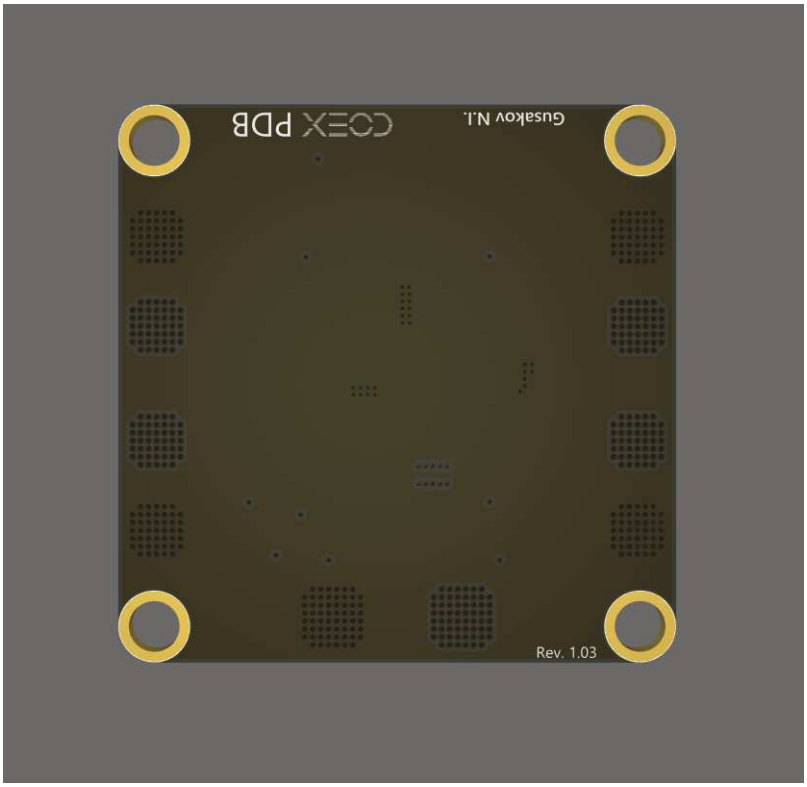
> Исходные файлы платы COEX PDB выложены в открытый доступ под лицензией CC BY-NC-SA.

Схемы расположения контактов

Вид сверху



Вид снизу


Gusakov N.I.
Rev. 1.03

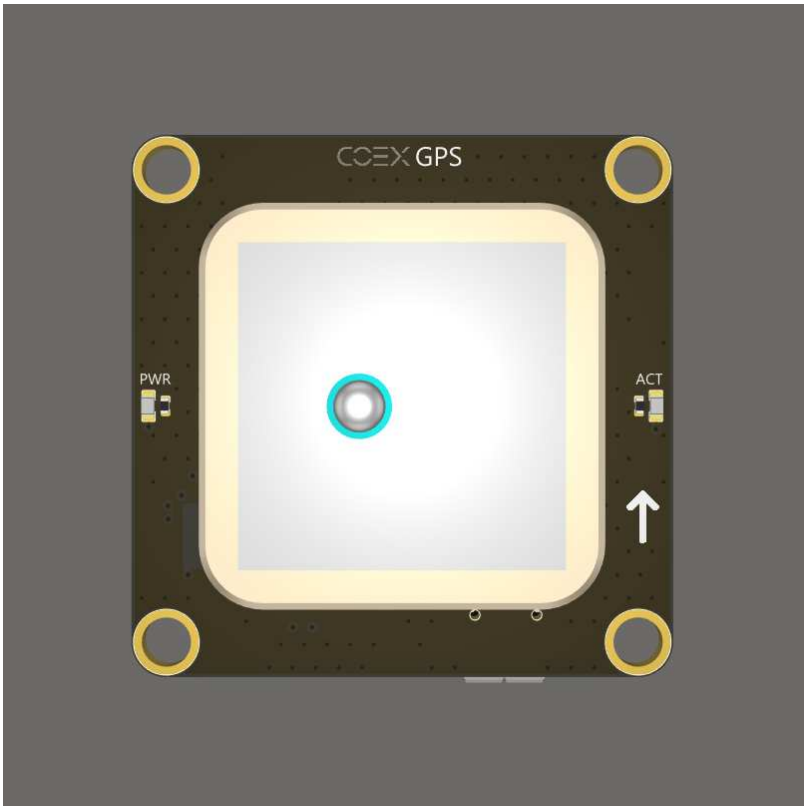
COEX GPS

ГНСС-приемник **COEX GPS** совместим с полетным контроллером COEX Pix. Этот приемник поставляется с наборами COEX Клевер 4 Pro.

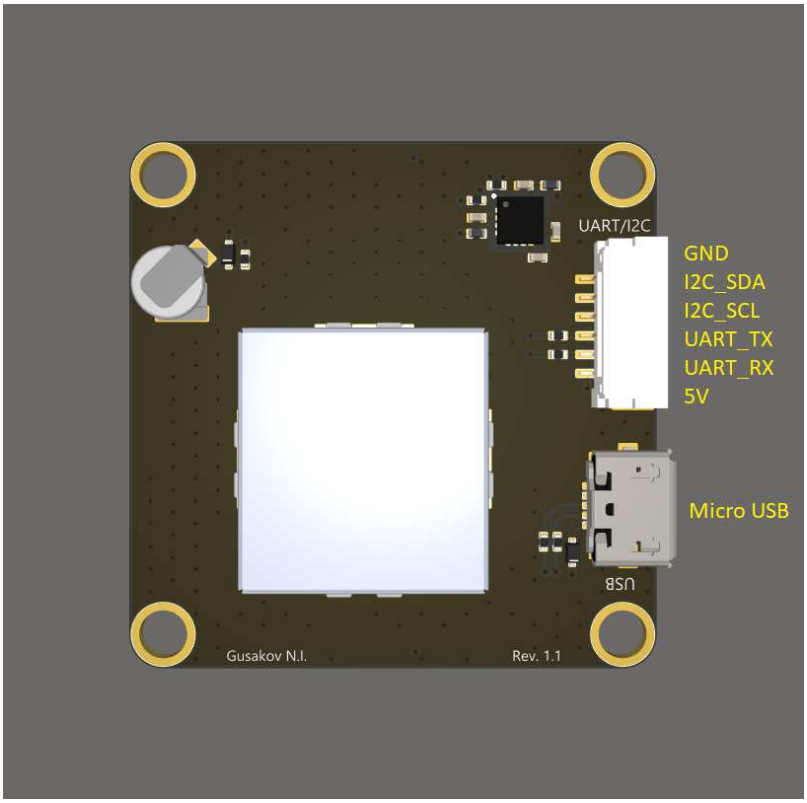
Исходные файлы платы COEX GPS выложены в открытый доступ под лицензией CC BY-NC-SA.

Схемы расположения контактов

Вид сверху



Вид снизу


UART/I2C
GND
I2C_SDA
I2C_SCL
UART_TX
UART_RX
5V
Micro USB
USB
Gusakov N.I.
Rev. 1.1

Подключение к Raspberry Pi с использованием SSH-ключей

Эта инструкция позволит вам быстро подключаться к Raspberry Pi. Всего за одну секунду. Без ввода пароля.

Базовые сведения по работе с SSH вы можете найти в разделе Доступ по SSH к Raspberry Pi. А в этом разделе вы найдёте расширенную информацию по использованию SSH, а также ряд рекомендаций по использованию SSH при работе с Клевером.

Общая информация

SSH (англ. *secure shell* — "безопасная оболочка") - сетевой протокол, позволяющий удалённо управлять операционной системой на компьютере, к которому вы подключились. Аналогичен такому протоколу, как *telnet*, но позволяет выполнять шифрование сетевого трафика по время взаимодействия. Таким образом передача паролей и другой секретной информации оказываются скрыты. Операционная система Raspberry Pi поддерживает взаимодействие по SSH, как и многие другие распространённые системы на базе Linux.

SSH позволяет не только организовывать работу в командной оболочке, но и передавать файлы, а также туннелировать передаваемые данные других протоколов, например информацию с видеокамеры или телеметрию. Кроме того, SSH поддерживает несколько режимов аутентификации (то есть проверки подключающегося пользователя), с его помощью возможно подключение к Клеверу не только с использованием пароля, но и беспарольный доступ (аутентификация по ключевой паре, т.е. SSH-ключи).

Аутентификация по паролю

Аутентификация по паролю на образе RPi для Клевера включена и пароль может быть использован для входа в командную оболочку мини-компьютера. На ЭВМ с операционными системами Linux (и в первую очередь на серверах, подключенных к интернету) возможность входа по паролю обычно отключают, поскольку есть более безопасный способ аутентификации.

> Вход в Клевер по паролю отключать не рекомендуется, поскольку можно совсем утратить доступ к командной оболочке по сети.

При первом подключении к RPi пользователю показывается уведомление с предложением сохранить уникальный отпечаток *fingerprint*. Сохранённая информация накапливается на компьютерах с которых выполняется вход по SSH на RPi, и проверяется на внезапную подмену.

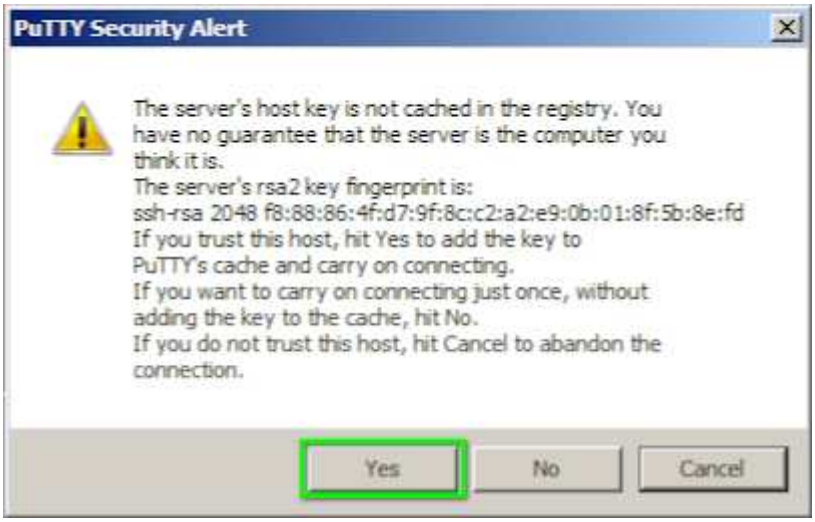
В ОС Linux и Unix (Mac OS) в текстовом SSH-клиенте первое подключение к RPi выглядит таким образом:

```
ssh pi@192.168.11.1
# The authenticity of host '192.168.11.1 (192.168.11.1)' can't be established.
# ED25519 key fingerprint is SHA256:4w/7MqTgrtsqPwKnVAMISpouaOJNqzUew2NkJjldMWI
# This key is not known by any other names
# Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
# Warning: Permanently added '192.168.11.1' (ED25519) to the list of known host
# pi@192.168.11.1's password: *********
# Linux clover-3270 5.10.17-v7l+ #1414 SMP Fri Apr 30 13:20:47 BST 2021 armv7l

whoami
# pi

exit
```

В графических программах в Windows у вас будут периодически возникать окошки с похожими предупреждениями.



В Windows 10 появился встроенный SSH-клиент, который можно запускать из командной строки, см. руководство по использованию от Microsoft по этой ссылке.

Аутентификация с использованием SSH-ключей

SSH-ключи - это удобный, быстрый альтернативный способ подключения к Raspberry Pi, для которого не требуется ввод пароля. В частности, при эксплуатации Клевера такой способ удобен потому, что экономит время, а значит и заряд аккумулятора, и лимит времени отведённого на мероприятия в полётных зонах. Кроме того, использование SSH-ключей открывает возможности по удобному использованию других программ, которыми бы вы вряд-ли воспользовались бы при необходимости всякий раз набирать пароль.

SSH-ключ делится на две части: пара состоит из т.н. *закрытого* и *открытого* ключа. Ключ однократно генерируется. Одна часть ключа (открытая) однократно передаётся на удалённый компьютер к которому будет выполняться подключение, вторая часть ключа (закрытая) хранится на компьютере, который будет подключаться, закрытая часть ключа никуда не передаётся.

> Открытый ключ однократно копируется на Raspberry Pi, а закрытый ключ сохраняется в ноутбуке в виде файла.

Подготовка

Для того, чтобы пара ключей появилась, её необходимо сгенерировать. В ОС Linux и Unix (Mac OS) есть программа `ssh-keygen` с помощью которой мы и получим нужную нам пару ключей (**внимание!** команды выполняются не в Raspberry Pi, и не в виртуальной машине симулятора Gazebo, а в командной оболочке ноутбука с которого вы будете подключаться к Клеверу):

Прежде чем пользоваться ключами, необходимо выполнить ряд действий для настройки прав доступа *на ноутбуке*:

```
# однократная настойка прав доступа к пользовательским директориям
chmod o-rwx $HOME
mkdir ~/.ssh
chmod g-rwx,o-rwx ~/.ssh
touch ~/.ssh/config ~/.ssh/known_hosts
chmod 600 ~/.ssh/config ~/.ssh/known_hosts
```

> Директория `.ssh` в домашней папке пользователя - это стандартное место хранения и ключевых пар, и настроек подключения с использованием SSH, поэтому доступ к ней запрещаем группе Others (*посторонние*). Современные дистрибутивы Linux проверяют права доступа к файлам в директории `.ssh` и могут отказать в аутентификации по ключевым парам.

Генерация пары SSH-ключей

Генерируем пару SSH-ключей в директории `~/.ssh` на ноутбуке:

```
ssh-keygen -f ~/.ssh/id_clover -C "SSH key for Clover" -N ""
# Your identification has been saved in /home/galina/.ssh/id_clover
# Your public key has been saved in /home/galina/.ssh/id_clover.pub

chmod 400 ~/.ssh/id_clover*
```

Копирование SSH-ключа на Raspberry Pi

После чего подключаемся к Raspberry Pi по Wi-Fi и продолжаем вводить команды *на ноутбуке* для копирования ключа на мини-компьютер:

```
ssh-copy-id -i ~/.ssh/id_clover.pub pi@192.168.11.1
# pi@192.168.11.1's password: *********
```

В результате с ноутбука на микрокомпьютер RPi будет скопирована т.н. *открытая* часть ключа, а *закрытая* останется на ноутбуке. Для проверки подключения без ввода пароля используем команду с указанием пути где находится SSH-ключ:

```
ssh -i ~/.ssh/id_clover pi@192.168.11.1
```

Если терминал не потребует ввода пароля для подключения к RPi, то вы всё сделали правильно и пара SSH-ключей работает. Теперь можно набрать команду выхода из SSH-терминала, чтобы продолжить настройку ноутбука:

```
pi@clover-3270:~ $ exit
# logout
# Connection to 192.168.11.1 closed.

galina@Thinkpad-X1:~/.ssh$
```

Настройка SSH-подключения к Клеверу

Теперь давайте настроим SSH-терминал таким образом, чтобы не приходилось всякий раз вписывать путь к закрытому ключу. Это делается с помощью редактирования файла `~/.ssh/config` *на ноутбуке*. Откройте файл в текстовом редакторе и добавьте в файл следующие строки (если там уже есть какая-то информация, то поместите их в конец файла):

```
Host 192.168.11.1
    User pi
    IdentityFile ~/.ssh/id_clover
    PreferredAuthentications publickey,password
    PubkeyAuthentication yes
    PasswordAuthentication yes
    ConnectTimeout 1
    TCPKeepAlive yes
    ServerAliveInterval 2
    ServerAliveCountMax 3
    StrictHostKeyChecking no
```

Эта настройка:

- влияет на работу SSH-терминала при подключении к компьютеру с ip-адресом `192.168.11.1`;
- если имя пользователя не указано, то автоматически будет использоваться имя `pi`;
- будет автоматически использоваться приватный ключ `~/.ssh/id_clover`;
- если ключ по каким-то причинам не подойдёт (был заменён на одном ноутбуке, но забыт заменить на другом), то SSH-терминал перейдёт к аутентификации по паролю (настройки `PreferredAuthentications`, `PubkeyAuthentication`, `PasswordAuthentication`);

- если связь с RPi не может установиться (WiFi ещё не включился), то SSH-подключение не зависнет, а быстро завершится (настройка `ConnectTimeout`);
- если связь с RPi будет внезапно разорвана, то SSH-подключение не зависнет, а быстро завершится (настройки `TCPKeepAlive`, `ServerAliveInterval`, `ServerAliveCountMax`);
- уникальные SSH-отпечатки RPi-микрокомпьютеров (*fingerprints*) о которых упоминалось выше, проверяться больше не будут (настройка `StrictHostKeyChecking`).

Таким образом будет решено множество неудобств, связанных с использованием SSH-подключений.

> Если у вас в лаборатории несколько дронов на базе Raspberry Pi, и несколько ноутбуков, то можно **однократно** сгенерировать SSH-ключи, скопировать их на все дроны и разложить по всем ноутбукам, тогда с любого ноутбука можно будет быстро зайти на любой из дронов.

Теперь, чтобы подключиться к RPi из терминала Linux вам достаточно набрать `ssh 1[TAB][TAB][ENTER]` и ip-адрес `192.168.11.1` автоматически дополнится в командной строке, т.к. командная оболочка использует адреса из файла `~/.ssh/config` и способна "угадать" ваши намерения для подключения к Клеверу. Нажав ввод вы мгновенно окажетесь в терминале RPi.

> Графические программы для Windows, которые поддерживают работу с SSH-ключами, которыми вы можете воспользоваться: PuTTY и MobaXterm.

Копирование файлов при помощи SSH

Чтобы скопировать файл `circle_flight.py` с ноутбука на RPi в домашнюю папку пользователя `pi` можно также воспользоваться SSH. Для этого наберите в командной оболочке команду:

```
# сначала указываем 'что' копируем, а потом 'куда'
scp circle_flight.py 192.168.11.1
```

Для того, чтобы скопировать файл `output.avi` с RPi из папки `examples` на ноутбук используем похожую команду:

```
# после символа ':' (двоеточие) можно указать путь на удалённом компьютере
# путь указанный как './' означает текущую папку, куда будет скопирован файл
scp 192.168.11.1:examples/output.avi ./
```

Удалённый запуск команд по SSH

Чтобы запустить команду с ноутбука на RPi (то есть удалённо) можно также воспользоваться SSH.

Команда выключения Raspberry:

```
ssh 192.168.11.1 'sudo shutdown now'
```

Пример команды запуска Python-скрипта:

```
ssh -t 192.168.11.1 'ROS_HOSTNAME=`hostname`.local && . /opt/ros/noetic/setup.b
```

Для того, чтобы удалённо запустить запись видео можно выполнить команду:

```
ssh -t 192.168.11.1 'ROS_HOSTNAME=`hostname`.local && . /opt/ros/noetic/setup.b
```

Использование радиотелеметрии

Радиотелеметрийный приемник позволяет выполнять подключение к полетному контроллеру без использования USB-кабеля или Wi-Fi. Это особенно полезно при полетах с GPS в режиме автономных миссий, так как дальность действия радиотелеметрии намного выше, чем у Wi-Fi.

Подключение радиотелеметрии

Комплект радиотелеметрии состоит из двух модулей, один из которых подключается к компьютеру, а другой – непосредственно к полетному контроллеру.



> Перед использованием радиотелеметрии убедитесь, что на обоих модулях установлены антенны. Использование модуля без антенны может привести к выходу его из строя.

Подключите один из модулей к разъему "Telem 1" полетного контроллера при помощи кабеля, который поставляется вместе с полетным контроллером.



Второй модуль подключите к компьютеру при помощи кабеля Micro-USB. Непрерывно горящий зеленый светодиод означает что связь между модулям установлена.

Моргающий красный светодиод означает, что между модулями передаются данные.

Откройте QGroundControl. Если всё было подключено верно, связь с дроном должна установиться без каких-либо дополнительных действий.

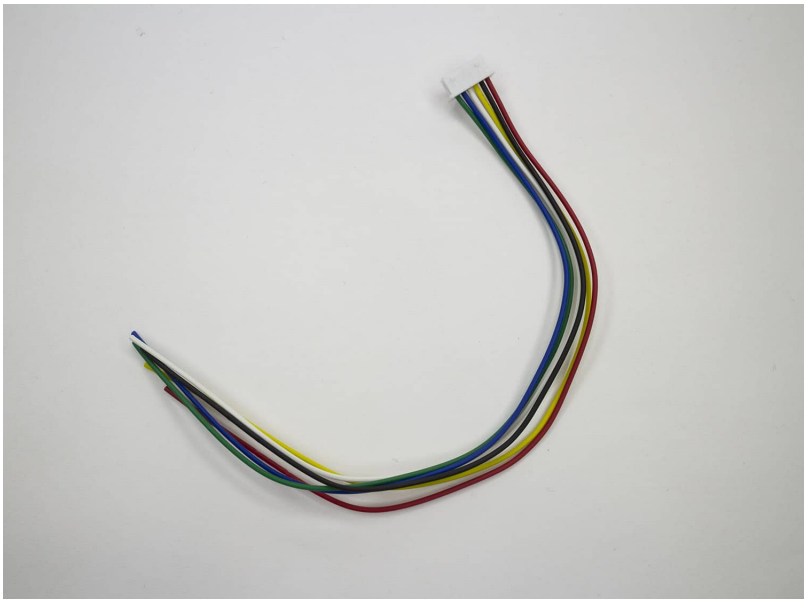
Камера Hawk Eye

Камера Hawk Eye позволяет записывать видео в разрешении до 4K 30FPS, а также имеет аналоговый выход, что позволяет использовать её вместе с FPV-видеопередатчиком.



Подключение

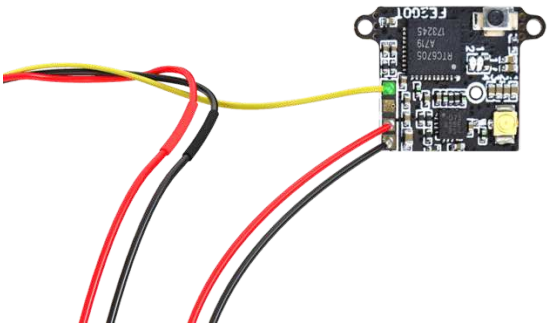
Установите плату на нейлоновые стойки сверху полетного контроллера. Для подключения камеры используйте кабель, идущий в комплекте.



Красный провод необходимо припаять к контакту BAT+ на плате распределения питания (от напряжения 5 воль камера работать не будет), черный провод необходимо припаять к контакту GND.

Для настройки камеры вам потребуется видеопередатчик и FPV шлем или очки, однако производить запись полетов можно и без видеопередатчика.

Для настройки камеры и полетов в FPV подключите желтый провод кабеля камеры к видеопередатчику, как показано на изображении:

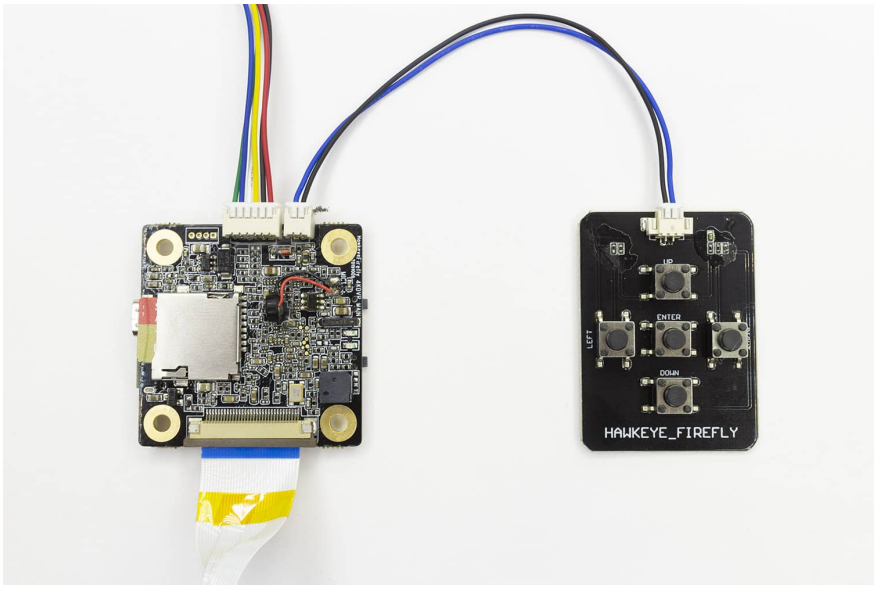


Белый, синий и зеленый провода в данном случае не используются и их можно не подключать.

> Внимание! Необходимо соблюдать аккуратность при подключении питания. Камера работает от напряжения 7–25 вольт, а видеопередатчик – от напряжения 5 вольт. Неправильное подключение питания может привести к выходу из строя камеры или видеопередатчика.

Настройка

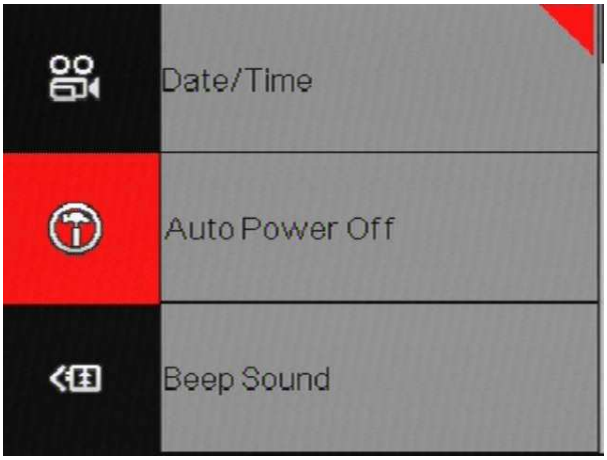
Подключите пульт для настройки к камере.



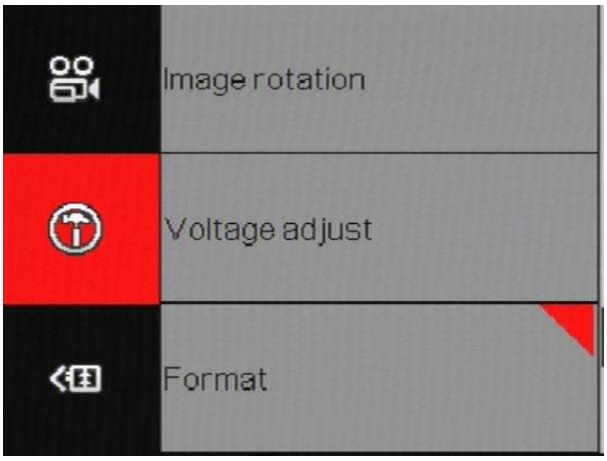
Установите SD-карту в разъем. Перед первым использованием её необходимо отформатировать. Для этого нажмите кнопку Left на пульте, вы попадете в меню настроек.



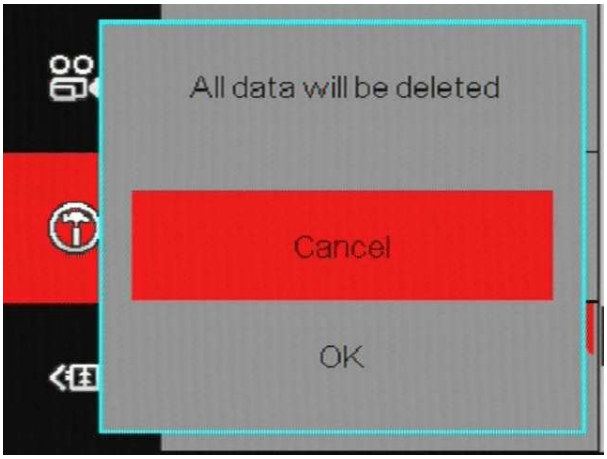
Нажмите кнопку left ещё раз.



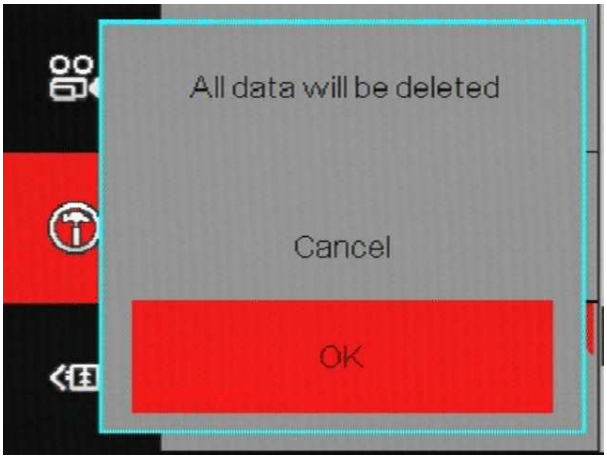
Нажимая на кнопку down, пролистайте список до пункта *Format*.



Нажмите enter. Появится предупреждение о том, что все данные с карты будут удалены.



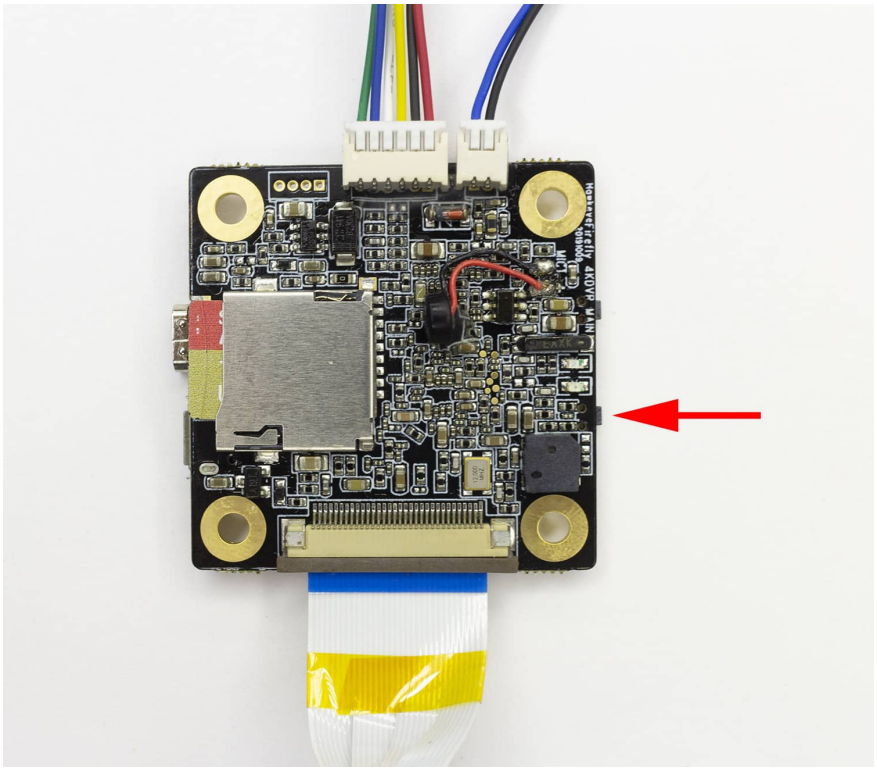
Нажмите на кнопку *OK*.



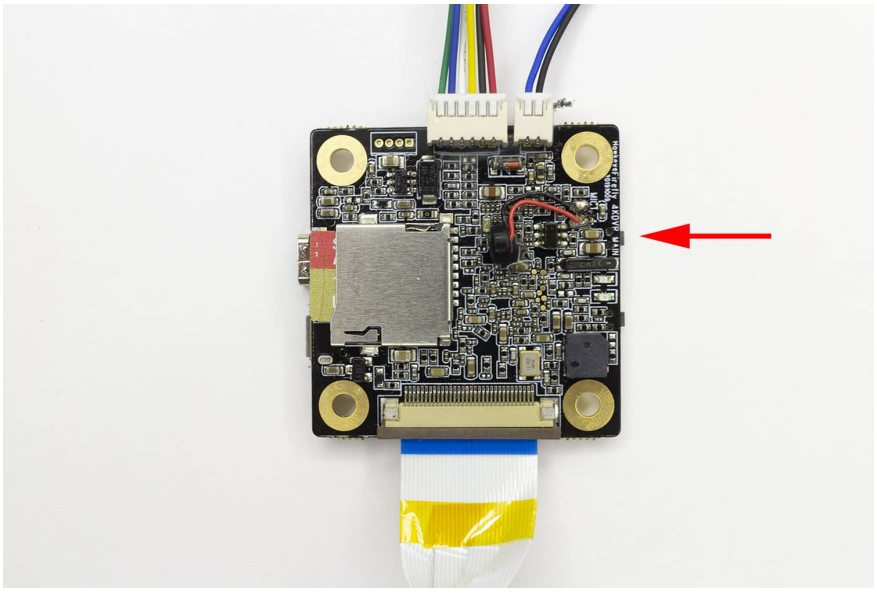
Карта памяти отформатирована и готова к записи. В этом же меню можно найти настройки разрешения, частоты кадров и другие.

Использование

Для начала записи нажмите кнопку на плате камеры. Камера сообщит о начале записи звуковым сигналом, а также мигающей красной точкой в верхней части экрана. Для прекращения записи повторно нажмите на кнопку записи.



Для смены разрешения видео без использования видеопередатчика или пульта нажмите кнопку смены разрешения на две секунды.



Пошаговая инструкция по настройке автономного полета Клевера 4

> Документация для версий образа, начиная с **0.20**. Для более ранних версий см. документацию для версии **0.19**.

Данная инструкция содержит ссылки на другие статьи, в которых каждая из затронутых тем разобрана более подробно. Если вы столкнулись с трудностями во время прочтения одной из таких статей, рекомендуется вернуться к данной инструкции, так как здесь многие операции описаны пошагово, а также отсутствуют ненужные шаги.

Первоначальная настройка Raspberry Pi

- Установите Raspberry Pi и камеру на квадрокоптер по инструкции.
- Скачайте образ системы по ссылке.
- Запишите образ на MicroSD карту.
- Вставьте карту в Raspberry Pi.
- Подключите питание к Raspberry Pi и ожидайте появления Wi-Fi-сети. Для этого подключите Raspberry Pi к компьютеру через MicroUSB-кабель. На Raspberry Pi должен периодически мигать зеленый светодиод. Он сигнализирует о нормальной работе операционной системы.

  > Перед подключением Raspberry Pi к компьютеру по USB необходимо вытащить из Raspberry Pi провод питания (который идет от BEC). Иначе могут быть проблемы с питанием.

- Подключитесь к Wi-Fi и зайдите в веб-интерфейс (статья).

  Во время первого включения сеть появляется не сразу. Нужно дождаться полной загрузки системы. Если в списке сетей долго не появляется сети Клевера, закройте окно с выбором сети и откройте снова. Тогда список сетей обновится.

  > Если на этом шаге вы подключились к Wi-Fi-сети коптера, рекомендуется открыть локальную версию этой статьи, иначе ссылки не будут работать.

- Подключитесь к Raspberry Pi по SSH.

  Самый быстрый способ – веб-доступ. Следуйте инструкциям в статье "Доступ по SSH".

- Если необходимо, можно поменять название и пароль сети. См. статью "Настройка сети". Остальные операции с сетью производить не нужно.
- Для редактирования файлов пользуйтесь редактором nano. Инструкция по работе с редактором.

  > В редакторе перемещать курсор можно только стрелками на клавиатуре.

- Перезагрузите Raspberry Pi:

  ```
  sudo reboot
  ```

  Соединение временно закроется, создастся новая сеть. К ней надо подключиться заново.

- Убедитесь в корректной работе камеры. В браузере зайдите на адрес http://192.168.11.1:8080 и выберите `image_raw`.

  Более подробно можно прочитать в статье "Просмотр изображений с камер".

  Если изображение размыто, необходимо сфокусировать линзу. Для этого покрутите объектив в одну или в другую сторону. Продолжайте крутить, пока изображение не станет четким.

  > На камере должен гореть красный светодиод: он означает, что камера в данный момент производит съемку. Если светодиод не горит: либо камера подключена неправильно, либо операционная система не загрузилась, либо в настройках допущена ошибка.

Базовые команды

Вам пригодятся основные команды Linux, а также специальные команды Clover, чтобы уверенно работать в системе.

Показать список файлов:

```
ls
```

Перейти в папку с прописыванием пути к ней:

```
cd catkin_ws/src/clover/clover/launch/
```

Перейти в домашнюю директорию:

```
cd
```

Открыть файл file.py:

```
nano file.py
```

Открыть файл clover.launch с прописыванием полного пути к нему (сработает, если вы находитесь в другой папке):

```
nano ~/catkin_ws/src/clover/clover/launch/clover.launch
```

Сохранить файл (нажимать последовательно):

```
Ctrl+X; Y; Enter
```

Удалить файл или папку с названием name (ВНИМАНИЕ: операция выполнится без подтверждения. Будьте осторожны!):

```
rm -rf name
```

Создать папку с названием myfolder:

```
mkdir myfolder
```

Полная перезагрузка Raspberry Pi:

```
sudo reboot
```

Перезапуск только систем Клевера:

```
sudo systemctl restart clover
```

Выполнить самопроверку Клевера:

```
rosrun clover selfcheck.py
```

Остановить программу

```
Ctrl+C
```

Запустить программу myprogram.py на Питоне:

```
python3 myprogram.py
```

Журнал событий процессов Клевера. Пролистывать список можно нажатием Enter или сочетанием клавиш Ctrl+V (пролистывает быстрее):

```
journalctl -u clover
```

Открыть файл sudoers от имени администратора (он не откроется без прописывания sudo. Через sudo можно запускать другие команды, если они не открываются без прав администратора):

```
sudo nano /etc/sudoers
```

Настройка параметров Raspberry Pi для автономного полета

Большинство параметров, необходимых для полета, хранится в папке `~/catkin_ws/src/clover/clover/launch/`.

- Зайти в папку:

  ```
  cd ~/catkin_ws/src/clover/clover/launch/
  ```

  Символ `~` обозначает домашнюю директорию вашего пользователя. Если вы уже находитесь в ней, можно обойтись командой: `cd catkin_ws/src/clover/clover/launch/`

  > Клавишей Tab можно автоматически дополнить названия файлов, папок или команд. Нужно начать вводить желаемое название и нажать Tab. Если не будет конфликтов, название напишется полностью. Например, чтобы быстро ввести путь к папке с настройками, после ввода `cd` можно начать вводить следующую комбинацию клавиш: `c-Tab-s-Tab-c-Tab-c-Tab-l-Tab`. Таким образом можно сэкономить много времени при написании длинной команды, а также избежать возможных ошибок в написании пути.

- В этой папке необходимо сконфигурировать несколько файлов:
  - `clover.launch`
  - `aruco.launch`
  - `main_camera.launch`
- Открыть файл `clover.launch`:

  ```
  nano clover.launch
  ```

  Вы должны находиться в папке, в которой располагается файл. Если вы находитесь в другой папке, файл можно открыть, прописав полный путь к нему:

  ```
  nano ~/catkin_ws/src/clover/clover/launch/clover.launch
  ```

  Если файл одновременно редактируют два пользователя, а также если в прошлый раз закрытие файла произошло некорректно, программа nano не отобразит файл сразу, а попросит дополнительное разрешение. Для этого нужно нажать клавишу Y.

Если содержимое файла все равно пусто, возможно, вы неверно ввели имя файла. Нужно обращать внимание на расширение и вписывать его полностью. Если вы вписали неверное имя или расширение, программа nano создаст пустой файл с этим названием, что нежелательно. Такой файл следует удалить.

- В файле clover.launch найти строчку:

```
<arg name="aruco" default="false"/>
```

и заменить `false` на `true`:

```
<arg name="aruco" default="true"/>.
```

Это активирует модуль распознавания ArUco-маркеров.

- Откройте файл `aruco.launch`.
- В нем нужно активировать несколько параметров. Подробнее в статье.

Должно получиться:

```
<arg name="aruco_detect" default="true"/>
<arg name="aruco_map" default="true"/>
<arg name="aruco_vpe" default="true"/>`
```

- Сгенерируйте поле с метками. Смотрите подробности в статье Навигация по картам ArUco-маркеров. Для генерации меток нужно ввести команду с определенными значениями.

Пример команды для генерации поля, где:

  - длина маркера = 0.335 м (`length`)
  - 10 столбцов (x)
  - 10 строк (y)
  - расстояние между центрами меток по оси x = 1 м (`dist_x`)
  - расстояние между центрами меток по оси y = 1 м (`dist_y`)
  - номер первого маркера = 0 (`first`)
  - название карты остается стандартным: map.txt
  - нумерация идет с верхнего левого угла (ключ `--top-left`)

```
rosrun aruco_pose genmap.py 0.335 10 10 1 1 0 > ~/catkin_ws/src/clover/aru
```

В большинстве полей нумерация начинается с нулевой метки. Также в большинстве случаев нумерация начинается с верхнего левого угла, поэтому при генерации очень важно указывать ключ `--top-left`.

> Если вы зададите другое имя для файла с картой, его нужно прописать в файле `aruco.launch`. Найдите строку `<param name="map" value="$(find aruco_pose)/map/map.txt"/>` и замените название map.txt на название вашего файла.

- Отредактируйте файл `main_camera.launch` для настройки камеры:

Подробнее в статье "Настройка расположения основной камеры".

В этом файле необходимо отредактировать строку с параметрами расположения камеры. Строка выглядит так:

```
<node pkg="tf2_ros" type="static_transform_publisher" name="main_camera_fr
```

В файле вы найдете много строк, похожих на эту, но большинство из них закомментированы (то есть не читаются) и только одна раскомментирована. Это заранее заготовленные настройки, из которых можно выбрать нужную вам.

Комментарий в языке XML — это символы `<!--` в начале строки и `-->` в конце строки. Пример закомментированной строки:

```
<!--<node pkg="tf2_ros" type="static_transform_publisher" name="main_camer
```

Пример незакомментированной строки (строка будет учитываться программой):

```
<node pkg="tf2_ros" type="static_transform_publisher" name="main_camera_fr
```

Над этими строками написано, какому расположению камеры соответствует настройка. Если шлейф от камеры выходит вперед относительно коптера, а камера направлена вниз, нужно выбрать настройку:

```
<!-- camera is oriented downward, camera cable goes forward  [option 2] --
```

Чтобы выбрать нужную настройку, необходимо раскомментировать соответствующую строку, и закомментировать другую аналогичную строку, чтобы не возникло конфликтов.

- Сохранить изменения. Последовательно нажмите:

```
Ctrl+x; y; Enter
```

- Перезагрузите модуль Клевер:

```
sudo systemctl restart clover
```

Настройка полетного контроллера для автономного полета

- Перепрошить полетный контроллер модифицированной прошивкой. Скачать её можно здесь в разделе "Загрузка прошивки в полетный контроллер".
- Инструкция по прошивке и настройке полетного контроллера — в той же статье.

> Обязательно выберете файл скачанной прошивки после нажатия Firmware.

Соединение полетного контроллера и Raspberry Pi

- Соедините Raspberry Pi и Pixracer через MicroUSB-кабель. Кабель должен быть аккуратно плотно закручен и пропущен снизу коптера, чтобы не попасть в пропеллеры.
- Удаленно подключитесь к полётному контроллеру через QGroundControl. В системе Clover уже выставлены нужные настройки, остается лишь создать новое подключение в QGroundControl, выбрать его и подключиться. Настраивается оно, как на картинке в статье "Подключение QGroundControl по Wi-Fi".

Настройка пульта

- Настройка полетных режимов описана в статье "Полетные режимы".

  Канал 5 должен располагаться на переключателе SwC; Канал 6 - на SwA. Однако вы можете настроить эти каналы любым удобным для вас образом.

Выполнение автоматической проверки

Проверку следует выполнить, когда вы полностью настроили дрон, а также при возникновении неполадок. Подробно процедура описана в статье "Автоматическая проверка".

- Выполнить команду:

```
rosrun clover selfcheck.py
```

Написание программы

В статье "Автономный полет" описана работа с модулем `simple_offboard`, который создан для простого программирования дрона. В ней даны описания основных функций, а также примеры кода.

- Скопируйте из раздела "Использование из языка Python" пример кода и вставьте в редактор (например, в Visual Studio Code, PyCharm, Sublime Text, Notepad++).
- Сохраните документ с расширением .py для включения подсветки текста.
- Далее необходимо добавить полётные команды в программу. Примеры таких команд представлены в статье. Нужно написать функции для взлета и полета в точку, а также для посадки.
- Взлет.

  Для взлета можно использовать функцию `navigate`:

  ```
  navigate(x=0, y=0, z=1.5, speed=0.5, frame_id='body', auto_arm=True)
  ```

  Добавьте эту строку внизу программы.

  Также добавьте команду ожидания:

  ```
  rospy.sleep(3)
  ```

> Важно выделить время на выполнение команды `navigate`, иначе коптер, не дожидаясь выполнения предыдущей команды, сразу перейдет к выполнению следующей. Для этого используется команда `rospy.sleep()`. В скобках указывается время в секундах. Функция `rospy.sleep()` относится к предыдущей команде `navigate`, а не к последующей, то есть это время, которое мы даем на то, чтобы долететь до точки, обозначенной в предыдущем `navigate`.

- Зафиксировать положение дрона в системе координат маркерного поля.

  Для этого нужно выполнить `navigate` и указать в нем необходимые координаты (например, x=1, y=1, z=1.5) и выбрать систему координат (`frame_id`):

  ```
  navigate(x=1, y=1, z=1.5, speed=1, frame_id='aruco_map')
  ```

- В итоге должно получиться:

  ```
  navigate(x=0, y=0, z=1.5, speed=0.5, frame_id='body', auto_arm=True)
  rospy.sleep(3)
  navigate(x=1, y=1, z=1.5, speed=1, frame_id='aruco_map')
  ```

  > Обратите внимание, что параметр `auto_arm=True` ставится только при первом взлете. В остальных случаях его выставлять нельзя, иначе возникнут проблемы с перехватом управления.

- Если вы хотите добавить другие точки для пролета, нужно дописать еще один `navigate` и `rospy.sleep()`. Время нужно вычислить отдельно для каждой точки в зависимости от скорости полета и расстояния между

точками.

Например, если мы хотим полететь в точку (3, 3, 1.5):

```
navigate(x=3, y=3, z=1.5, speed=1, frame_id='aruco_map')
rospy.sleep(3)
```

> Координаты не должны выходить за пределы вашего поля. Если поле имеет размер 4x4 метра, максимальное значение координат, которое стоит указывать, — 4.

- После пролета по точкам нужно приземлиться. Следующая строка ставится в конце программы:

```
land()
```

Запись программы на дрон

Самый простой способ – это скопировать текст программы, создать новый файл в командной строке Клевера и вставить текст программы в файл.

- Для создания файла `myprogram.py` введите команду:

```
nano myprogram.py
```

Название можно выбрать любое, однако не рекомендуется использовать пробелы и специальные символы. Также расширение у программы всегда должно быть `.py`.

- Вставить текст в поле ввода. Если вы пользуетесь веб-доступом Butterfly на Windows или Linux:

```
Ctrl+Shift+V
```

На Mac нажмите `cmd+v`.

- Сохранить файл:

```
Ctrl+x; Y; Enter
```

Запуск программы

- Необходимо тщательно подготовить дрон, пульт и программу. Запустите `selfcheck.py`. Убедитесь, что дрон летает в ручном режиме.
- Включите дрон и дождитесь, пока загрузится система. Красный огонек на камере означает, что систем загрузилась.
- Проверьте полет в режиме POSCTL.

Для этого взлетите над метками в режиме STABILIZED и переведите переключатель SwC в нижнее положение - режим POSCTL.

> Будьте готовы сразу же переключиться обратно в режим STABILIZED в случае выхода дрона из-под контроля!

Установите левый стик (газ) в центральное положение. Дрон должен зависнуть на месте. В таком случае можно сажать дрон и переходить к следующему шагу. Если нет, нужно разобраться в проблеме.

- Установите переключатель SwC в центральное положение. С помощью него вы будете перехватывать дрон: стоит лишь переключить его в верхнее положение.
- Установите левый стик (газ) в центральное положение, чтобы в случае перехвата дрон не упал на пол.
- Запустите программу. Для этого выполните команду:

```
python3 my_program.py
```

> После выполнения программы дрон может некорректно приземлиться и продолжать лететь над полом. В таком случае нужно перехватить управление.

Имя хоста

> Документация для версий образа, начиная с **0.20**. Для более ранних версий см. документацию для версии **0.19**.

По умолчанию на Клевере установлено имя хоста (hostname) `clover-xxxx`, где `xxxx` – случайные цифры. Имя хоста соответствует SSID точки доступа Wi-Fi.

Таким образом, Клевер доступен на машинах, поддерживающих mDNS, под именем `clover-xxxx.local`. Вы можете использовать это имя для SSH-доступа на Клевер:

```
ssh pi@clover-xxxx.local
```

Также это имя может быть использовано вместо IP-адреса для открытия страницы Клевера в браузере и т. д.

Изменение имени хоста

В некоторых ситуациях необходимо изменение имени хоста Клевера. Для это используйте утилиту `hostnamectl`:

```
sudo hostnamectl set-hostname newname
```

Где `newname` – новое имя машины. Утилита `hostnamectl` поменяет имя в файле `/etc/hostname`.

Также необходимо прописывание нового имени в файл `/etc/hosts`:

```
127.0.1.1    newname newname.local
```

Прописывание `newname.local` необходимо, чтобы ROS смог разрешить это имя в ситуациях, когда все сетевые интерфейсы неактивны (отключение/разрыв связи Wi-Fi).

> Изменение имени хоста не повлечёт за собой изменений SSID точки доступа Wi-Fi (и наоборот, изменение SSID точки доступа не поменяет имя хоста).

Симуляция PX4

> Это статья описывает только установку PX4 и **является устаревшей**. Используйте официальную конфигурацию для Gazebo или образ виртуальной машины со всем необходимым для запуска симуляции Клевера.

Основная статья: https://dev.px4.io/en/simulation/

Симуляция PX4 возможна в ОС GNU/Linux и macOS с использованием систем симуляции физической среды jMAVSim и Gazebo.

jMAVSim является легковесной средой, предназначенной только для тестирование мультироторных летательных систем; Gazebo – универсальная среда для любых типов роботов.

Запуск PX4 SITL

1. Склонировать репозиторий с PX4.

```
git clone https://github.com/PX4/Firmware.git
cd Firmware
```

jMAVSim

Основная статья: https://dev.px4.io/en/simulation/jmavsim.html

Для симуляции с использованием легковесной среды jMAVSim используйте команду:

```
make posix_sitl_default jmavsim
```

Для использования модуля расчета позиции LPE вместо EKF2, используйте:

```
make posix_sitl_lpe jmavsim
```

Gazebo

Основная статья: https://dev.px4.io/en/simulation/gazebo.html

Для начала установите Gazebo 7. На Mac:

```
brew install gazebo7
```

На Linux (Debian):

```
sudo apt-get install gazebo7 libgazebo7-dev
```

Запустите симуляцию, находясь в папке Firmware:

```
make posix_sitl_default gazebo
```

Можно запустить симуляцию в headless режиме (без оконного клиента). Для этого используйте команду:

```
HEADLESS=1 make posix_sitl_default gazebo
```

Подключение

QGroundControl автоматически подключится к запущенной симуляции при запуске. Работа будет осуществляться также, как и с настоящим полетным контроллером.

Для подключение MAVROS к симуляции необходимо использовать протокол UDP, локальный IP-адрес и порт 14557, например:

```
roslaunch mavros px4.launch fcu_url:=udp://@127.0.0.1:14557
```

Запуск SITL своими руками на чистой Ubuntu

Настройка среды для запуска Gazebo

Для того, чтобы запустить симулятор полета дрона, Gazebo или jMAVSim вам потребуется сделать соответственные настройки вашей среды.

> Среда `ROS Melodic` изначально ориентированна для Ubuntu версии 18.04 (Bionic), поэтому актуальность данной инструкции гарантируется только для данной версии операционной системы.

В первую очередь вам потребуется установить полный пакет ROS Melodic desktop-full, инструкцию по установке вы можете найти в статье по установке ROS.

После того, как вы выполнили указанные выше инструкции, вам нужно проверить, есть ли в вашем пакете `ROS` все нужные пакеты.

```
sudo apt-get install ros-melodic-gazebo-ros \
                     ros-melodic-gazebo-dev \
                     ros-melodic-gazebo-plugins \
                     ros-melodic-gazebo-ros-pkgs \
                     ros-melodic-gazebo-msgs \
                     ros-melodic-geographic-msgs
```

Чтобы избежать ошибок во время запуска симулятора, вам нужно будет установить Gazebo v9.11, для этого подключите необходимый репозиторий и добавьте соответствующие ключи:

```
sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
```

Вам нужно установить пакеты которые потребуются во время запуска симуляции:

```
sudo apt-get update && sudo apt-get -y --quiet --no-install-recommends install
```

Для того, чтобы установить актуальные Python-модули, вам потребуется новая версия системы управления пакетами pip:

```
wget -qO- http://bootstrap.pypa.io/get-pip.py | sudo python
```

Теперь установите необходимые модули:

```
pip install --user setuptools pkgconfig wheel && pip install --user argparse a
```

Вам необходимо установить спецификацию для библиотеки `geographiclib`:

```
wget -qO- https://raw.githubusercontent.com/mavlink/mavros/master/mavros/script
```

Склонируйте себе папку содержащую программное обеспечение PX4 и верните ее к стабильной версии v1.8.2:

```
git clone https://github.com/PX4/Firmware.git
cd Firmware/
git checkout v1.8.2
```

Если вы все настройки были произведены правильно, вы можете произвести сборку пакета Gazebo, чтобы в дальнейшем быстрее его запустить. Для этого вы должны находиться в директории `Firmware`:

```
make posix_sitl_default sitl_gazebo
```

Теперь все готово к запуску самого симулятора, для этого пропишите в переменных окружения, где искать собранные библиотеки и запустите симулятор. Обратите внимание, что если вы хотите вызвать симулятор в другом окне терминала, вам повторно потребуется прописать переменные окружения (первая строка последующей команды):

```
. Tools/setup_gazebo.bash $(pwd) $(pwd)/build/posix_sitl_default
roslaunch gazebo_ros empty_world.launch world_name:=$(pwd)/Tools/sitl_gazebo/wo
```

Запуск PX4 для Gazebo

> Для того, чтобы открыть окно PX4 параллельно с симулятором, откройте дополнительное окно терминала.

Чтобы запустить PX4 и подключить его к Gazebo, в директории `Firmware` соберите сам пакет симулятора:

```
make posix_sitl_default
```

Теперь при запущенном симуляторе, вы можете вызвать окно `PX4`. Для этого в той же директории вызовите команду:

```
./build/posix_sitl_default/px4 . posix-configs/SITL/init/ekf2/iris
```

После загрузки консоли, вы можете проверить то что соединение с симулятором установлено, вызвав команду `commander takeoff` для взлета и `commander land` для посадки.

Сборка образа Клевера в симуляторе

Для того, чтобы пользоваться командами, предоставляемыми образом Клевера, вам потребуется его скачать и настроить. Создайте директорию, в которой вы будете собирать образ, перейдите в созданную директорию и воспользуйтесь системой сборки, предоставляемой ROS, для инициализации рабочей среды.

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin_make
```

Подтяните зависимости, прописанные в файле `setup` и склонируйте образ `Clover` в директорию `src`:

```
./devel/setup.bash
cd src
git clone https://github.com/copterexpress/clover
```

Перейдите в корневую папку и обновите зависимости ROS:

```
cd ..
rosdep install -y --from-paths src --ignore-src -r
```

Повторите сборку среды, но теперь с добавленным пакетом `Clover`:

```
catkin_make
```

Если сборка прошла успешно то вы можете запустить ноду Клевера и пользоваться пакетом `clover` точно так же, как и на реальном коптере:

```
. devel/setup.bash
roslaunch clover_simulation simulator.launch type:=none
```

Для того, чтобы воспользоваться функциями предоставляемыми нашим пакетом, в новом окне терминала подтяните зависимости из файла `setup`:

```
source ~/catkin_ws/devel/setup.bash
```

Теперь вы можете воспользоваться всеми возможностями пакета `clover` в вашем симуляторе.

Настройка PID регуляторов

Основная статья:
https://docs.px4.io/master/en/config_mc/pid_tuning_guide_multicopter.html.

В этой статье описаны методы и основные технологии настройки каскадного ПИД-регулятора. Приведенные советы и методики подходят для любых видов рам (Квадрокоптеров, Гексакоптеров, Октокоптеров и т.д.).

Усредненные рекомендованные настройки для комплектов Клевер приведены в статье "Первоначальная настройка".

> Эта статья только для продвинутых пользователей. Плохая или неполная настройка ПИД регуляторов может привести к сильным поломкам вашего дрона.

Введение

В PX4 используются Пропорционально-Интегрально-Дифференцирующий (ПИД) регулятор, являющийся одним из самых распространенных методов управления.

Регулятор реализованный в PX4 является каскадным. Это подразумевает, что он состоит из нескольких элементов, которые по очереди (каскадно) передают свои значения элементам находящимся ниже. Соответственно регуляторы более высокого уровня передает свой результат регулятору более низкого уровня.

Регулятор самого низкого уровня, это регулятор угловых скоростей, далее регулятор положения и последние, регуляторы скорости и позиции. Настройка ПИД-регулятора должна выполняться в том же порядке, начиная с регулятора угловой скорости, поскольку это повлияет на все остальные уровни.

Шаги настройки

В случае, если у вас нет опыта настройки ПИД регуляторов, придерживайтесь представленных ниже особенностей настройки.

- Все изменения значений регулятора должны быть постепенными, в противном случае могут появится сильные осцилляции и вы совершенно потеряете контроль над аппаратом. Увеличивайте значения регулятора не более чем на 25-30% от настоящего значения, уменьшайте на 5-10% до окончательной настройки.
- Сажайте аппарат перед следующей итерацией смены коэффициентов. Перед следующим взлетом очень медленно увеличивайте стик газа для проверки аппарата на сильные осцилляции.

Регулятор угловых скоростей

Регулятор угловых скоростей является самым низкоуровневым в каскаде, он состоит из трех независимых ПИД-регуляторов, управляющих угловыми скоростями коптера по трем осям (крен, тангаж, рысканье).

- Регулятор угловых скоростей по крену (`MC_ROLLRATE_P` , `MC_ROLLRATE_I` , `MC_ROLLRATE_D`)
- Регулятор угловых скоростей по тангажу (`MC_PITCHRATE_P` , `MC_PITCHRATE_I` , `MC_PITCHRATE_D`)
- Регулятор угловых скоростей по рысканью (`MC_YAWRATE_P` , `MC_YAWRATE_I` , `MC_YAWRATE_D`)

> Хорошая настройка коэффициентов регулятора угловых скоростей очень важна и влияет на все режимы полета. В то время как плохая настройка регуляторов будет заметна во всех режимах полета, к примеру в режиме Position вы можете видеть подергивания во время зависания аппарата.

Регулятор угловых угловых скоростей можно настраивать как в режиме ACRO, так и в режиме STABILIZED.

Предпочтительнее настраивать регуляторы в режиме ACRO, поскольку вам будет легче визуально заметить произведенные изменения коэффициентов. Если вы собираетесь использовать этот режим отключите Expo-параметры и снизьте чувствительность стиков для облегчения управления.

- `MC_ACRO_EXPO` = 0, `MC_ACRO_EXPO_Y` = 0, `MC_ACRO_SUPEXPO` = 0, `MC_ACRO_SUPEXPOY` = 0
- `MC_ACRO_P_MAX` = 200, `MC_ACRO_R_MAX` = 200
- `MC_ACRO_Y_MAX` = 100

Режим STABILIZED легче для управления но также в нем вам будет сложнее заметить изменения поведения вашего аппарата при настройке коэффициентов.

Если ваш аппарат вообще не летает обратите внимание на две основные вещи:

- Если вы видите сильные осцилляции при попытке взлета, уменьшайте все коэффициенты *P* и *D* до тех пор, пока аппарат не поднимется в воздух.
- С другой стороны, если аппарат почти не реагирует на управление передаваемое с пульта, увеличивайте коэффициент *P*.

Концепция настройки регуляторов одинакова как в режиме STABILIZED, так и в режиме ACRO. Итеративно с указанным шагом настраиваете коэффициенты *P* и *D* для крена и тангажа, а затем изменяете *I*. Первоначально вы можете использовать одинаковые значения для крена, когда регуляторы настроены достаточно хорошо вы можете точно донастроить их по крену и тангажу отдельно (если ваш аппарат

симметричен, можете оставить коэффициенты одинаковыми). Идея настройки регулятора рыскания идентична настройке крена и тангажа, за исключением того, что коэффициент *D* может оставаться нулевым.

Настройка коэффициента *P*

Коэффициент *P* (пропорциональный) используется для минимизации ошибки отслеживания и отвечает за скорость отклика, по этому должен быть установлен как можно выше, но без осцилляций.

При настройке коэффициента *P* пользуйтесь двумя основными наблюдениями:

- Если *P* слишком большой: вы увидите высокочастотные осцилляции.
- Если *P* слишком маленький:
  1. Аппарат медленно реагирует на входящее управление
  2. В режиме ACRO аппарат будет постоянно дрейфовать и вам нужно будет его корректировать, чтобы сохранить его уровень.

Настройка коэффициента *D*

Коэффициент *D* (дифференциальный) используется для демпфирования. Этот коэффициент должен быть как можно выше, но таким образом, что бы не было "перестрелов" по управлению.

При настройке коэффициента *D* пользуйтесь двумя основными наблюдениями:

- Если *D* слишком большой: моторы могут подергиваться и сильно нагреваться во время полета, поскольку коэффициент *D* увеличивает шумы управления.
- Если *D* слишком маленький: возникнут "перестрелы" по входящему управляющему сигналу.

Настройка коэффициента *I*

Коэффициент *I* сохраняет "воспоминания" об ошибке. Это значит, что элемент *I* увеличивается в случае, если желаемая скорость не устанавливается в течении некоторого времени. Этот параметр важен для режима ACRO, а также оказывает достаточно сильное влияние на режимы POSITION и OFFBOARD.

- Если *I* слишком большой: вы можете увидеть медленные осцилляции
- Если *I* слишком маленький: можно заметить ошибку по выполнению управляющего воздействия. Также заниженный коэффициент *I* заметен на логах, это характеризуется тем, что на графиках желаемая скорость длительное время отличается от фактической.

Процедура тестирования

После изменения коэффициентов регулятора, для того, чтобы протестировать новые значения, подайте на вход аппарата быстрый ступенчатый ввод. Для этого быстро наклоните стик радиоаппаратуры в сторону, при этом коптер точно должен выполнять переданное управление, без осцилляций и "перестреливания". Поскольку обычно стики радиоаппаратуры подпружинены, в случае если их отпустить, они начинают колебаться, хорошо настроенный аппарат будет колебаться вместе со стиком.

Конфигурация логгера

Для получения более полной информации с графиков вы можете настроить логгер удобным вам образом. Для его настройки вы можете пользоваться параметрами:

- `SDLOG_PROFILE` - включение большого количества функций приводит к увеличению размера файла лога, а также к увеличению требований по скорости записи, перед начало работой убедитесь, что используете накопитель с достаточной пропускной способностью.
  1. default set - запись общих логов системы
  2. estimator replay (EKF2) - более подробное логирование при использовании EKF2
  3. thermal calibration - высокочастотные данные с IMU и барометра
  4. system identification - высокочастотные данные приводов и IMU
  5. high rate - высокочастотные данные радио, угловых скоростей и приводов
  6. debug - для записи пользовательских отладочных топиков
  7. sensor comparison - низкочастотные данные IMU, барометра и компаса, для сравнения показаний датчиков
- SDLOG_MODE
  1. when armed until disarm - лог записывается с момента арма коптера, до момента дизарма коптера
  2. from boot until disarm - лог записывается с момента запуска системы, до момента дизарма коптера
  3. from boot until shutdown - лог записывается с момента запуска системы, до момента выключения системы

Анализ логов

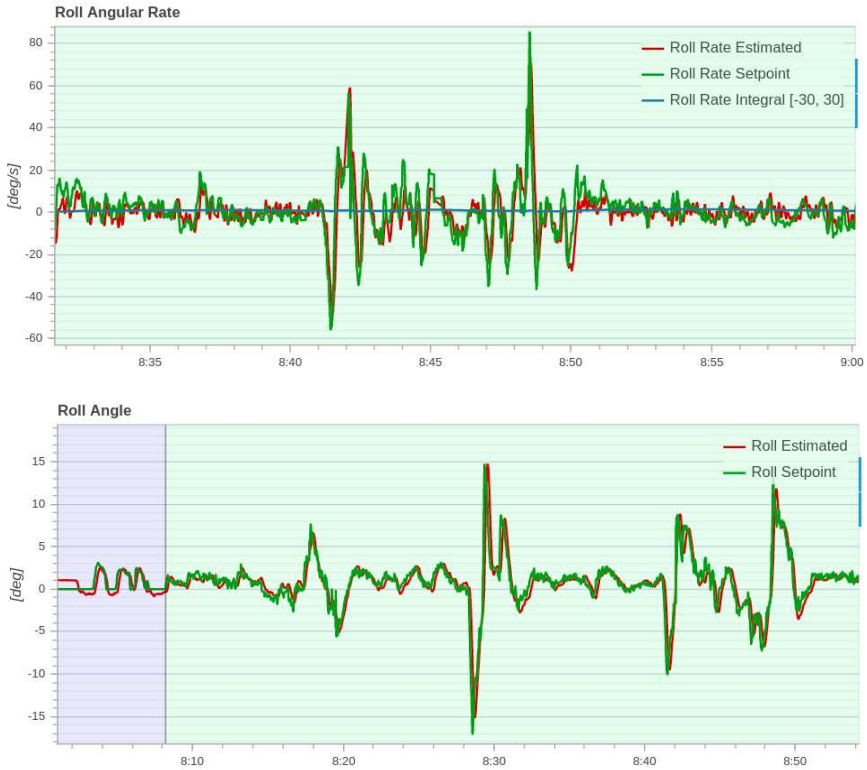
> Настройка параметров с использованием логов сильно сложнее чем визуальная настройка и при неправильных действиях вы можете ухудшить качество полета вашего коптера.

Просмотр логов сильно помогает в настройке и позволяет более точно оценивать качество произведенной настройки.

Среди представленных графиков, больше всего вам могут понадобиться:

- Roll/Pitch/Yaw Angular Rate - графики угловых скоростей
- Roll/Pitch/Yaw Angle - графики углов наклона
- Local Position X/Y/Z - позиция вашего коптера в пространстве
- Step Response for Roll/Pitch/Yaw Rate - отклик коптера на входящее управление

Ниже представлены графики угловых скоростей и углов при хороших настройках регулятора.



На графиках красной линией отмечено рассчитанное значение, а зеленой требуемое.

Качество настройки характеризуется тем, что рассчитанное значение должно быть максимально близким к требуемому. Если оба графика совпадают, это значит, что ваш коптер точно выполняет все переданные ему команды, если же графики сильно отличаются, во время полета вы заметите, что коптер неправильно выполняет ваши команды управления.

Регулятор положения

Данный регулятор является вторым уровнем каскада и настраивается после регуляторов угловых скоростей. Он отвечает за угол наклона коптера и настраивается с помощью параметров:

- Регулятор угла по крену (`MC_ROLL_P`).
- Регулятор угла по тангажу (`MC_PITCH_P`).
- Регулятор угла по рысканью (`MC_YAW_P`).

В данном регуляторе настраиваемыми являются только пропорциональные значения и в большинстве случаев их стоит оставить со стандартными значениями.

Идея настройки данных параметров точно такая же, как и настройки P параметров регуляторов угловых скоростей. При высокочастотных осцилляциях уменьшите значение, пре медленном выполнение команды увеличьте его.

Навигация по вертикальным ArUco-маркерам

Алгоритм навигации по визуальным ArUco-маркерам, реализованный в образе Клевера, поддерживает гибкую настройку положения маркеров в пространстве, что позволяет располагать их на любой поверхности, под любым углом.

Установка вертикального крепления камеры

Для более точного распознавания маркеров необходимо установить камеру вертикально таким образом, чтобы объектив был направлен параллельно горизонту.

> Конфигурационный файл позволяет настраивать расположение камеры в пространстве относительно коптера любым образом. Для удобства далее будет рассматриваться вариант установки камеры под 90° к горизонту, по направлению носа коптера.

Крепление камеры, 3D печать

Распечатайте крепление камеры.

Установите крепление в удобное место таким образом, чтобы в камере было минимальное количество лишних объектов (защита, ножки, пропеллеры, лучи) — все это будет негативно сказываться на распознавании маркеров.

Настройка расположения камеры

Чтобы задать расположение камеры под необходимым углом, откройте файл `main_camera.launch`, расположенный в `~/catkin_ws/src/clover/clover/launch/`.

```
nano ~/catkin_ws/src/clover/clover/launch/main_camera.launch
```

Версии 0.20+

В параметрах `direction_x`, `direction_y` установите пустые значения вручную или введите строки:

```
sed -i "/direction_z/s/default=\".*\"/default=\"\"/" /home/pi/catkin_ws/src/clo
sed -i "/direction_y/s/default=\".*\"/default=\"\"/" /home/pi/catkin_ws/src/clo
```

Отредактируйте одну из конфигурационных строк или добавьте строку, представленную ниже:

```
<node pkg="tf2_ros" type="static_transform_publisher" name="main_camera_frame"
```

> Одновременно может использоваться только одна конфигурация камеры — если вы вставляете представленную выше строку, не забудьте закомментировать активную на данный момент. Для определения этого вам поможет подсветка синтаксиса — активная строка будет подсвечена отличным от комментариев цветом. Для комментирования в начало и конец строки добавьте символы `<!--` и `-->` соответственно.

Если на используемой вами карте, маркеры имеют равное расстояние по осям *X* и *Y*, можете воспользоваться утилитой для создания карт `gen_map.py`. В ином случае, вам потребуется задать их вручную — для этого перейдите в директорию `~/catkin_ws/src/clover/aruco_map/map` и создайте файл карты `map_name.txt`. Заполните вашу карту в соответствии с синтаксисом карт. Пример карты маркеров со случайным расположением маркеров:

> При введении карты выберите один из маркеров в качестве начала координат и относительно него отмеряйте расстояние до всех остальных маркеров. Если все ваши маркеры ориентированы одинаково, вы можете не указывать все 8 параметров, а указать только первые 5: индекс маркера, размер и его расположение в пространстве по осям x, y, z соответственно.

```
106 0.33    0   0   0
103 0.33    1.53    0.23    0
153 0.40    -0.56   1.36    0
```

После того, как вы заполните карту, необходимо применить ее — для этого отредактируйте файл `aruco.launch`, расположенный в `~/catkin_ws/src/clover/clover/launch/`. Измените в нем строку `<param name="map" value="$(find aruco_pose)/map/map_name.txt"/>`, где `map_name.txt` название вашего файла с картой.

При использовании маркеров, не привязанных к горизонтальным плоскостям (пол, потолок), необходимо также сделать пустым значение аргумента `placement` в том же файле:

```
<arg name="placement" default=""/>
```

После всех настроек вызовите `sudo systemctl restart clover` для перезагрузки сервиса `clover`.

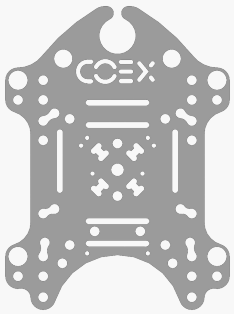
CAD-модели

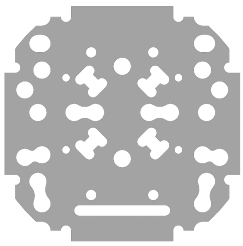
На этой странице представлены CAD-модели некоторых деталей квадрокоптеров Клевер.

Клевер 4.2

Фрезеровка

Изображение	Деталь	Файлы

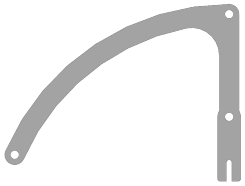
	**Дека монтажная**. Функция: Дека для установки АКБ и Raspberry Pi. Материал: Монолитный поликарбонат 2мм. Количество: 1 шт.	`Deck Mount.dxf`
		`Deck Mount.ipt`
		`Deck Mount.stp`

	**Дека монтажная малая**. Функция: Дека для установки FPV камеры и крепления пластин жесткости. Материал: Монолитный поликарбонат 2мм. Количество: 1 шт.	`Deck Mount Small.dxf`
		`Deck Mount Small.ipt`
		`Deck Mount Small.stp`

	**Дека захвата**. Функция: Дека для установки захватов и внешней периферии (камера, дальномер). Материал: Монолитный поликарбонат 2мм. Количество: 1 шт.	`Grab Deck.dxf`
		`Grab Deck.ipt`
		`Grab Deck.stp`

	Пластина для LED. Функция: Крепление светодиодной ленты. Материал: Монолитный поликарбонат 2мм. Количество: 1 шт.	`LED mount plate.dxf`
		`LED mount plate.ipt`
		`LED mount plate.stp`

	**Дуга**. Функция: Предотвращение повреждения пропеллеров. Материал: Монолитный поликарбонат 2мм. Количество: 4 шт.	`Prop Guard.dxf`
		`Prop Guard.ipt`
		`Prop Guard.stp`

	**Дуга монтажная**. Функция: Дуга для закрепления контура защиты. Материал: Монолитный поликарбонат 2мм. Количество: 2 шт.	`Prop Guard Mount.dxf`
		`Prop Guard Mount.ipt`
		`Prop Guard Mount.stp`

	**Ножка маленькая**. Функция: Стандартный опорный элемент. Материал: Монолитный поликарбонат 2мм. Количество: 2 шт.	`Small Leg.dxf`
		`Small Leg.ipt`
		`Small Leg.stp`

	**Луч**. Функция: крепление моторов. Материал: углепластик композитный (карбон) 2мм. Количество: 4 шт.	`Arm.dxf`
		`Arm.ipt`
		`Arm.stp`

	**Пластина центральная**. Функция: установка электроники. Материал: углепластик композитный (карбон) 2мм. Количество: 1 шт.	`Central Plate.dxf`
		`Central Plate.ipt`
		`Central Plate.stp`

	**Пластина монтажная малая**. Функция: прижатие пластины жесткости. Материал: углепластик композитный (карбон) 2мм. Количество: 1 шт.	`Deck Mount Small.dxf`
		`Deck Mount Small.ipt`
		`Deck Mount Small.stp`

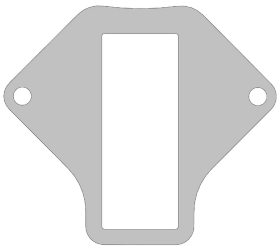
	**Пластина жесткости**. Функция: усиление рамы. Материал: углепластик композитный (карбон) 2мм. Количество: 4 шт.	`Stiffener Plate.dxf`
		`Stiffener Plate.ipt`
		`Stiffener Plate.stp`

Клевер 4.2 WorldSkills

Фрезеровка

Изображение	Деталь	Файлы

	**Ножка большая**. Функция: Опорный элемент увеличенной высоты. Материал: Монолитный поликарбонат 2мм. Количество: 2 шт.	`Big Leg.dxf`
		`Big Leg.ipt`
		`Big Leg.stp`

	**Проставка для захвата**. Функция: Опорный элемент для механического захвата. Материал: Монолитный поликарбонат 2мм. Количество: 1 шт.	`Grab Spacer.dxf`
		`Grab Spacer.ipt`
		`Grab Spacer.stp`

3D печать

Механический захват

- **Левая клешня**: `grip_left.stl`.
- **Правая клешня**: `grip_right.stl`.

Материал: SBS Glass. Заполнение 100%. Количество: 1 шт.

Груз для магнитного захвата

- **Груз для магнитного захвата**: `load_for_magnetic_grip.stl`.
- **Дополнение для подставки груза**: `add-on_for_load_support.stl`.
- **Подставка под теннисный мяч для магнитного захвата**: `tennis_ball_stand_for_magnetic_grip.stl`.

Материал: PETG. Заполнение 100%. Количество: 1 шт.

Клевер 4

3D печать

- **АКБ холдер** – `battery_holder.stl` Функция: Фиксация АКБ и тестера напряжения. Материал: ABS пластик(или аналог). Заполнение не менее 50%. Количество: 1шт.

Дополнительные модели для Клевер 4

Усиленная пластина для монтажа Jetson Nano и дополнительного оборудования, автор: Вячеслав Бузов.

Лазерная резка

- **Усиленная пластина - основа** – `reinforced_plate_base.dxf` Функция: Крепление компьютеров формата Jetson Nano и виброразвязок на раме Клевер 4. Материал: Монолитный поликарбонат 2мм. Количество: 1шт.
- **Ребро жёсткости усиленной пластины** – `reinforced_plate_rib.dxf` Функция: Увеличение жёсткости сборки. Материал: Монолитный поликарбонат 2мм. Количество: 2шт.
- **Пластина для камеры** – `reinforced_plate_camera_pad.dxf` Функция: Крепление камеры при использовании усиленной пластины. Материал: Монолитный поликарбонат 2мм. Количество: 1шт.

Клевер 3

3D печать

- **Кейс для камеры** – `camera_case.stl` Функция: Кейс для крепления модуля камеры. Материал: PLA/ABS(или аналог). Заполнение 30%. Количество: 1шт. **Крепежная пластина для камеры** – `camera_mount.stl` . Функция: Крепежный модуль, для монтажа кейса. Материал: PLA/ABS(или аналог). Заполнение 30%. Количество: 1шт. **Пластина для камеры** – `camera_plate.stl` . Функция: Пластина для закрепление модуля камеры в кейсе. Материал: PLA/ABS(или аналог). Заполнение 30%. Количество: 1шт.
- **Малая монтажная дека** – `mounting_deck_small.stl` . Функция: Крепление камеры и полетного контроллера. Материал: PLA/ABS(или аналог). Заполнение 60%. Количество: 1шт.

Лазерная резка

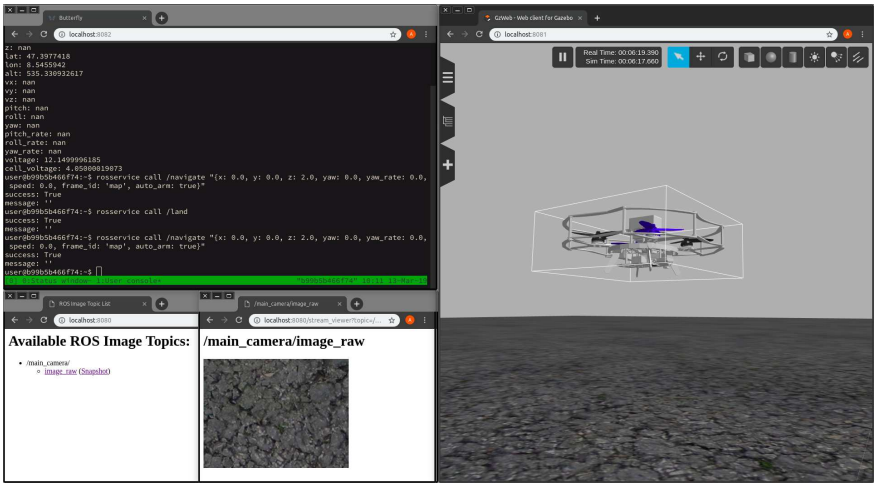
- **Ножка** (вариант с захватом) – `big_leg.dxf` Функция: Опорный элемент для ситуации установки захвата. Материал: Монолитный поликарбонат 2мм. Количество: 4шт.
- **Дека монтажная** – `deck.dxf` Функция: Крепление АКБ сопутствующей периферии. Материал: Монолитный поликарбонат 2мм. Количество: 1шт.
- **Дуга** – `prop_guard.dxf` Функция: Элемент каркаса защиты. Материал: Монолитный поликарбонат 2мм. Количество: 16шт.
- **Рогатка + ушко** – `prop_guard_mount.dxf` Функция: Крепление защиты пропеллеров к раме. Материал: Монолитный поликарбонат 2мм. Количество: 4шт.
- **Проставка** – `grab_spacer.dxf` . Функция: Монтаж сервопривода к деке захвата. Материал: Монолитный поликарбонат 2мм. Количество: 1шт.
- **Ножка** – `leg.dxf` . Функция: Опорный элемент. Материал: Монолитный поликарбонат 2мм. Количество: 4шт.
- **Обруч (LED)** – `led_mount_plate.dxf` . Функция: Крепление светодиодной ленты. Материал: Монолитный поликарбонат 2мм. Количество: 4шт.
- **Малая монтажная дека** – `mounting_deck_small.dxf` . Функция: Крепление камеры и полетного контроллера. Материал: Монолитный поликарбонат 2мм. Количество: 1шт.

Фрезеровка

- **Дека центральная** – `central_plate.dxf` . Функция: Несущая часть коптера. Материал: Стеклотекстолит/карбон 2мм. Количество: 1шт.
- **Луч** – `arm.dxf` . Функция: Крепление моторов. Материал: Стеклотекстолит/карбон 2мм. Количество: 4шт.

Docker-контейнер с преднастроенным SITL

> Рекомендуется использовать образ виртуальной машины или нативную установку для работы в симуляторе.



Для упрощения запуска симулятора предлагается использовать предварительно настроенный Docker-контейнер с симулятором Gazebo, автопилотом PX4 и предустановленными пакетами Клевера.

Состав контейнера

В Docker-контейнере с симулятором установлены и настроены:

- Симулятор Gazebo с плагинами для симуляции камеры, дальномера и связью с ROS
- Пакеты ROS, требуемые для запуска нод Клевера
- Собранный для симулятора PX4
- Легковесный web-интерфейс для Gazebo Gzweb
- Web-терминал Butterfly

Предварительная настройка

Запуск контейнера рекомендуется производить на ОС Ubuntu версии не ниже 16.04 с использованием Docker версии не ниже 18.09. Для комфортной работы с симулятором следует использовать компьютер с не менее чем 4 ядрами CPU (Intel Core i5/i7, не ниже Haswell) и не менее чем 8 ГБ ОЗУ. Работа с симулятором может происходить как на компьютере с запущенным контейнером, так и на другом компьютере в той же локальной сети.

> Не забудьте установить и настроить Docker на своей системе!

Запуск симулятора

Для запуска симулятора можно использовать следующую команду:

```
docker pull sfalexrog/clever-sitl:slim

docker run \
    -it \
    --rm \
    -p 14556:14556 \
    -p 14557:14557 \
    -p 8080:8080 \
    -p 8081:8081 \
    -p 57575:57575 \
    -p 9090:9090 \
    -p 35602:35602 \
    -p 2222:22 \
    --name clever_sitl \
    sfalexrog/clever-sitl:slim
```

> Здесь и далее предполагается, что при настройке Docker на своей системе Вы настроили запуск Docker от обычного пользователя (раздел "Manage Docker as a non-root user")

Первая команда загружает последнюю версию контейнера с симулятором `sfalexrog/clever-sitl:slim`, вторая его запускает. Ключ `-p` позволяет задать соответствие между портом компьютера, на котором запущен контейнер, и портом "внутри" контейнера. Порты 14556 и 14557 нужны для подключения к симулятору с помощью QGroundControl, порт 8080 - для просмотра топиков ROS с изображениями и видеопотоками, порт 8081 - для подключения к визуализации симуляции Gazebo, порт 57575 - для доступа к web-терминалу Butterfly.

После запуска контейнера можно перейти по следующим ссылкам в браузере для доступа к сервисам симулятора:

- http://localhost:8080 - просмотр топиков с камеры (аналогично тому, как это сделано в Клевере)
- http://localhost:8081 - визуализация текущего состояния симулятора через Gzweb
- http://localhost:57575 - Web-терминал Butterfly с запущенным сеансом tmux

Доступ к этим сервисам также есть с других компьютеров, расположенных в той же локальной сети; для этого в ссылках, указанных выше, следует `localhost` поменять на IP-адрес компьютера с запущенным контейнером.

Работа с симулятором

Основное взаимодействие с симулятором происходит через Web-терминал Butterfly. По умолчанию в нём открывается сессия tmux, так что происходящее в терминале можно демонстрировать сразу на нескольких

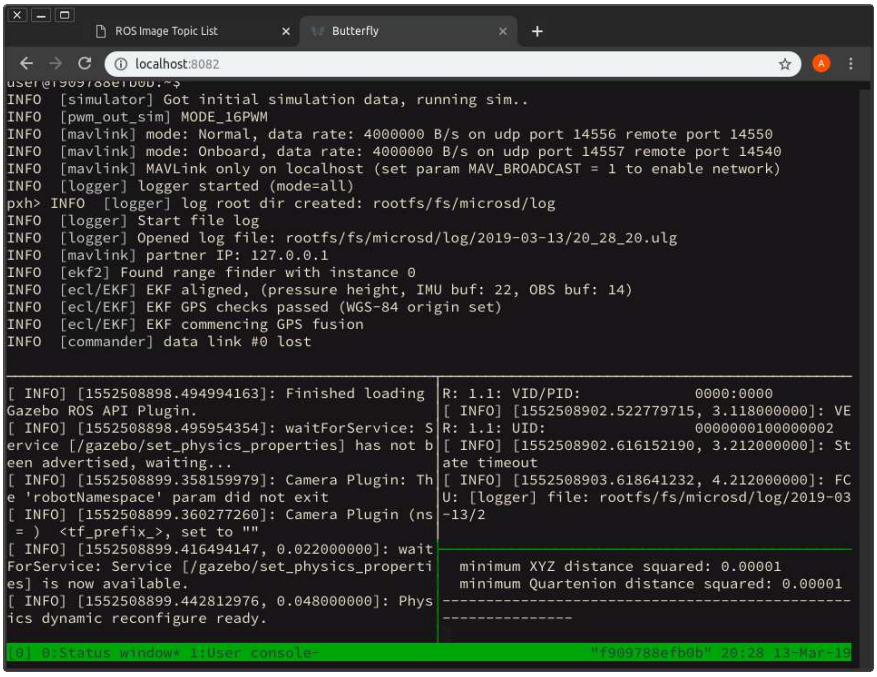
компьютерах.

> Можно также создавать дополнительные сеансы средствами Docker. Для этого воспользуйтесь командой `docker exec -it clever_sitl /bin/bash`

В web-терминале работают команды ROS, доступны редакторы vim и nano, поддерживается работа с интерпретатором Python. В симуляторе можно запускать программы, написанные для Клевера и не использующие специфический функционал бортового компьютера или периферии (например, LED-ленты).

Визуализация текущего состояния симулированного мира доступна в Gzweb. Камеру можно перемещать, передвигая мышь с зажатой левой кнопкой. Поворот камеры производится при зажатой средней кнопке мыши, приближение/удаление камеры - при повороте колёсика. При выполнении этих манипуляций появляется небольшая жёлтая сфера, означающая центр поворота/приближения.

В web-терминале также можно просмотреть текущее состояние PX4, отладочный вывод нод Клевера, лог Gazebo и Gzweb. Для этого надо переключиться на нулевой экран tmux; это делается комбинацией клавиш `ctrl+B` и последующим нажатием клавиши `0`. Появится примерно следующее:



Верхняя панель - отладочная консоль PX4; на нижней панели: слева находится отладочный вывод симулятора Gazebo, справа сверху - лог нод Клевера, справа снизу - лог Gzweb.

Для переключения между панелями следует использовать комбинацию клавиш `ctrl+B` и последующее нажатие кнопки `Q` и номера панели. Номера панелей будут кратковременно выведены поверх самих панелей при нажатии `ctrl+B` и `Q`.

Подключение локальных директорий к контейнеру

Для подключения директории, доступной как на основной системе, так и в контейнере, при запуске следует указать ключ `-v` с указанием директории в основной системе и в контейнере.

Так, для того, чтобы текущая директория стала доступна в контейнере по пути `/home/user/data`, запустите контейнер со следующими параметрами:

```
docker run \
    -it \
    --rm \
    -p 14556:14556 \
    -p 14557:14557 \
    -p 8080:8080 \
    -p 8081:8081 \
    -p 57575:57575 \
    -p 9090:9090 \
    -p 35602:35602 \
    -p 2222:22 \
    -v $(pwd):/home/user/data:rw \
    --name clever_sitl \
    sfalexrog/clever-sitl:slim
```

> В команде для запуска контейнера ключ `-v` может встречаться многократно. Это позволяет указать несколько общих директорий.

Разумно использовать механизм подключения локальных директорий для добавления своих моделей в симулятор, сохранения и загрузки параметров PX4, а также своих программ для Клевера на Python.

> Программы, скомпилированные в контейнере, могут не запускаться на основной системе. Аналогично, программы, скомпилированные на основной системе, могут не запускаться в контейнере. Это следует учитывать при написании нод, использующих компилируемые языки.

Завершение работы с симулятором

Для завершения работы с симулятором достаточно завершить работу соответствующего контейнера. Это можно сделать с помощью инструментов управления Docker-контейнерами или нажатием комбинации клавиш `ctrl+C` в терминале, в котором был запущен контейнер.

Увеличение скорости работы симулятора

> Предложенный в этом режиме метод является экспериментальным; вполне возможно, что он не заработает, и в этом случае некоторые элементы симулятора также перестанут работать

По умолчанию симулятор будет создавать изображения в симулируемых камерах, используя программную растеризацию. Это создаёт повышенную нагрузку на CPU компьютера с запущенным симулятором, а также не позволяет получить приемлемую частоту кадров для большинства задач компьютерного зрения. Как правило, Docker не используют для графически интенсивных задач, поэтому возможности по увеличению производительности ограничены.

При использовании достаточно современного графического оборудования с открытыми драйверами (например, Intel HD Graphics 520+ и mesa на Linux) можно попробовать "пробросить" сеанс X Server и видеокарту в контейнер; в этом случае будет использована гораздо более быстрая аппаратная растеризация.

Для проброса видеокарты в контейнер следует выполнить следующие команды на компьютере, на котором будет запущен контейнер:

```
touch /tmp/.docker.xauth

xauth nlist $DISPLAY | sed -e 's/^..../ffff/' | xauth -f /tmp/.docker.xauth nme

docker run \
    -it \
    --rm \
    -v /tmp/.X11-unix:/tmp/.X11-unix:rw \
    -v /tmp/.docker.xauth:/tmp/.docker.xauth:rw \
    -e DISPLAY=$DISPLAY \
    -e XAUTHORITY=/tmp/.docker.xauth \

    --device /dev/dri/card0:/dev/dri/card0 \
    -p 14556:14556 \
    -p 14557:14557 \
    -p 8080:8080 \
    -p 8081:8081 \
    -p 57575:57575 \
    -p 9090:9090 \
    -p 35602:35602 \
    -p 2222:22 \
    sfalexrog/clever-sitl:slim
```

При этом на компьютере с контейнером должна быть запущена графическая среда, использующая X Server.

Установка и настройка пакета ROS Melodic

Для работы с такими инструментами как: rqt, rviz и т. д., а также для запуска симулятора (SITL) вам потребуется установленный и настроенный пакет ROS.

> Более подробную инструкцию по установке смотрите в основной статье.

Установка ROS Melodic на Ubuntu

Для того, чтобы загрузить и установить правильную версию пакета требуется сделать настройки репозиториев, для этого откройте "Программы и обновления" и разрешите `restricted`, `universe` и `multiverse`.

Настройте свою систему, для того что бы вы могли принимать программное обеспечение с `packages.ros.org`, выполнив команду:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) ma
```

Настройте ключи доступа в своей системе для правильной загрузки:

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key C1
```

Убедитесь в том, что вы имеете последние версии индексов пакетов:

```
sudo apt-get update
```

Теперь установите сам пакет ROS.

- Если вы планируете использовать ROS вместе с симуляцией (также содержит инструменты: rqt, rviz и т. д):

  ```
  sudo apt-get install ros-melodic-desktop-full
  ```

- Если вы планируете использовать ROS исключительно работать с инструментами rqt, rviz и т. д:

  ```
  sudo apt-get install ros-melodic-desktop
  ```

После установки пакета вам нужно инициализировать `rosdep`. Пакет `rosdep` позволит вам легко устанавливать системные зависимости для источника, который вы хотите скомпилировать, а также необходим для запуска

некоторых основных компонентов в ROS:

```
sudo rosdep init
rosdep update
```

Если вам не удобно запускать переменное окружение вручную каждый раз, вы можете настроить его так, чтобы оно добавлялось в ваш сеанс bash при каждом запуске новой оболочки:

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Если вы хотите установить какие-либо дополнительные пакеты для вашего ROS Melodic просто используйте:

```
sudo apt-get install ros-melodic-PACKAGE
```

Калибровка камеры

Калибровка камеры может значительно повысить качество работы модулей, связанным с компьютерным зрением: распознавание ArUco-маркеров и Optical Flow.

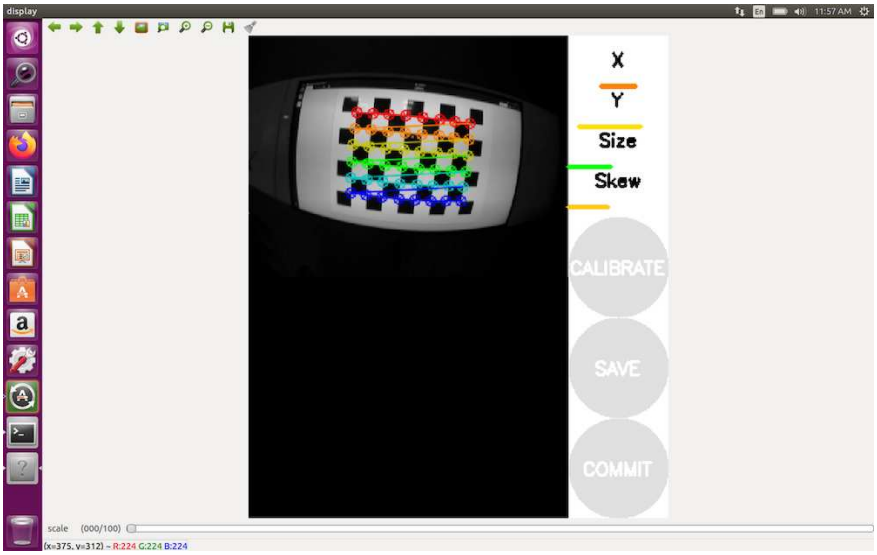
При калибровке камеры подбираются параметры, наиболее хорошо описывающие конкретный установленный объектив. Данные параметры включают в себя фокусные расстояния, расположение точки principal point (которое зависит от того, насколько ровно по центру установлен объектив), коэффициенты дисторсии *D*. Подробнее про использующуюся модель искажений камеры можно прочитать в документации OpenCV.

Существует несколько инструментов, которые позволяют откалибровать камеру и прописать вычисленные параметры в систему. Обычно, они используют калибровочные изображения: "шахматные доски" (*Chessboard*), а так же комбинации шахматной доски и сетки ArUco-маркеров (*ChArUco*).

ROS-пакет camera_calibration

Основной туториал:
http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration.

Для калибровки камеры с использованием ROS-пакета camera_calibration необходим компьютер с установленным ОС GNU/Linux и ROS Noetic.



1. Используя Терминал, установите на компьютер пакет `camera_calibration`:

   ```
   sudo apt-get install ros-noetic-camera-calibration
   ```

2. Скачайте калибровочную доску – `chessboard.pdf`. Распечатайте доску на принтере либо выведите ее на экран компьютера.

3. Подключитесь к Wi-Fi Клевера.
4. Запустите калибровку (на компьютере):

```
ROS_MASTER_URI=http://192.168.11.1:11311 rosrun camera_calibration camerac
```

> Вместо значения *0.108* укажите реальный размер квадрата на распечатанной доске или на экране (в метрах). Например, значение *0.03* будет соответствовать 3 см.

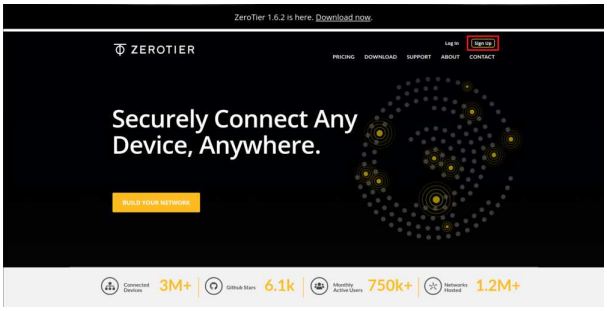
5. Когда программа для калибровки запустится, начните перемещать дрон таким образом, чтобы калибровочная доска попадала в кадр под разными углами.
   - Перемещайте калибровочную доску в левый, правый, верхний и нижний торец кадра.
   - Вращайте калибровочную доску вокруг всех 3-х осей.
   - Отдаляйте и приближайте камеру к калибровочной доске.
6. Нажмите кнопку *CALIBRATE*, когда она станет активной. Процесс вычисления параметров калибровки займет несколько минут.

   Когда калибровка завершится, в терминале вы увидите полученные параметры. В окне отобразится изображение с выправленными искажениями. При успешной калибровке все реальные прямые линии должны остаться прямыми на полученном изображении.
7. Нажмите *COMMIT*, чтобы сохранить полученные параметры калибровки. Результат будет записан в файл калибровки основной камеры Клевера: `/home/pi/catkin_ws/src/clover/clover/camera_info/fisheye_cam.yaml`.

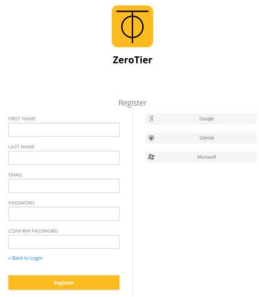
Создание виртуальной сети ZeroTier и подключение к ней

Создание и настройка сети ZeroTier

1. Зайдите на сайт ZeroTier.

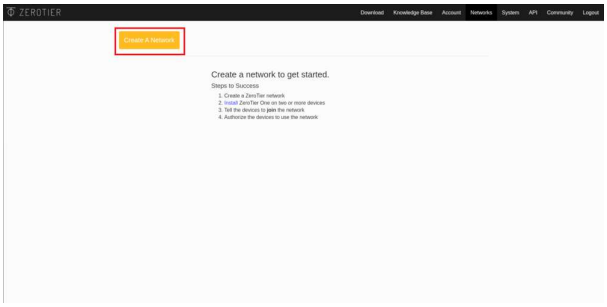


2. Зарегистрируйтесь в ZeroTier.

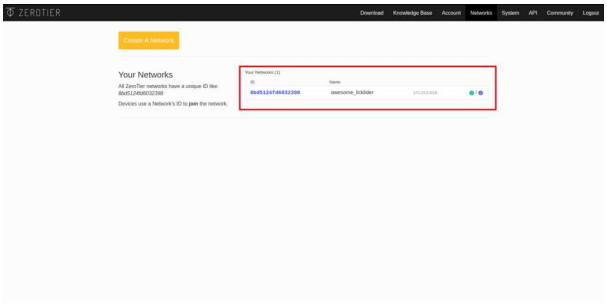


3. Зайдите в свой аккаунт.

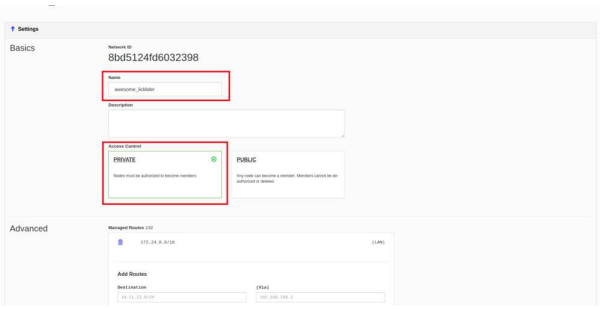
4. Нажмите кнопку *Create A Network*.



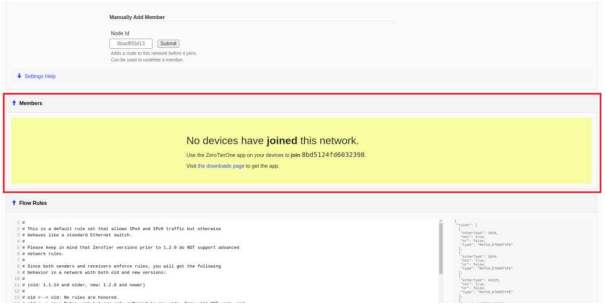
5. После этого вы увидите созданную вами сеть, ее ID и название. Для настройки сети нажмите на нее.



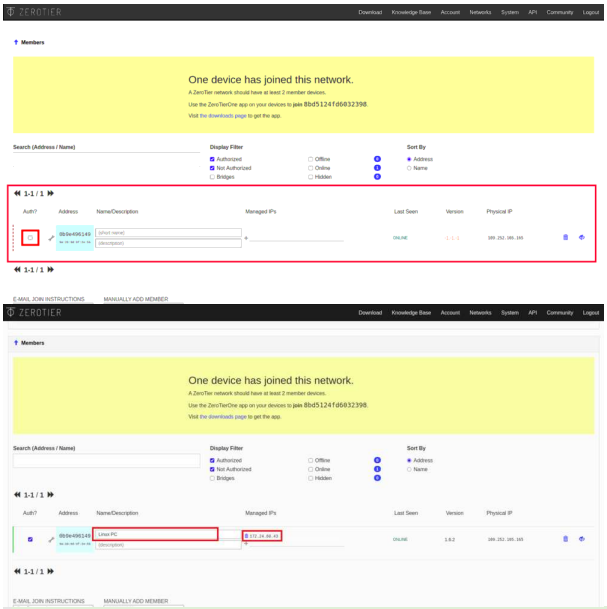
6. В открывшемся окне можно изменить имя сети и приватность подключения.



7. Пролистайте ниже, до графы *Members*. В ней будет написано о том, что в сети нету пользователей.



8. Устройства подключенные к сети будут отображаться в данной графе, для того, чтобы позволить им подключиться к сети, активируйте чекбокс *Auth?*. При этом, подключенному устройству автоматически выдастся внутренний IP адрес, в дальнейшем он будет использоваться для связи с данным устройством.



Указывайте имена для новых устройств, этом поможет вам в дальнейшем отличать их друг от друга.

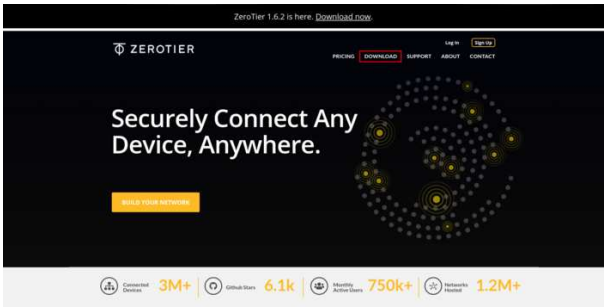
9. Повторите последний шаг для всех подключаемых устройств.

Сеть ZeroTier в случае бесплатного использования поддерживает до 50 пользователей одновременно.

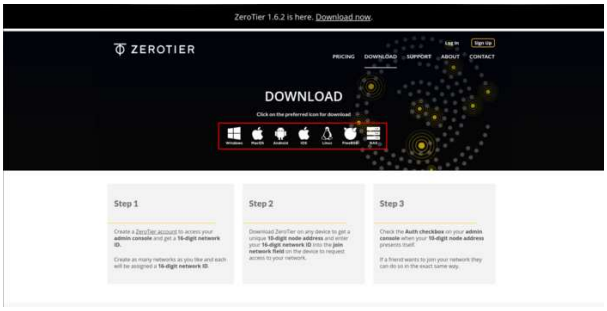
Настройка на Windows

Установка приложения

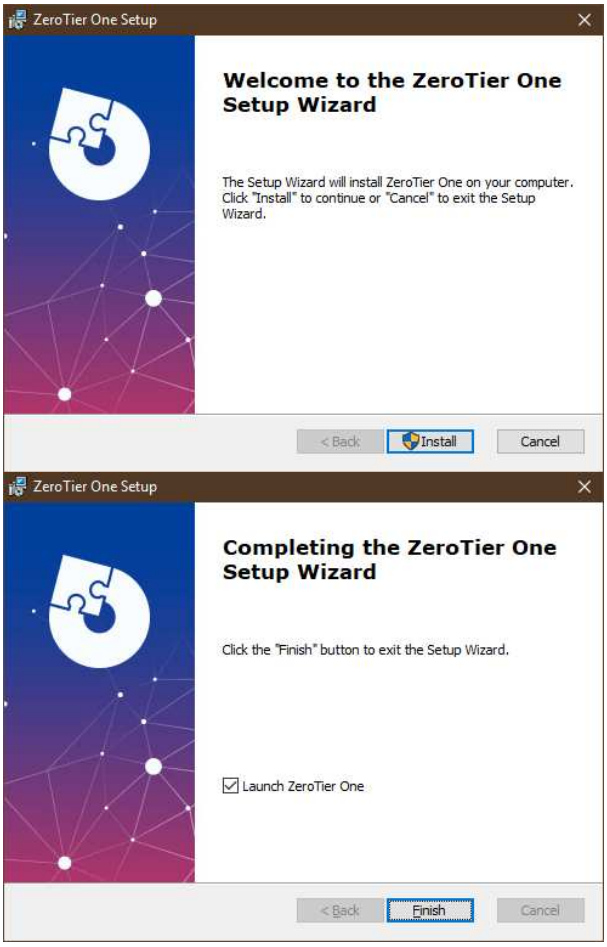
1. Перейдите на сайт ZeroTier.



2. Нажмите на иконку Windows.

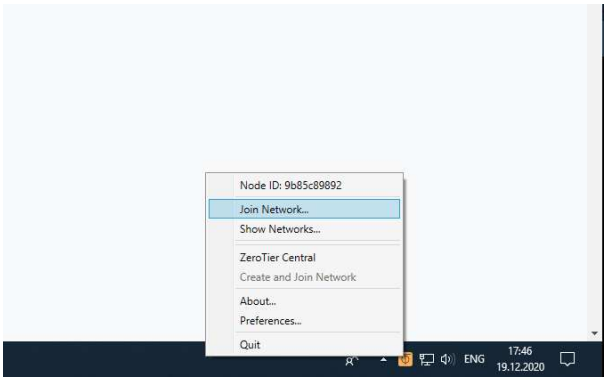


3. Скачайте и запустите файл `ZeroTier One.msi`.

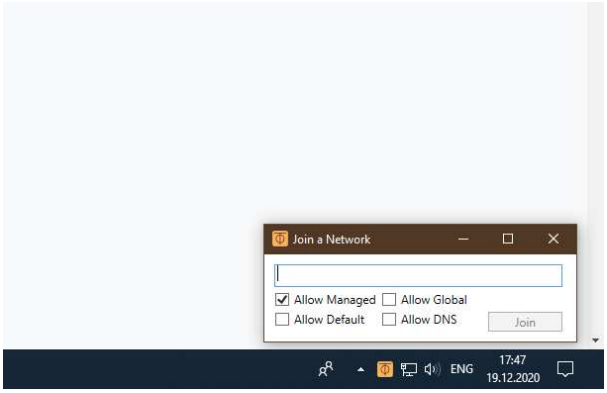


Подключение к сети

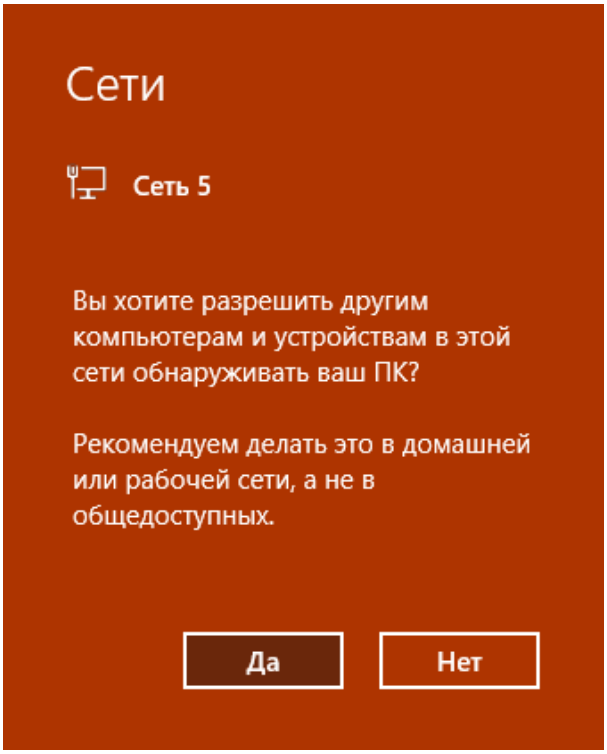
1. Запустите ZeroTier One.
2. Нажмите на иконку ZeroTier One в панели задач.
3. Нажмите на кнопку *Join Network...* для подключения к сети.



4. В появившемся окне введите ID вашей сети и нажмите кнопку *Join*.



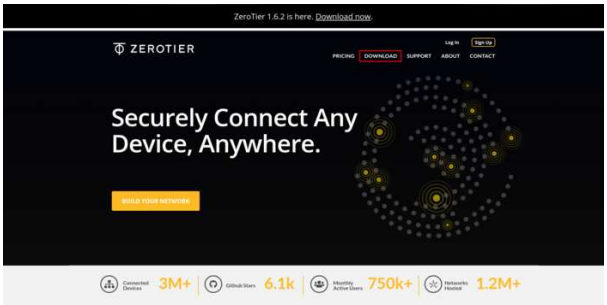
5. Разрешите использование новой сети.



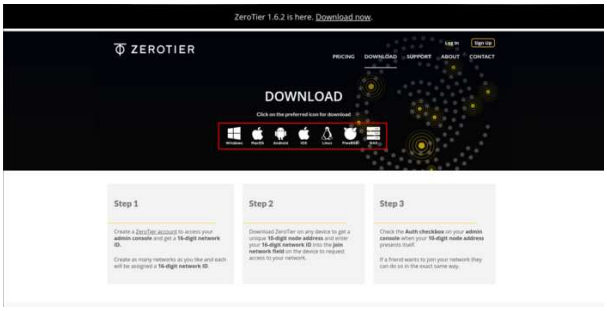
Настройка на iOS

Установка приложения

1. Перейдите на сайт ZeroTier.



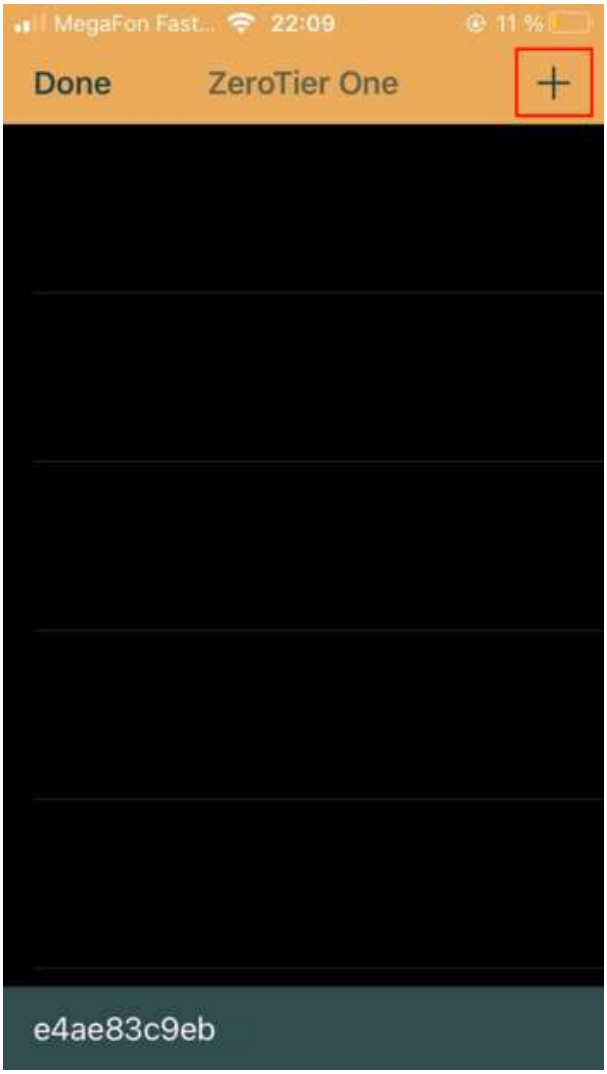
2. Нажмите на иконку iOS.



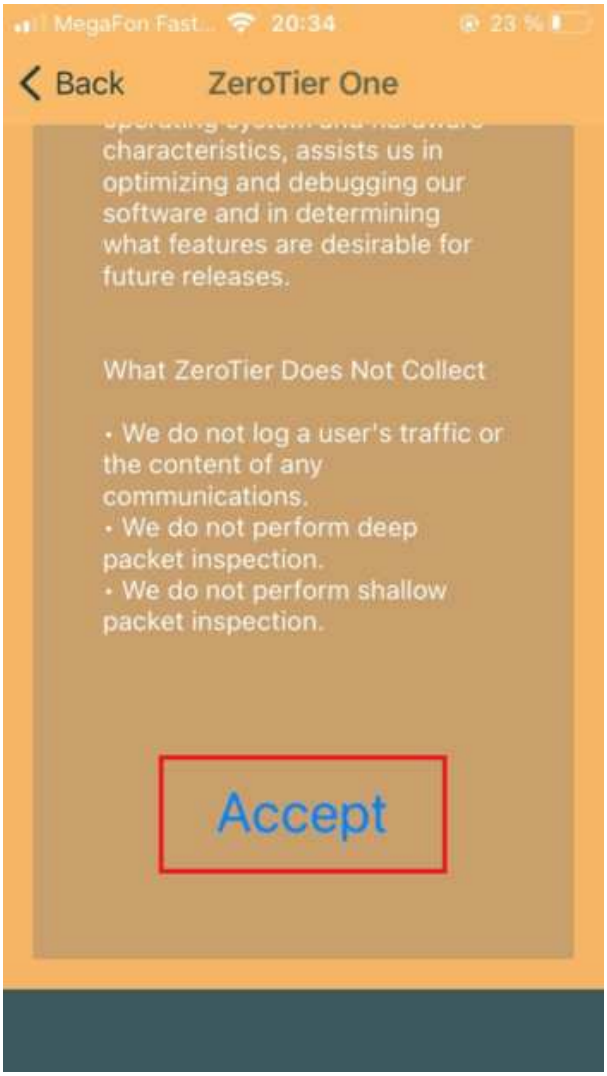
3. Установите приложение *ZeroTier One*.

Подключение к сети

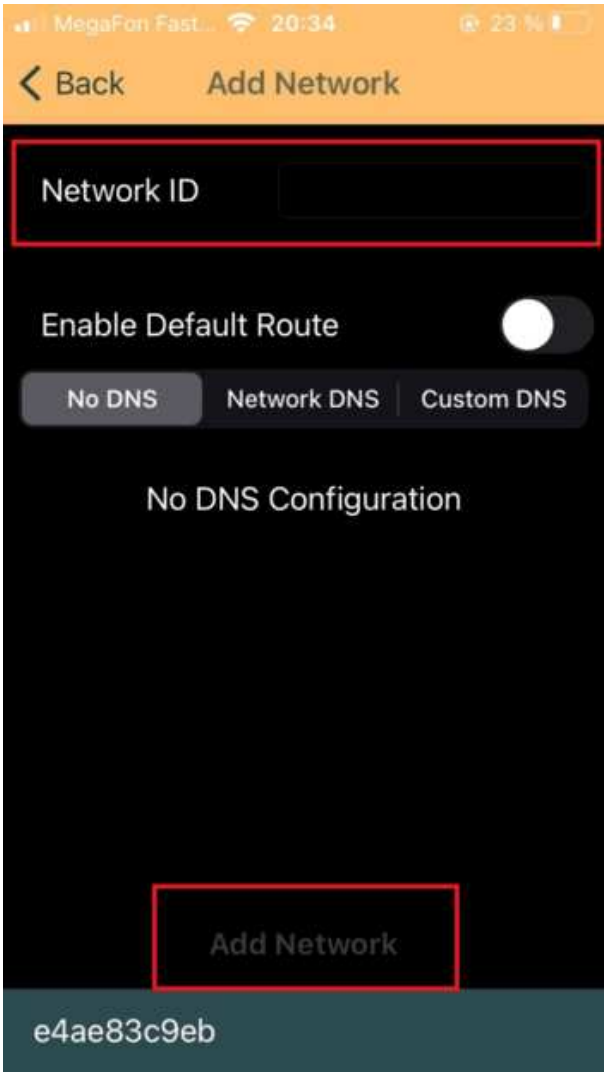
1. Запустите приложение *ZeroTier One*.
2. Нажмите на + для добавления нового подключения.



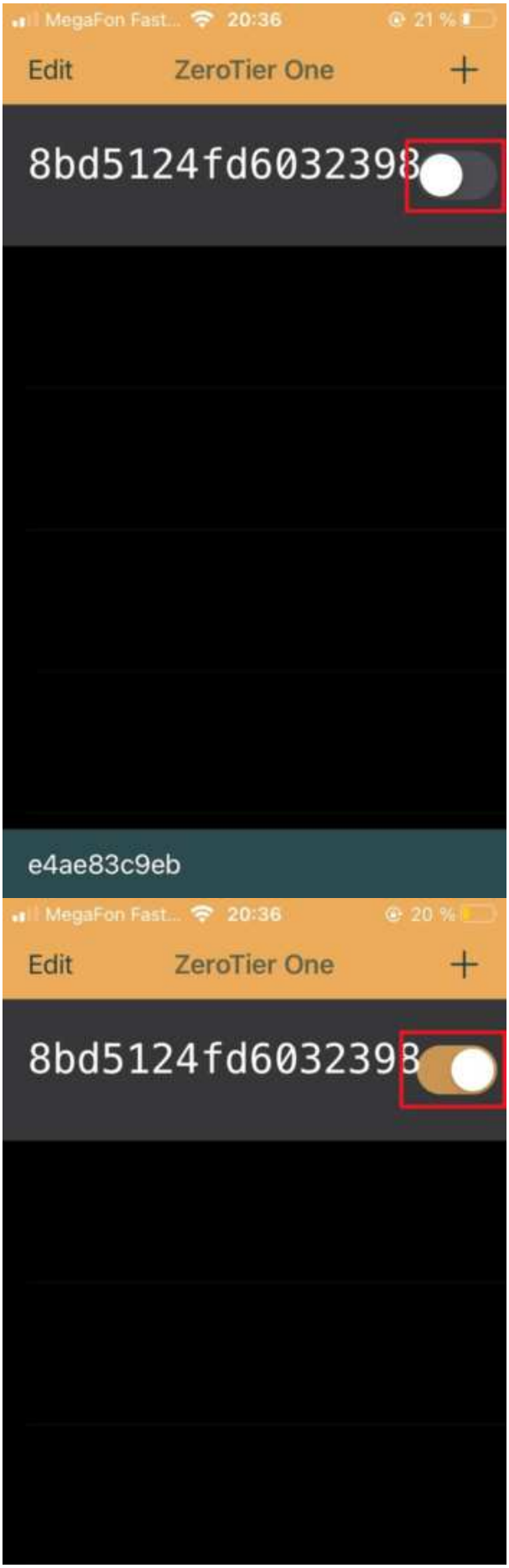
3. Подтвердите политику конфиденциальности.

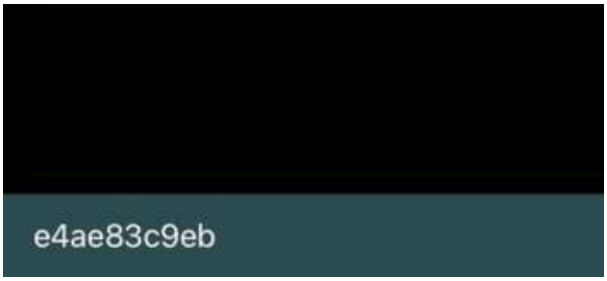


4. Введите ID вашей сети и нажмите кнопку *Add Network*.



5. Подтвердите добавление новой конфигурации VPN.
6. Подключитесь к VPN сети, сдвинув ползунок активации сети.


MegaFon Fast...
20:36
21 %
Edit
ZeroTier One
8bd5124fd6032398
e4ae83c9eb
MegaFon Fast...
20:36
20 %
Edit
ZeroTier One
8bd5124fd6032398



Настройка на Linux (PC, Raspberry Pi)

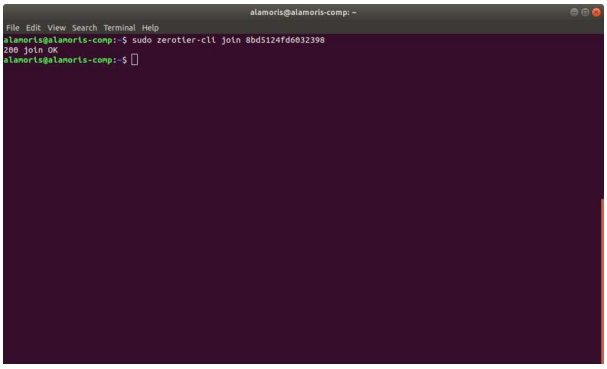
Установка приложения

1. Откройте консоль, для этого нажмите сочетание клавиш *Ctrl* + *Alt* + *T* или в строке поиска программ введите *Terminal*
2. Введите команду установки ZeroTier.

```
curl -s https://install.zerotier.com | sudo bash
```

Подключение к сети

1. Откройте консоль.
2. Введите команду `sudo zerotier-cli join network-id` , где `network-id` это ID вашей сети.

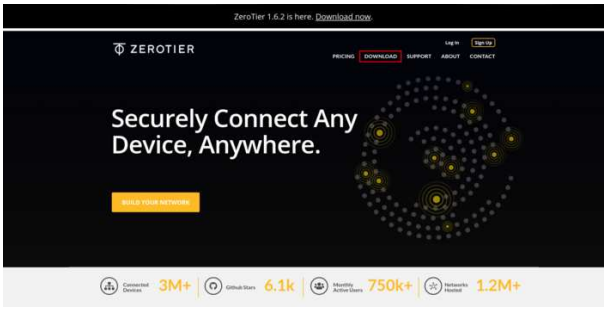


3. При успешном подключении, в консоль будет выведено соответствующее сообщение.

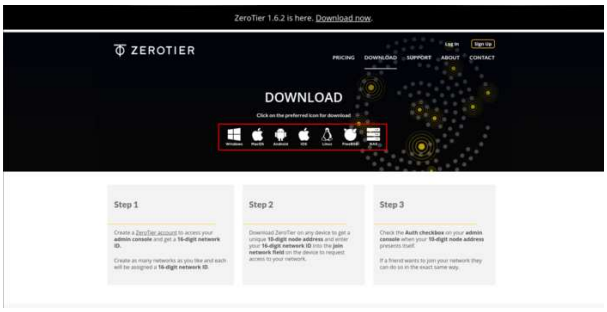
Установка и настройка на macOS

Установка приложения

1. Перейдите на сайт ZeroTier.



2. Нажмите на иконку macOS.



3. Скачайте и запустите файл `ZeroTier One.pkg`.
4. Установите приложение ZeroTier One.

Подключение к сети

1. Запустите приложение ZeroTier One.
2. В панели задач нажмите на иконку ZeroTier One.
3. В открывшемся окне нажмите *Join Network...*.

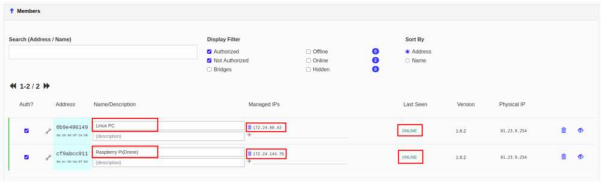


4. В поле *Enter Network ID* введите ID вашей сети.



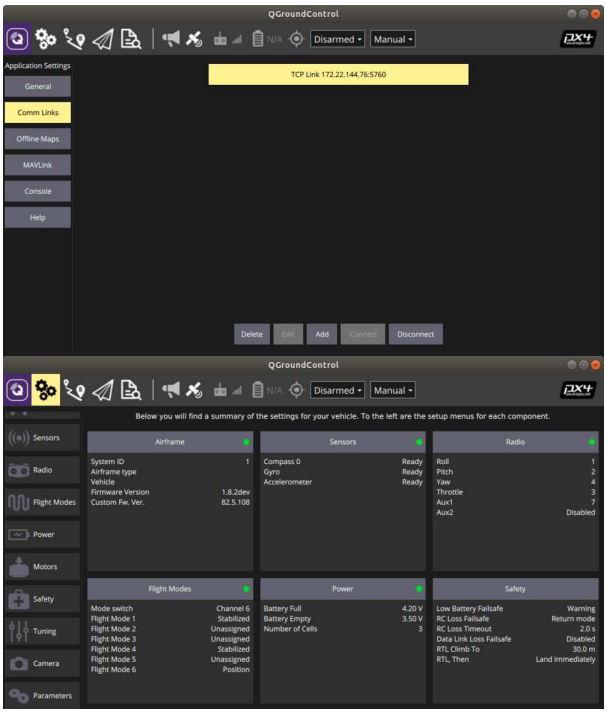
Подключение к коптеру

1. Убедитесь, что ZeroTier работает и имеет соединение с сетью на дроне и управляющем устройстве. Для этого убедитесь, что интересующие вас устройства имеют статус *Online*.



2. Убедитесь, что у всех устройств есть локальные IP адреса - *Managed IPs*.

3. Откройте GQC и во вкладке *Comm Links* добавьте TCP подключение, указав IP дрона. Подробнее про удаленное подключение читайте тут.



Быстрое подключение к виртуальной сети

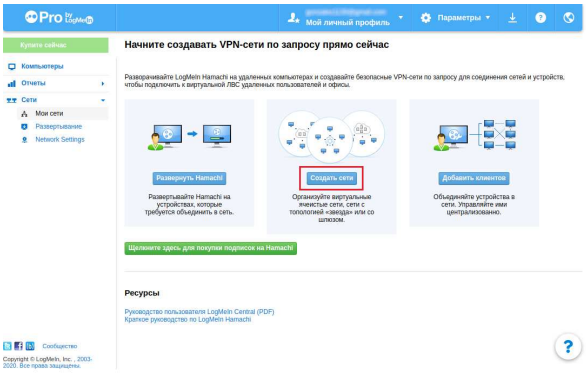
Для удаленного подключения и управления дроном или получения видеоизображения можно подключить его к виртуальной сети VPN.

Вы можете подключить вашу систему к любой доступной вам сети, если у вас есть соответствующие правда доступа. В этой статье будет рассмотрен способ подключения к сети *LogMeIn*, как удобной и легкой в использовании.

Создание виртуальной сети

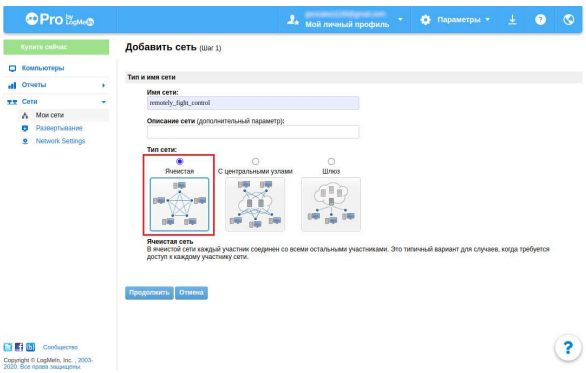
Создайте аккаунт и войдите на сайте LogMeIn.

После входа вы увидите основное меню управления вашими сетями.

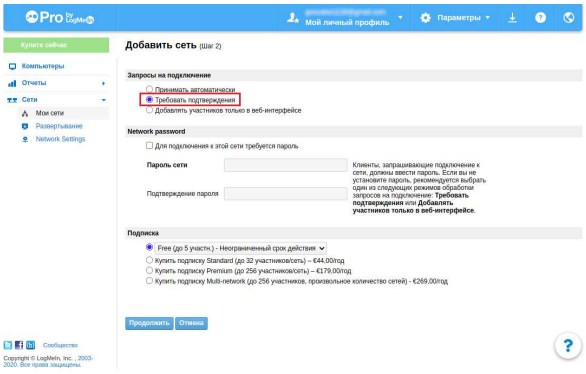


Выберите пункт *Создать сети*.

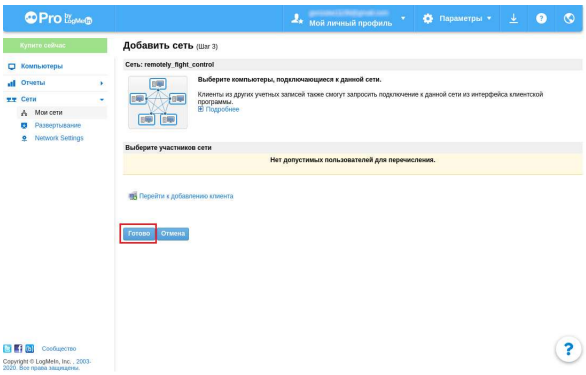
В открывшемся окне введите название сети и выберите тип *Ячеистая*.



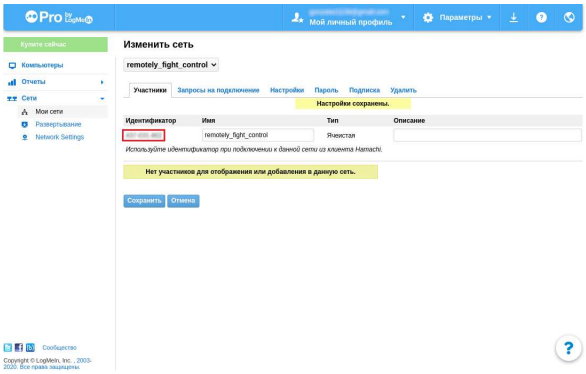
Далее меню *Запросы на подключение* выберите *Требовать подключение*.



Нажмите *Готово* и перейдите к настройки сети.



В открывшемся окне *Изменить сеть* необходимо запомнить значение поля *Идентификатор сети*, он будет использоваться в дальнейшем для подключения.



Установка менеджера Hamachi и подключение к сети

1. Скачайте Debian-пакет `logmein_hamachi`.

```
wget https://www.vpn.net/installers/logmein-hamachi_2.1.0.203-1_i386.deb
```

2. Установите пакет.

```
sudo dpkg -i logmein-hamachi_2.1.0.203-1_i386.deb
```

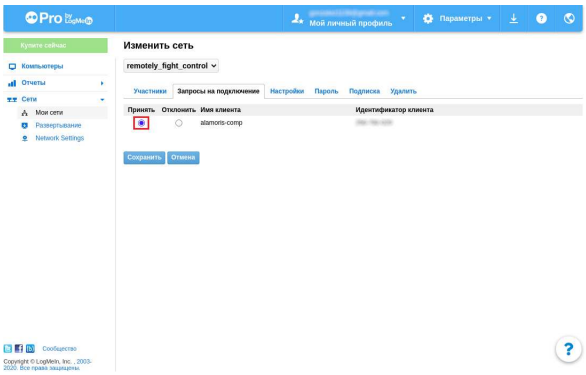
3. Подключите установленный модуль к сети.

```
hamachi login
```

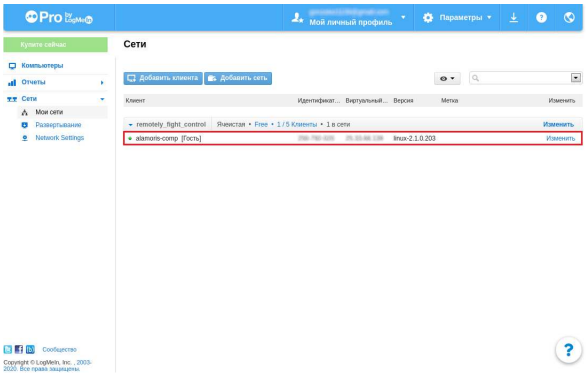
4. Подключитесь к сети используя ее идентификатор.

```
sudo hamachi do-join xxx-xxx-xxx
```

5. В поле ввода пароля нажмите *Enter*, если пароль не задан или введите его.
6. При успешном подключении вы увидите сообщение: *Joining 435-995-378 .. ok, request sent, waiting for approval*.
7. Подтвердите подключение к сети в меню *Изменить сеть*, во вкладке *Запросы на подключение*.



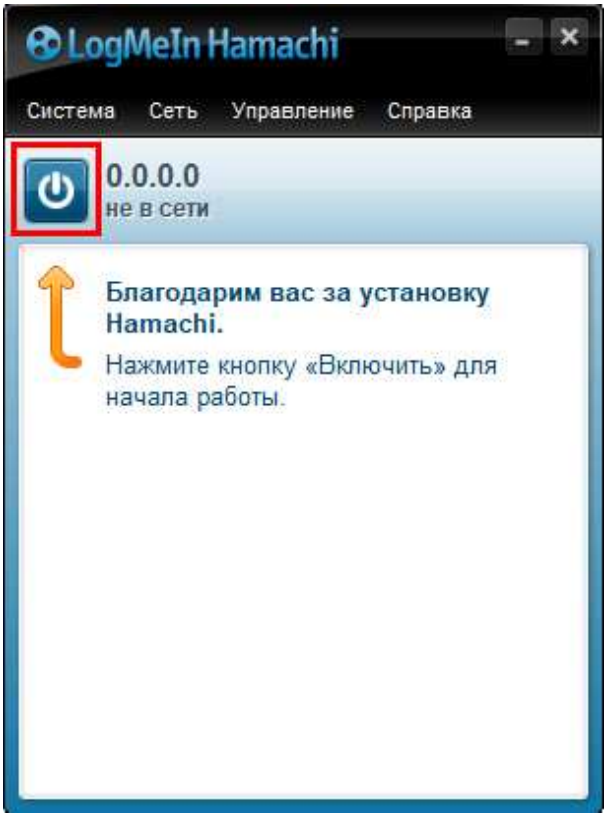
8. Можно проверить, удалось ли подключить пользователя в окне *Мои сети*.



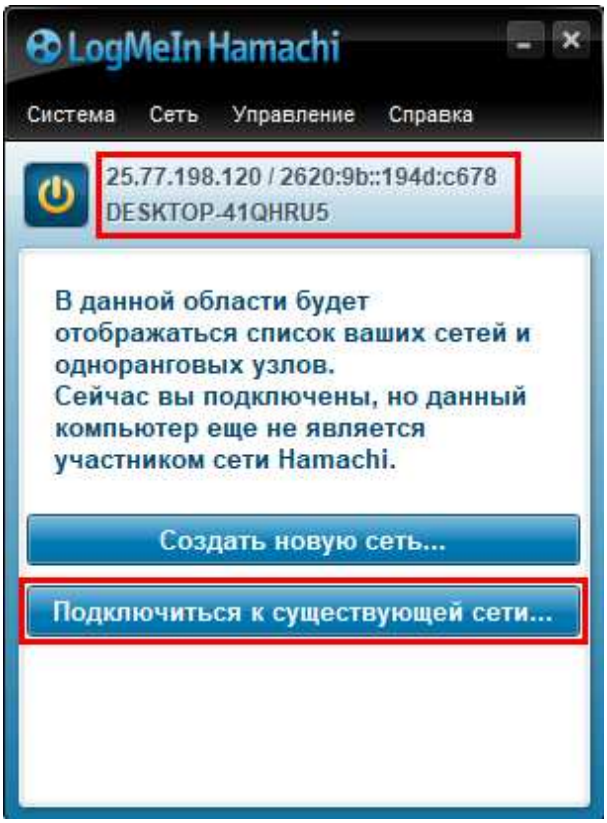
9. Повторите шаги 4–7 для подключения компьютера в случае, если вы пользуетесь операционной системой Linux, или обратитесь к инструкции для Windows.

Подключение к сети с помощью Windows

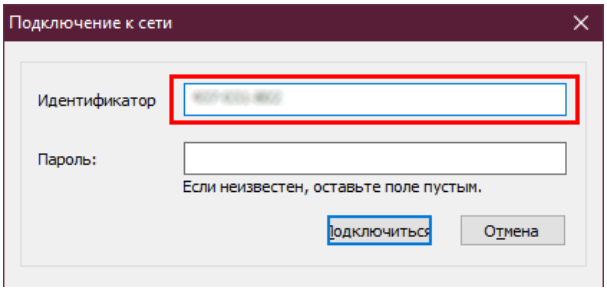
1. Установите приложение Hamachi.



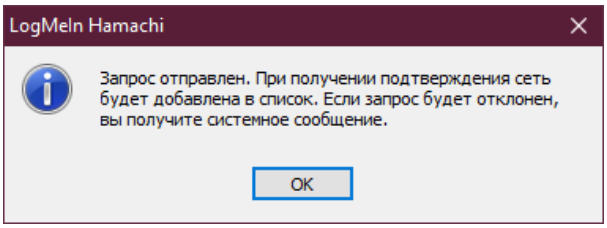
2. Запустите приложение Hamachi и нажмите кнопку включения. При необходимости введите свой логин.



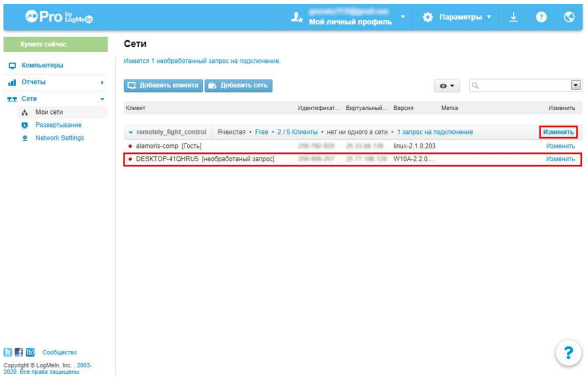
3. Если в шапке приложения отображается ваш виртуальный IP-адрес, подключитесь к сети, нажав соответствующую кнопку.



4. Введите идентификатор вашей сети.

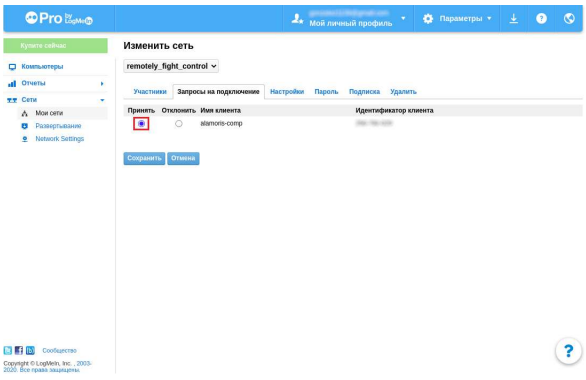


5. Если идентификатор введен верно, вы увидите соответствующее сообщение.



6. В меню вашей сети появится пользователь с префиксом "необработанный запрос".

7. Для подтверждения пользователя зайдите в меню *Изменить сеть* и во вкладке *Запросы на подключение* подтвердите подключение к сети.



См. также продолжение настройки для удаленного управления коптером и настройки стрима видео.

Управление при помощи 4G связи

Мобильная связь четвертого поколения удобный инструмент для передачи и получения информации на большой скорости. В настоящее время область покрытия мобильных операторов позволяет подключаться к интернету на большой скорости практически из любой точки.

> Для передачи каких либо данных с вашего коптера на наземную станцию(QGroundControl) и обратно, вам необходимо настроить собственную VPN сеть.

Подключение 4G модема к Raspberry Pi

Подключите 4G модем с сим-картой в USB порт вашей Raspberry Pi.

Обратите внимание, что при подключении, некоторые модемы распознаются в системе как сетевая карта, без каких либо дополнительных настроек.

Пример 4g модема: *USB 4G Huawei E3372H*



Однако, некоторые модемы, например, *Quectel EP06*, автоматически не стартуют соединение. В этом случае необходимо воспользоваться утилитами `qmi-network` и `udhcpc`. Установить эти утилиты можно командой:

```
sudo apt install libqmi-utils udhcpc
```

Далее запустить соединение с интернетом следующими командами:

```
sudo ip link set wwan0 down
echo 'Y' | sudo tee /sys/class/net/wwan0/qmi/raw_ip
sudo ip link set wwan0 up
sudo qmi-network /dev/cdc-wdm0 start
sudo udhcpc -q -f -i wwan0
```

Более подробно прочитать про настройку сетевого подключения вы можете в этой статье.

Проверить наличие соединения с интернетом можно командой:

```
ping -I wwan0 -c 5 8.8.8.8
```

Проверить скорость соединения с интернетом можно утилитой `speedtest`:

```
sudo apt install speedtest-cli
speedtest
```

Подключение Raspberry Pi к VPN

Сформируйте необходимые ключи VPN сети, для подключения Raspberry Pi и наземной станции.

Для того, чтобы подключить Raspberry Pi к вашей сети, установите пакет `openvpn`:

```
sudo apt-get install openvpn
```

Перенесите ваши ключи в директорию `/etc/openvpn/client`. Для удобства используйте графический SFTP интерфейс передачи данных, к примеру: WinSCP, FileZilla и т.д.

Для включения режима клиента, необходимо активировать переданные вами ключи. Ключи могу быть сформированы в различных форматах, к примеру: `.ovpn`, `.conf`. Ключ или конфигурация использующийся на вашем коптере, должны быть строго в формате `.conf`.

Инициализируйте сервис применяющий ваши ключи для подключения в режиме клиента:

```
sudo systemctl enable openvpn-client@config-name
```

где `config-name` - название вашего конфигурационного файла.

Если все сделано правильно, при каждом перезапуске системы, сервис-клиент будет автоматически подключаться к вашей сети.

> Перед началом работы не забудьте настроить и включить VPN подключение на вашем ПК.

> В качестве удобной альтернативы вы можете воспользоваться VPN-сервисом ZeroTier.

Управление коптером через QGroundControl

> Для управления коптером предлагается использовать UDP протокол передачи, обеспечивающий меньшую задержку, ценой отсутствия гарантии получения пакета, что очень важно во время пилотирования коптера.

Убедитесь, что ваш коптер и наземная станция подключены к вашей сети.

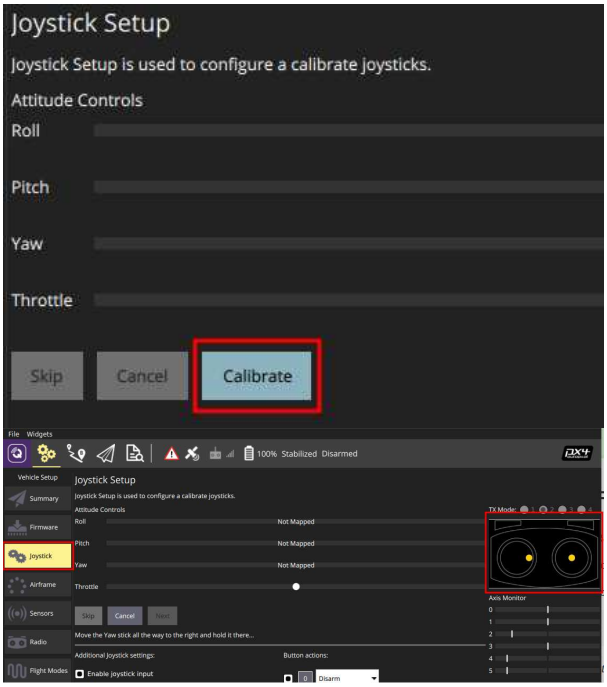
Для этого можете воспользоваться командой `ip addr`. Результатом ее выхода будет пронумерованный список активных сетей включенных на вашем устройстве. Обратить внимание стоит на подключение с префиксом *tun* и указанным вами IP адресом, если оно присутствует в вашем списке, ваш коптер подключился к сети.

В QGroundControl, аналогично подключению по Wi-Fi, настройте подключение к вашему коптеру по протоколу использующемуся в вашей сети: *UDP/TCP*. Рекомендуется использовать *UDP* подключение, за счет большей скорости передачи данных.

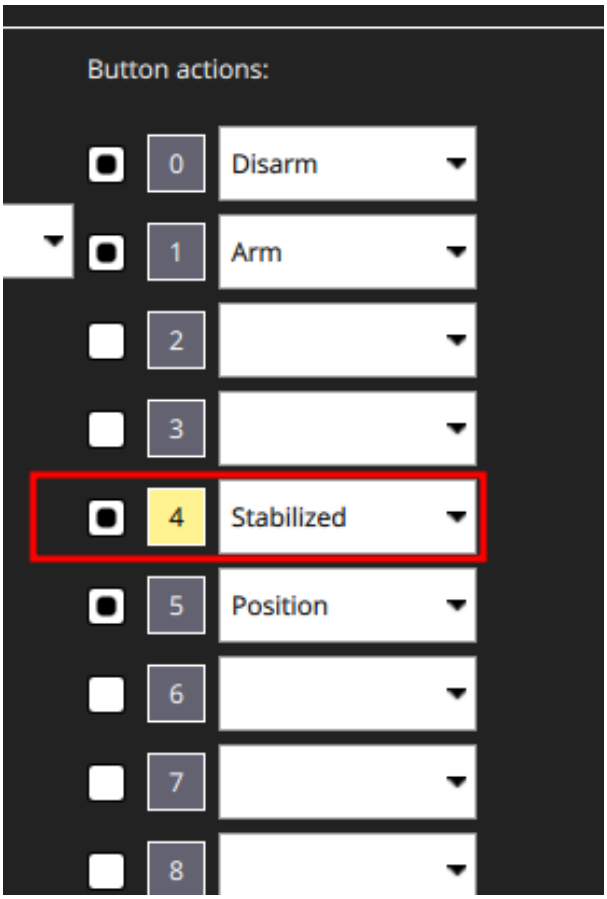
Если у вас появилась связь с коптером, подключите какой-либо джойстик к вашему ПК. Роль джойстика может выполнять как радио пульты, такие как, FlySky-i6X, Taranis x7 и т.д., так и джойстики от приставок или любые их эмуляции, которые распознаются системой.

Когда джойстик распознается системой, в колонке *Vehicle Setup* появится пункт *Joystick*, в случае, если он подсвечивается красным цветом, это значит, что требуется настройка.

Для калибровки джойстика, во вкладке *Joystick* нажмите кнопку *Calibrate* и следуйте инструкциям положения стиков пульта, указанных с левой стороны окна.



После успешной калибровки необходимо настроить полетные режимы. Чтобы это сделать несколько раз переключите необходимые тумблеры. В ходе переключения, вы увидите виртуальные каналы, на которых работают тумблеры. В активном положении будет подсвечиваться один из каналов.

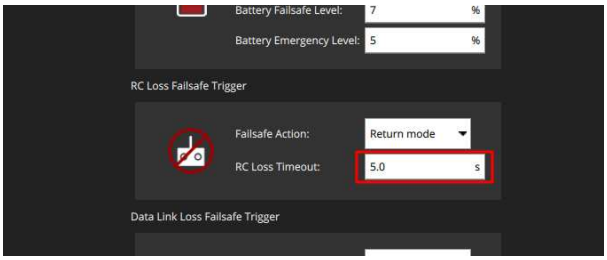


> При выборе джойстика, обратите внимание на количество рабочих каналов и на поддержку его, в QGroundControl(SDL2). Встречаются пульты поддерживающие всего 4 канала, что не удобно для такого типа управления.

Если изменения положения стиков отображается в окне QGroundControl, вам остается только применить параметр, определяющий, что управление коптером происходит с помощью джойстика, а не радиоаппаратуры:

`COM_RC_IN_MODE` - Joystick/No RC Checks

Поскольку мобильная связь не всегда бывает стабильна, рекомендуется увеличить таймаут на потерю сигнала управления до 5 секунд.



Коптер готов к полету!

> Если коптер не армится при переведении левого стика в нижний правый угол, установите состояние Arm/Disarm на один из тумблеров.

Передача видео с камеры в QGroundControl

Передача видео возможна практически с любой камеры подключенной к вашей Raspberry Pi. Для этого вам потребуется установить или собрать пакет *gst-rtsp-launch*:

```
sudo apt update
sudo apt install gst-rtsp-launch
```

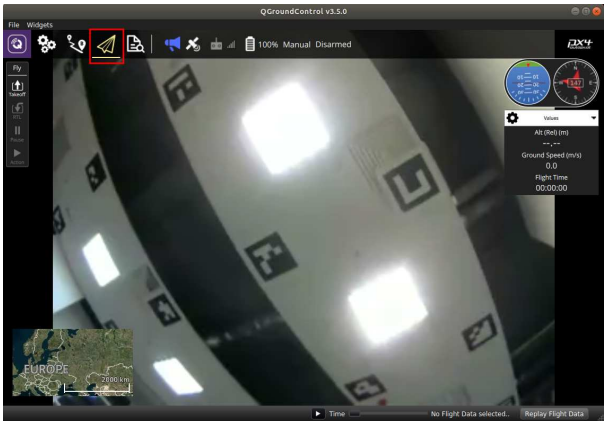
Чтобы запустить передачу изображений, необходимо ввести соответствующую командную строку:

```
gst-rtsp-launch "( v4l2src device=/dev/video0 ! video/x-raw,framerate=30/1,widt
```

Данная командная строка содержит параметры передачи видео, такие как: устройство передачи, частота кадров, высота/ширина изображения, метод кодирования и т.д. Подробнее о настройках можно узнать тут.

> Камера Raspberry Pi `/dev/video0` может использоваться сервисом `clover` и тогда `gst-rtsp-launch` не сможет получить к ней доступ. Чтобы остановить сервис `clover` выполните команду `sudo systemctl stop clover`. Также можно пустить трансляцию с USB-камеры, для этого замените устройство передачи на `/dev/video1`.

В приложении QGroundControl проверьте, что по ссылке `rtsp://192.168.11.1:8554/video` (IP-адрес вашей Raspberry Pi может отличаться) начался стрим изображения.



Автоматизация запуска передачи видео

Создайте файл и добавьте в него командную строку:

```
nano script_name.sh
```

Чтобы корректно запускать файл, необходимо присвоить ему соответствующие флаги доступа.

```
chmod a+x script_name.sh
```

Для того, чтобы передача видео запускалась каждый раз при включении системы, необходимо создать стартап-скрипт с помощью менеджера systemd. В директории `/etc/systemd/system` создайте файл `qgc_video.service`.

```
sudo nano /etc/systemd/system/qgc_video.service
```

В файл запишите соответствующие строки:

```
[Unit]
Description=VideoStream

[Service]
ExecStart=/bin/bash /home/pi/script_name.sh

[Install]
WantedBy=multi-user.target
```

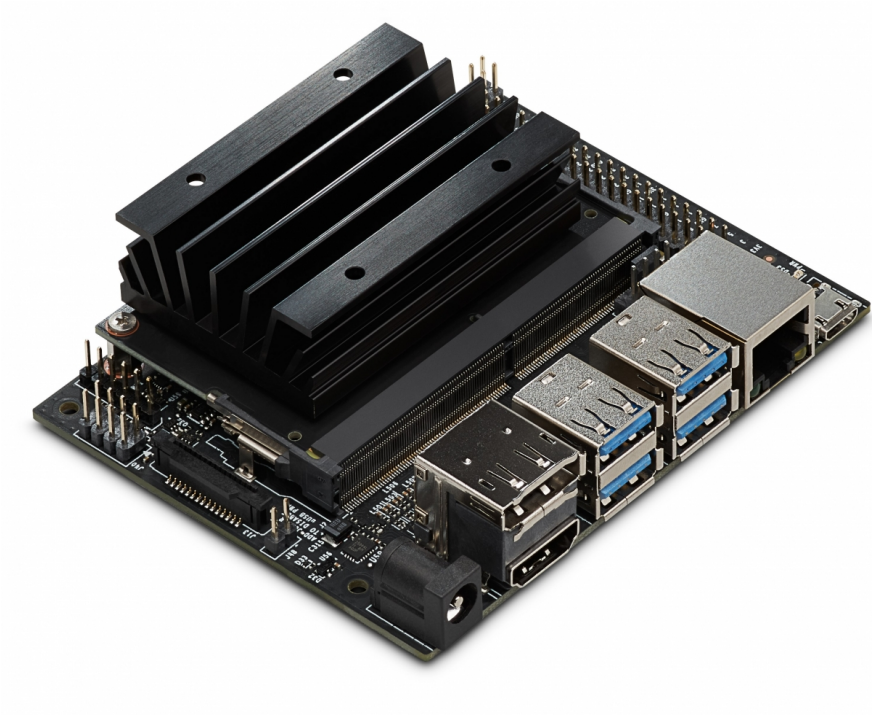
Осталось только инициализировать ваш скрипт в системе и он будет запускаться при каждом ее включении.

```
sudo systemctl enable qgc_video.service
```

Пакеты Клевера на Jetson Nano

О Jetson Nano

Jetson Nano – система на модуле (SoM), выпускаемая компанией Nvidia. Система построена на базе платформы Tegra X1 и несёт на себе четырёхядерный процессор ARM Cortex-A57 частотой 1.4 ГГц, 4 ГБ оперативной памяти и видеоядро на базе Nvidia Maxwell.



Набор для разработчиков Jetson Nano включает в себя как сам модуль, так и плату-носитель с портами USB 3.0, CSI, Ethernet и GPIO-пинами. Плата-носитель ненамного больше одноплатного компьютера Raspberry Pi и может быть использована в качестве бортового компьютера.

> На плате-носителе изначально нет Wi-Fi-адаптера. Для организации беспроводного подключения к Jetson Nano следует использовать USB-адаптер или Wi-Fi карту стандарта M.2. Следует свериться со списком совместимого оборудования перед установкой адаптера.

Установка ПО

Nvidia предоставляет образ системы для Jetson Nano на базе Ubuntu 18.04. Эта версия системы поддерживается как база для ROS Melodic.

Начальная установка

> Более подробные инструкции можно получить на официальном сайте Nvidia для Jetson Nano.

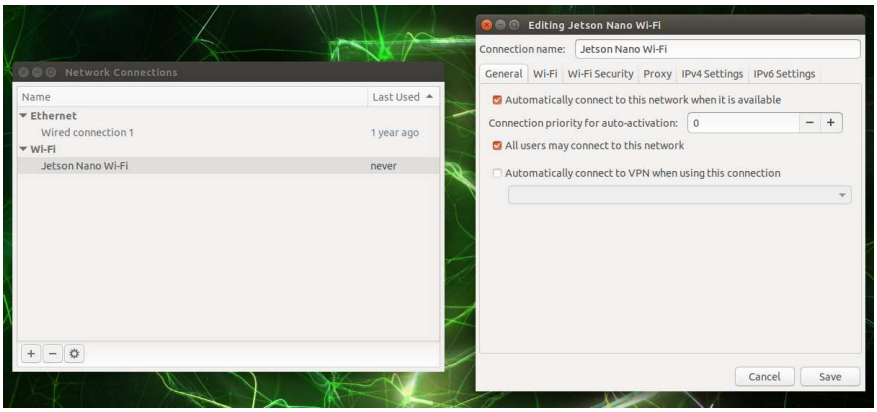
Для начальной установки требуется использование клавиатуры, мыши и HDMI-монитора. Скачайте образ системы Jetson Nano и запишите его на карту microSD (размером не менее 16 ГБ; рекомендуется использовать карту объёмом не менее 32 ГБ). Вставьте записанную карту в модуль Jetson Nano, подключите клавиатуру, мышь и монитор к плате-носителю и подайте питание на модуль.

> Jetson Nano можно питать от кабеля microUSB, но для повышения надёжности рекомендуется использовать специальный разъём питания. При использовании этого разъёма следует поместить перемычку на пины J48 (расположены рядом с разъёмом CSI).

Примите лицензионное соглашение на использование системы и следуйте дальнейшим указаниям установщика. После установки система автоматически перезагрузится и покажет экран входа в систему. Выберите созданного вами пользователя и введите пароль, указанный на этапе установки.

> Настоятельно рекомендуется выбрать английский язык как системный, дабы избежать возможных проблем с ROS.

Рекомендуется настроить автоматические подключение к Wi-Fi сети. Для этого после установки системы нажмите на значок Wi-Fi подключения в верхней части экрана, выберите опцию "Edit Connections..." в выпадающем меню, выделите текущую Wi-Fi сеть в открывшемся списке и нажмите на кнопку с иконкой шестерёнки.



Во вкладке "General" поставьте галочку возле пункта "All users may connect to this network". Нажмите на кнопку "Save" для применения параметров и закрытия окна.

> Убедитесь, что Jetson Nano доступен в сети. В образе по умолчанию установлен и включен SSH-сервер; для выполнения дальнейших операций рекомендуется подключиться к нему.

Установка ROS

> Ubuntu 18.04 официально поддерживается дистрибутивом ROS Melodic, и поэтому на официальном сайте есть подробная инструкция по его установке.

Добавьте ключи и репозитории OSRF в систему:

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E3
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) ma
sudo apt update
```

Установите базовые ROS-пакеты (стек `ros-base`):

```
sudo apt install ros-melodic-ros-base
```

Активируйте окружение ROS и обновите кэш утилиты `rosdep`:

```
source /opt/ros/melodic/setup.bash
sudo rosdep init
rosdep update
```

> Добавьте строчку `source /opt/ros/melodic/setup.bash` в конец файла `.profile` в своей домашней директории, чтобы не производить активацию окружения ROS каждый раз заново.

Установите пакетный менеджер pip для Python 2 (он требуется для установки некоторых зависимостей):

```
sudo apt install curl
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
sudo python ./get-pip.py
```

Сборка нод Клевера

Создайте в своей домашней директории "рабочее пространство" (workspace) и добавьте туда необходимые для Клевера пакеты:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
git clone https://github.com/CopterExpress/clover
git clone https://github.com/CopterExpress/ros_led
git clone https://github.com/okalachev/vl53l1x_ros
```

Установите зависимости этих пакетов с помощью утилиты `rosdep`:

```
cd ~/catkin_ws
rosdep install --from-paths src --ignore-src -y
```

Для функционирования mavros потребуется также скачать географические данные. Это делается следующими командами:

```
curl https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/ins
chmod a+x ./install_geographiclib_datasets.sh
sudo ./install_geographiclib_datasets.sh
```

Для сборки пакетов также потребуется установить заголовочные файлы OpenCV 3.2 (по умолчанию в новых образах для Jetson Nano установлены заголовочные файлы для OpenCV 4.1.1; с ними компиляция пакетов провалится):

```
sudo apt install libopencv-dev=3.2.0+dfsg-4ubuntu0.1
```

Наконец, запустите сборку нод Клевера:

```
cd ~/catkin_ws
catkin_make
```

> При подключении полётных контроллеров на базе PX4 через USB следует также добавить правила udev в систему. Скопируйте файл с правилами в `/etc/udev/rules.d` и выполните команду `sudo udevadm control --reload-rules && sudo udevadm trigger`.

Запуск Клеверных нод

Активируйте окружение "рабочего пространства":

```
cd ~/catkin_ws
source devel/setup.bash
```

Поменяйте launch-файлы так, чтобы это соответствовало вашей конфигурации, и запустите ноды с помощью `roslaunch`:

```
roslaunch clover clover.launch
```

> Вы можете настроить `systemd` так, чтобы ноды Клевера запускались автоматически. Примером такой настройки может служить образ Клевера для Raspberry Pi: там созданы сервисы `roscore` и `clover`. Их можно подкорректировать и использовать в Jetson Nano.

Возможные проблемы

CSI-камеры

Jetson Nano не поддерживает старые камеры для Raspberry Pi (v1, на базе сенсора Omnivision OV5647). Камеры Raspberry Pi v2 (на базе Sony IMX219) поддерживаются, но не показываются в виде Video4Linux-устройств.

Изображения с этих камер можно захватывать с помощью GStreamer. Для последующей передачи этих изображений в ROS можно использовать ноды `gscam` или `jetson_camera`. Для запуска ноды `jetson_camera` потребуется собрать OpenCV из ветки 3.4 с поддержкой GStreamer.

Примеры конвейеров GStreamer для захвата изображения доступны в репозитории JetsonHacksNano.

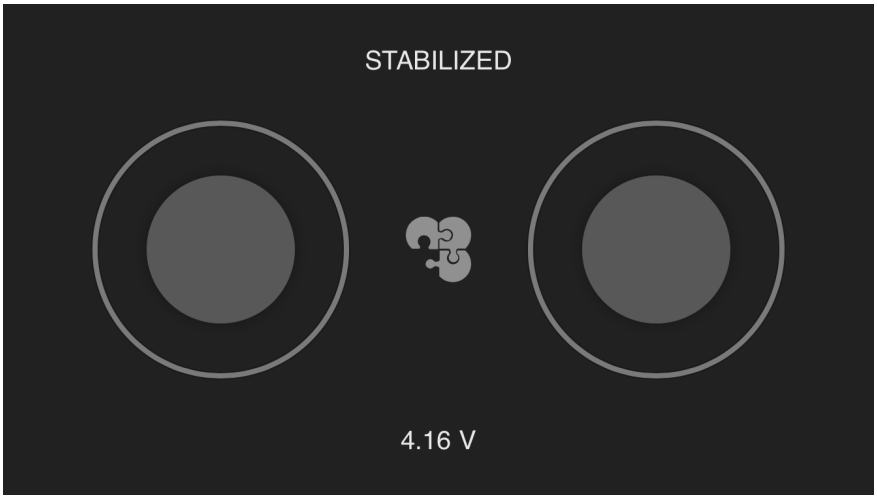
Изображение с камеры может становиться более красным по краям. Это можно исправить с помощью настройки процессора изображения. Эта процедура должна выполняться производителями камер; вот пример файла настройки процессора изображения от компании Arducam.

LED-лента

В данный момент Jetson Nano не поддерживает работу с LED-лентой через GPIO.

Управление Клевером со смартфона

Для управления Клевером со смартфона через Wi-Fi необходимо установить приложение – iOS, Android.



> Мобильный пульт предназначен в первую очередь для полетов в помещении на дальность не более 10-15 м. Большое количество Wi-Fi сетей также может ухудшить отзывчивость и дальность пульта.

Также управление со смартфона доступно в мобильной версия приложения QGroundControl.

Настройка

> Открытое соединение QGroundControl или rviz пересылает большие объемы данных по Wi-Fi, что может негативно сказаться на отзывчивости мобильного пульта. Рекомендуется не использовать эти приложения одновременно с ним.

Установите образ Clover на RPi. Для работы приложения параметры `rosbridge` и `rc` в launch-файле (`~/catkin_ws/src/clover/clover/launch/clover.launch`) должны быть включены:

```
<arg name="rosbridge" default="true"/>
```

```
<arg name="rc" default="true"/>
```

При изменении launch-файла необходимо перезапустить пакет `clover` :

```
sudo systemctl restart clover
```

Также необходимо убедиться, что PX4-параметр `COM_RC_IN_MODE` установлен в значение `0` (RC Transmitter).

Дополнительные параметры PX4:

- `COM_RC_LOSS_T` – таймаут для определения потери сигнала пульта (мобильного или физического). Рекомендуется увеличение таймаута до нескольких секунд.
- `NAV_RCL_ACT` – действие при потере сигнала пульта.

> Мобильный пульт конфликтует с реальной аппаратурой радиоуправления. Во время использования мобильного пульта она должна быть выключена.

Подключение

Подключите смартфон к Wi-Fi сети Клевера (`clover-xxxx`). Приложение должно подключиться с коптеру автоматически. При успешном подключении должны отобразиться текущий режим и заряд батареи.

Стики на экране приложения работают так же, как и реальные стики. Для арма коптера подержите левый стик в правом нижнем углу на протяжении нескольких секунд. Для дизарма – в левом нижнем углу.

Неисправности

- Если интерфейс пульта отображает явно неправильное напряжение (напр. > 5 V), проверьте, что значение PX4-параметра `BAT_N_CELLS` соответствует реальному количеству элементов батареи. Если отображаемое напряжение все равно неверно, откалибруйте датчик питания.
- Если вместо режима PX4 отображается текст "DISCONNECTED FROM FCU", проверьте подключение Raspberry Pi к Pixhawk.

Настройка Wi-Fi

Wi-Fi адаптер на Raspberry Pi имеет два основных режима работы:

1. **Режим клиента** – RPi подключается к существующей Wi-Fi сети.
2. **Режим точки доступа** – RPi создает Wi-Fi сеть, к которой вы можете подключиться.

При использовании образа для RPi по умолчанию Wi-Fi адаптер работает в режиме точки доступа.

Изменение пароля или SSID (имени сети)

1. Отредактируйте файл `/etc/wpa_supplicant/wpa_supplicant.conf` (используя SSH-соединение):

   ```
   sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
   ```

   Измените значение параметра `ssid`, чтобы изменить название Wi-Fi сети, и параметра `psk`, чтобы изменить пароль. Например:

   ```
   network={
       ssid="my-super-ssid"
       psk="cloverwifi123"
       mode=2
       proto=RSN
       key_mgmt=WPA-PSK
       pairwise=CCMP
       group=CCMP
       auth_alg=OPEN
   }
   ```

2. Перезагрузите Raspberry Pi.

> Длина пароля для Wi-Fi сети должна быть **не менее** 8 символов.
>
> При некорректных настройках `wpa_supplicant.conf` Raspberry Pi перестанет раздавать Wi-Fi!

Переключение адаптера в режим клиента

1. Выключите службу `dnsmasq`.

   ```
   sudo systemctl stop dnsmasq
   sudo systemctl disable dnsmasq
   ```

2. Включите получение IP адреса на беспроводном интерфейсе DHCP клиентом. Для этого удалите из файла `/etc/dhcpcd.conf` строки:

```
interface wlan0
static ip_address=192.168.11.1/24
```

3. Настройте `wpa_supplicant` для подключения к существующей точке доступа. Для этого замените содержимое файла `/etc/wpa_supplicant/wpa_supplicant.conf` на:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
    ssid="SSID"
    psk="password"
}
```

где `SSID` – название сети, а `password` – пароль.

4. Перезапустите службу `dhcpcd`.

```
sudo systemctl restart dhcpcd
```

Переключение адаптера в режим точки доступа

1. Включите статический IP адрес на беспроводном интерфейсе. Для этого добавьте в файл `/etc/dhcpcd.conf` строки:

```
interface wlan0
static ip_address=192.168.11.1/24
```

2. Настроите `wpa_supplicant` на работу в режиме точки доступа. Для этого замените содержимое файла `/etc/wpa_supplicant/wpa_supplicant.conf` на:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
    ssid="clover-1234"
    psk="cloverwifi"
    mode=2
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP
    group=CCMP
    auth_alg=OPEN
}
```

где `clover-1234` – название сети, а `cloverwifi` – пароль.

3. Включите службу `dnsmasq`.

```
sudo systemctl enable dnsmasq
sudo systemctl start dnsmasq
```

4. Перезапустите службу `dhcpcd`.

```
sudo systemctl start dhcpcd
```

Ниже вы можете узнать больше о том, как устроена работа с сетью на RPi.

Устройство сети RPi

Работа сети на образе поддерживается двумя предустановленными службами:

- **networking** — служба включает все сетевые интерфейсы в момент запуска [5].
- **dhcpcd** — служба обеспечивает настройку адресации и маршрутизации на интерфейсах, полученных динамически или указанных в файле настроек статически.

Для работы в режиме роутера (точки доступа) RPi необходим DHCP сервер. Он служит для автоматической выдачи настроек текущей сети подключившимся клиентам. В роли такого сервера может выступать `isc-dhcp-server` или `dnsmasq`.

dhcpcd

Начиная с Raspbian Jessie настройки сети больше не задаются в файле `/etc/network/interfaces`. Теперь за выдачу адресации и настройку маршрутизации отвечает `dhcpcd` [4].

По умолчанию на всех интерфейсах включен dhcp-клиент. Настройки интерфейсов меняются в файле `/etc/dhcpcd.conf`. Для того, чтобы поднять точку доступа необходимо прописать статический ip-адрес. Для этого в конец файла необходимо добавить следующие строки:

```
interface wlan0
static ip_address=192.168.11.1/24
```

> Если интерфейс является беспроводным (wlan), то служба `dhcpcd` триггерит `wpa_supplicant` [13], который в свою очередь работает непосредственно с wifi-адаптером и переводит его в заданное состояние.

wpa_supplicant

wpa_supplicant – служба конфигурирует Wi-Fi адаптер. Служба `wpa_supplicant` работает не как самостоятельная (хотя как таковая существует), а запускается как дочерний процесс от `dhcpcd`.

Конфигурационный файл по умолчанию должен иметь путь `/etc/wpa_supplicant/wpa_supplicant.conf`. Пример конфигурационного файла:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
        ssid=\"my-clover\"
        psk=\"cloverwifi\"
        mode=2
        proto=RSN
        key_mgmt=WPA-PSK
        pairwise=CCMP
        group=CCMP
        auth_alg=OPEN
}
```

Внутри конфига указываются общие настройки `wpa_supplicant` и параметры для настройки адаптера. Также конфигурационный файл содержит секции `network` – основные настройки Wi-Fi сети такие как SSID сети, пароль, режим работы адаптера. Таких блоков может быть несколько, но используется первый рабочий. Например, если вы указали в первом блоке подключение к некоторой недоступной сети, то адаптер будет настроен следующей удачной секцией, если такая есть. Подробнее о синтаксисе `wpa_supplicant.conf` см. в man-странице Debian.

wpa_passphrase

`wpa_passphrase` – утилита для создания секции `network`.

```
wpa_passphrase SSID PASSWORD
```

После выполнения команды скопируйте полученную секцию в ваш конфигурационный файл. Можно удалить закомментированное поле `psk` и оставить только поле с хешем пароля, либо наоборот.

```
network={
    ssid="SSID"
    #psk="PASSWORD"
    psk=c2161655c6ba444d8df94cbbf4e9c5c4c61fc37702b9c66ed37aee1545a5a333
}
```

Несколько Wi-Fi адаптеров

В системе может быть несколько Wi-Fi адаптеров. Если для них корректно подключены драйвера, то их можно увидеть вызвав `ifconfig` (например `wlan0` и `wlan1`).

Если у вас несколько адаптеров, для всех будет использоваться одна и та же самая рабочая секция `network`. Это связано с тем, что для каждого интерфейса, `dhcpcd` отдельно создает по дочернему процессу `wpa_supplicant`, в котором выполняется один тот же код (т. к. конфиг один и тот же).

Для работы нескольких адаптеров с отдельными настройками для каждого, в стандартном вызываемом скрипте `dhcpcd` реализован механизм запуска разных конфигурационных скриптов. Для его использования необходимо переименовать стандартный файл конфига по следующему образцу: `wpa_supplicant-<имя интерфейса>.conf`, например `wpa_supplicant-wlan0.conf`.

Для применения настроек необходимо перезапустить родительский процесс – службу `dhcpcd`. Сделать это можно следующей командой:

```
sudo systemctl restart dhcpcd
```

DHCP сервер

dnsmasq-base

`dnsmasq-base` – консольная утилита, не являющаяся службой, для использования dnsmasq как службы надо установить пакет `dnsmasq`.

```
sudo apt install dnsmasq-base
```

```
# Вызов dnsmasq-base
sudo dnsmasq --interface=wlan0 --address=/clover/coex/192.168.11.1 --no-daemon

# Подробнее о dnsmasq-base
dnsmasq --help

# или
man dnsmasq
```

dnsmasq

```
sudo apt install dnsmasq
```

```
cat << EOF | sudo tee -a /etc/dnsmasq.conf
interface=wlan0
address=/clover/coex/192.168.11.1
dhcp-range=192.168.11.100,192.168.11.200,12h
no-hosts
filterwin2k
bogus-priv
domain-needed
quiet-dhcp6

EOF
```

isc-dhcp-server

```
sudo apt install isc-dhcp-server
```

```
# https://www.shellhacks.com/ru/sed-find-replace-string-in-file/
sed -i 's/INTERFACESv4=\"\"/INTERFACESv4=\"wlan0\"/' /etc/default/isc-dhcp-serv
```

```
cat << EOF | sudo tee /etc/dhcp/dhcpd.conf
subnet 192.168.11.0 netmask 255.255.255.0 {
  range 192.168.11.11 192.168.11.254;
  #option domain-name-servers 8.8.8.8;
  #option domain-name "rpi.local";
  option routers 192.168.11.1;
  option broadcast-address 192.168.11.255;
  default-lease-time 600;
  max-lease-time 7200;
}

EOF
```

```
cat << EOF | sudo tee /etc/network/if-up.d/isc-dhcp-server && sudo chmod +x /et
#!/bin/sh
if [ "\$IFACE" = "--all" ];
then sleep 10 && systemctl start isc-dhcp-server.service &
fi

EOF
```

Ссылки

1. habr.com: Linux WiFi из командной строки с wpa_supplicant
2. wiki.archlinux.org: WPA supplicant (Русский)
3. blog.hoxnox.com: WiFi access point with wpa_supplicant
4. dmitrysnotes.ru: Raspberry Pi 3. Присвоение статического IP-адреса
5. thegeekdiary.com: Linux OS Service ‘network’
6. frillip.com: Using your new Raspberry Pi 3 as a Wi-Fi access point with hostapt (также здесь есть инструкция по настройке форвардинга для использования RPi в качестве шлюза для выхода в интернет)
7. habr.com: Настраиваем ddns-сервер на GNU/Linux Debian 6 (Хорошая статья по настройке ddns-сервера на основе `bind` и `isc-dhcp-server`)
8. pro-gram.ru: Установка и настройка DHCP сервера на Ubuntu 16.04. (Настройка isc-dhcp-server)
9. expert-orda.ru: Настройка DHCP-сервера на Ubuntu (Настройка isc-dhcp-server)
10. academicfox.com: Raspberry Pi беспроводная точка доступа (WiFi access point) (Настройка маршрутов, hostapd, isc-dhcp-server)
11. weworkweplay.com: Automatically connect a Raspberry Pi to a Wifi network (Есть настройки для создания открытой точки доступа)
12. wiki.archlinux.org: WPA supplicant

13. wiki.archlinux.org: dhcpcd (dhcpcd hook wpa_supplicant)

Интерфейс UART

UART – последовательный асинхронный интерфейс для передачи данных, применяемый во многих устройствах. Например GPS-антенны, Wi-Fi роутеры или Pixhawk.

Интерфейс обычно содержит две линии: TX – линия для передачи данных, RX – линия для приёма данных. А также обычно использует 5-ти вольтовую логику.

Для соединения двух устройств необходимо линию TX первого устройства подать на RX второго. Аналогичную операцию нужно совершить с другой стороны, чтобы обеспечить двустороннюю передачу данных.

> Необходимо синхронизировать уровни напряжений – соединить землю на двух устройствах.

Почитать больше про интерфейс и протокол можно в этой статье.

Linux TTY

В Linux есть понятие Posix Terminal Interface (подробнее здесь). Это некоторая абстракция над последовательным или виртуальным интерфейсом, позволяющая работать с устройством нескольким агентам одновременно.

В качестве примера такой абстракции в Raspbian можно привести `/dev/tty1` – устройство вывода текста на экран подключенный по HDMI.

UART на Raspberry Pi 3

В Raspberry Pi 3 есть два аппаратных UART интерфейса:

1. `Mini UART` (/dev/ttyAMA0) – для своей работы использует тактирование видеоядра RPi, в связи с чем ограничивает его частоту.
2. `PL011` (/dev/ttyS0) – полноценный UART интерфейс выполненный на отдельном блоке кристалла микроконтроллера.

Подробнее про UART на Raspberry Pi в официальной статье.

Данные интерфейсы с помощью вентилей микроконтроллера можно переключать между двумя физическими выходами:

1. разъём UART на GPIO;
2. Bluetooth модуль RPi.

По умолчанию в Raspberry Pi 3 `PL011` подключен к Bluetooth модулю. А `Mini UART` отключен с помощью значения директивы `enable_uart`, по дефолту равной `0`.

> Надо понимать, что директива `enable_uart` меняет свое дефолтное значение исходя из того, какой UART подключен к Bluetooth модулю RPi с помощью директивы `dtoverlay=pi3-miniuart-bt`.

Для удобства работы с этими выходами в Raspbian существуют алиасы:

- `/dev/serial0` – всегда указывает на то TTY устройство, что подключено к GPIO портам.
- `/dev/serial1` – всегда указывает на то TTY устройство, что подключено к Bluetooth модулю.

Настройка UART на Raspberry Pi

Для настроек UART существуют директивы, которые находятся в `/boot/config.txt`.

Для включения UART интерфейса на GPIO:

```
enable_uart=1
```

Для отключения UART интерфейса от Bluetooth модуля:

```
dtoverlay=pi3-disable-bt
```

Для подключения `Mini UART` к Bluetooth модулю:

```
dtoverlay=pi3-miniuart-bt
```

В случае отключения Bluetooth модуля следует отключить `hciuart` сервис:

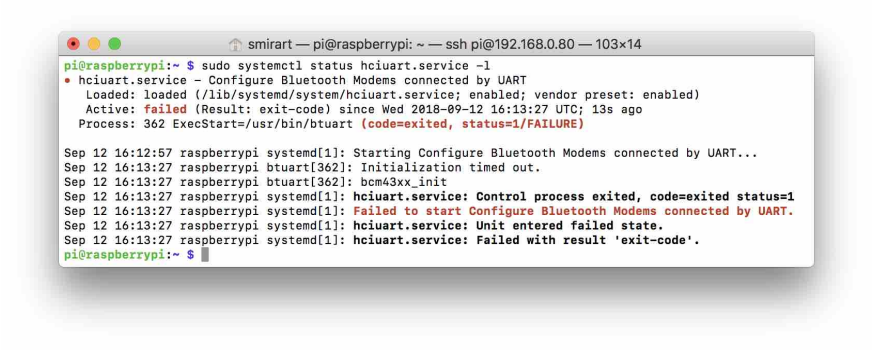
```
sudo systemctl disable hciuart.service
```

Настройка образа по умолчанию

На образе для RPi изначально выключен `Mini UART` и Bluetooth модуль.

Bugs

Если использовать подключение `Mini UART` к Bluetooth, `hciuart` падает с ошибкой:



В случае отключения Bluetooth

```
/dev/serial0 -> ttyAMA0
/dev/serial1 -> ttyS0
```

Параметры PX4

Полная документация по параметрам PX4: https://docs.px4.io/master/en/advanced_config/parameter_reference.html.

Для изменения параметров PX4 используйте программу QGroundControl, подключившись к Клеверу по Wi-Fi или USB. Перейдите в панель *Vehicle Setup* (кликнув на логотип QGroundControl в левом верхнем углу и выберите меню *Parameters*.

Рекомендованные значения

Общие параметры

Параметр	Значение	Примечание
`SENS_FLOW_ROT`	0 (*No rotation*)	В случае использования "железного" PX4Flow, оставьте значение по умолчанию
`SENS_FLOW_MINHGT`	0.0	Для дальномера VL53L1X
`SENS_FLOW_MAXHGT`	4.0	Для дальномера VL53L1X
`SENS_FLOW_MAXR`	10.0	
`SYS_HAS_MAG`	0	При невозможности запуска магнитометра (ошибка *No mags found*)

Настройки подсистемы Estimator

В случае использования LPE (прошивка COEX):

Параметр	Значение	Примечание
`LPE_FUSION`	86	Чекбоксы: *flow* + *vis* + *land detector* + *gyro comp*. При полете над ровным полом возможно включение *pub agl as lpos down*. Подробнее: Optical Flow, ArUco-маркеры, GPS.
`LPE_VIS_DELAY`	0.0	
`LPE_VIS_Z`	0.1	
`LPE_FLW_SCALE`	1.0	
`LPE_FLW_R`	0.2	
`LPE_FLW_RR`	0.0	
`LPE_FLW_QMIN`	10	
`ATT_W_EXT_HDG`	0.5	Включение использования внешнего угла по рысканью (при навигации по карте маркеров)
`ATT_EXT_HDG_M`	1 (*Vision*)	
`ATT_W_MAG`	0	Выключение магнитометра (при навигации внутри помещения)

В случае использования EKF2 (официальная прошивка):

Параметр	Значение	Примечание
`EKF2_AID_MASK` *	26	Чекбоксы: *flow* + *vision position* + *vision yaw*. Подробнее: Optical Flow, ArUco-маркеры, GPS.
`EKF2_OF_DELAY`	0	
`EKF2_OF_QMIN`	10	
`EKF2_OF_N_MIN`	0.05	
`EKF2_OF_N_MAX`	0.2	
`EKF2_HGT_MODE` *	3 (*Vision*)	При наличии дальномера и полете над ровным полом — 2 (*Range sensor*)
`EKF2_EVA_NOISE`	0.1 rad или 5 deg	
`EKF2_EVP_NOISE`	0.1	
`EKF2_EV_DELAY`	0	
`EKF2_MAG_TYPE`	5 (*None*)	Выключение магнитометра (при навигации внутри помещения)

* — начиная с версии PX4 1.14 помеченные звездочкой параметры заменены на следующие:

Параметр	Значение	Примечание
`EKF2_EV_CTRL`	11	Чекбоксы: *Horizontal position* + *Vertical position* + *Yaw*
`EKF2_GPS_CTRL`	0	Все чекбоксы сняты
`EKF2_BARO_CTRL`	0 (*Disabled*)	Барометр отключен
`EKF2_OF_CTRL`	1 (*Enabled*)	Optical flow включен
`EKF2_HGT_REF`	3 (*Vision*)	При наличии дальномера и полете над ровным полом — 2 (*Range sensor*)
`EKF2_RNG_CTRL`	2 (*Enabled*)	Дальномер включен

См. также: список параметров по умолчанию в симуляторе: https://github.com/CopterExpress/clover/blob/master/clover_simulation/airframes/4500_clover.

Дополнительная информация

Параметр `SYS_MC_EST_GROUP` отвечает за выбор Estimator'a.

Estimator это подсистема, которая вычисляет текущее состояние (state) коптера, используя показания с датчиков. В состояние коптера входит:

- угловая скорость коптера – roll_rate, pitch_rate, yaw_rate;
- ориентация коптера (в локальной системе координат) – roll (крен), pitch (тангаж), yaw (рысканье) (одно из представлений);
- позиция коптера (в локальной системе координат) – x, y, z;
- скорость коптера (в локальной системе координат) – vx, vy, vz;
- глобальные координаты коптера – latitude, longitude, altitude;
- высота над поверхностью;
- другие параметры (дрейф гироскопов, скорость ветра и пр.).

`SYS_AUTOCONFIG` – сброс всех параметров (при установке в значение 1).

EKF2

`EKF2_AID_MASK` – выбор датчиков, которые используются EKF2 для вычисления состояния коптера.

`EKF2_HGT_MODE` – основной источник данных о высоте (z в локальной системе координат):

- 0 – давление с барометра.
- 1 – GPS.
- 2 – дальномер (например, vl53l1x).
- 3 – данные с VPE.

Вариант 2 является наиболее точным, но его корректно использовать, только если поверхность, над которой летает коптер – плоская. В противном случае начало координат по Z будет двигаться вверх и вниз с изменением высоты поверхности.

Multicopter Position Control (полет по позиции)

Данные параметры настраивают полет коптера по позиции (режимы POSCTL, OFFBOARD, AUTO).

`MPC_THR_HOVER` – газ висения. Данный параметр необходимо установить на примерный процент газа, необходимый для того, чтобы коптер удерживал высоту. Если коптер имеет тенденцию набирать или терять высоту в режиме удержания высоты – можно уменьшить или увеличить это значение.

`MPC_XY_P` – коэффициент *P* регулятора по позиции. Этот параметр влияет на то, насколько резко коптер будет выполнять заданные команды по позиции. Слишком большое значение может вызвать перестрелы.

`MPC_XY_VEL_P` – коэффициент *P* регулятора по скорости. Данный параметр также влияет на точность и резкость выполнения коптером заданной позиции. При слишком большом значении возможны перестрелы.

`MPC_XY_VEL_MAX` – максимальная горизонтальная скорость в режимах POSCTL, OFFBOARD, AUTO.

`MPC_Z_P` , `MPC_Z_VEL_P` – коэффициенты *P* регуляторов по вертикальной позиции и скорости. Влияют на удерживание коптером необходимой высоты.

`MPC_LAND_SPEED` – вертикальная скорость посадки в режиме LAND.

LPE + Q attitude estimator

Данные параметры настраивают поведение модулей `lpe` и `q` , которые вычисляют состояние (ориентацию, позицию) коптера. Эти параметры применяются **только** если параметр `SYS_MC_EST_GROUP` установлен в значение `1` (local_position_estimator, attitude_estimator_q).

Commander

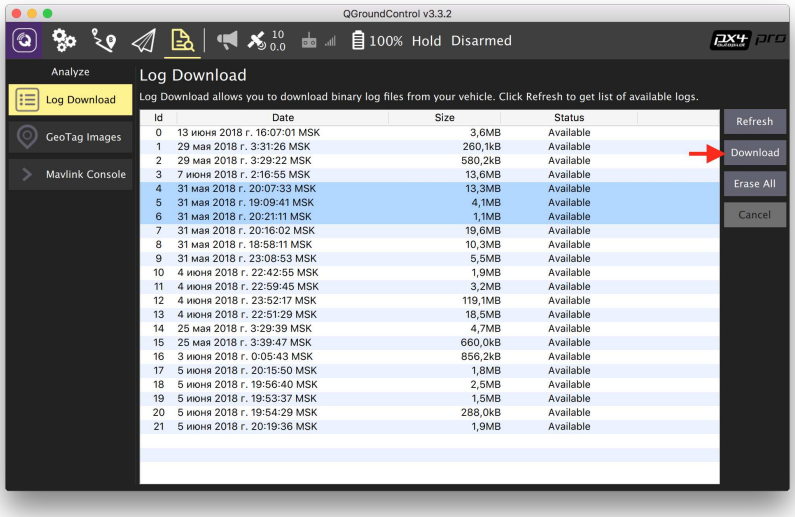
Преарм-чеки, переключение режимов и состояний коптера.

Sensors

Включение, выключение и настройка различных датчиков.

Логи и топики PX4

Для детального анализа поведения прошивки PX4 можно просмотреть полетные логи. Полетные логи представляют собой сообщения в uORB-топиках, записанные в файл с расширением `.ulg`. Лог-файл можно скачать с помощью QGroundControl по Wi-Fi или USB во вкладке *Log Download*:



Также необходимые `.ulg`-файлы можно скопировать непосредственно с MicroSD-карты, находившейся в полетном контроллере.

Анализ

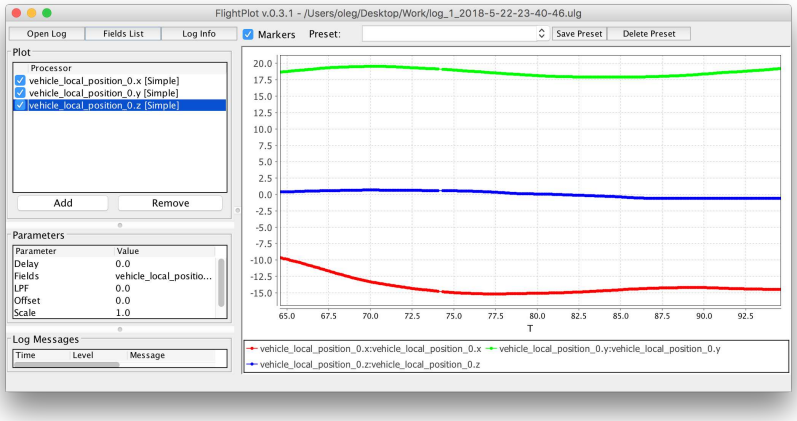
logs.px4.io

Записанные лог-файлы можно загрузить на сайт https://logs.px4.io и анализировать их через веб-интерфейс.

FlightPlot

Также лог-файл можно анализировать с помощью программы FlightPlot. Актуальную версию программы можно скачать на GitHub.

В программе можно просмотреть полный список записанных топиков (*Fields List*). В нем нужно выбрать необходимые топики, после чего они появятся на графике:



Основные топики в PX4

uORB представляет собой pub/sub механизм, аналогичный ROS-топикам, однако сильно упрощенный и подходящий для embedded-среды.

Полный список топиков можно узнать в исходном коде проекта в каталоге `msg`.

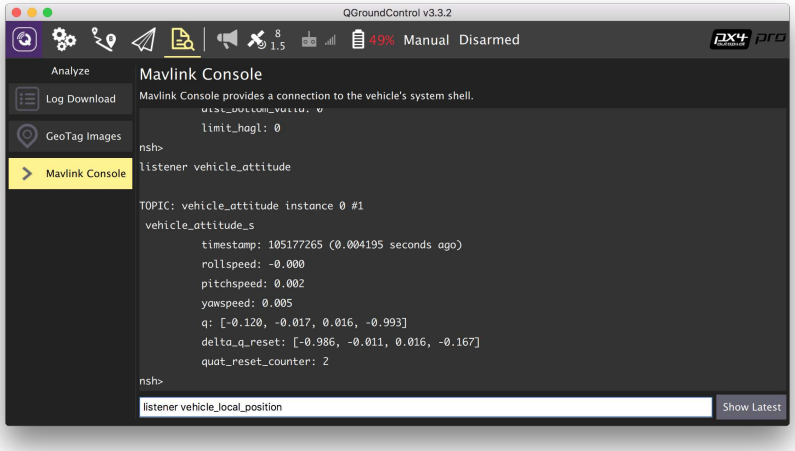
Список некоторых топиков:

- **vehicle_status** – состояние коптера (режим и т. д.);
- **vehicle_local_position** – локальная позиция коптера;
- **vehicle_attitude** – ориентация коптера;
- **vehicle_local_position_setpoint** – целевая точка (setpoint) коптера по позиции;
- **vehicle_global_position** – глобальная позиция коптера;
- **vehicle_vision_position** – полученная визуальная позиция коптера, аналог MAVLink пакета `VISION_POSITION_ESTIMATE` или MAVROS-топика `/mavros/vision_position_estimate/pose`;
- **att_pos_mocap** – полученная MOCAP-позиция коптера, аналог MAVLink пакета `ATT_POS_MOCAP` или MAVROS-топика `/mavros/mocap/pose`;
- **actuator_controls** – сигналы на моторы;
- **vehicle_land_detected** – статус Land-detector'a;
- **optical_flow** – данные с модуля optical flow.

Мониторинг топиков в режиме реального времени

Для более новых версий платы Pixhawk (`px4fmu-v3`), а также для плат Pixracer, в прошивку включен модуль `topic_listener`, который позволяет просматривать значения топиков в режиме реального времени (в том числе в полете).

Для ее использования нужно выбрать вкладку MAVLink Console в QGroundControl:



Команда `list_topics` выводит список топиков, доступных для просмотра (включена только в SITL).

Команда `listener <название топика>` выводит текущее значение в топике. Существует третий опциональный параметр, который определяет количество сообщений, которые необходимо вывести.

Примеры команд:

```
listener vehicle_local_position
```

```
listener vehicle_attitude 5
```

Прошивка полетного контроллера

Pixhawk, Pixracer и COEX Pix можно прошить, используя QGroundControl или утилиты командной строки.

Прошивка для Клевера

Для Клевера рекомендуется использование специальной сборки PX4, которая содержит необходимые исправления и более подходящие параметры по умолчанию. Используйте последний стабильный релиз в GitHub-репозитории, содержащий слово `clover`, например `v1.8.2-clover.4`.

> Если вы используете прошивку с версией ниже, чем *v1.10* (например `v1.8.2-clover.13`), то во избежание ошибок конфигурирования полётного контроллера, используйте QGroundControl версии *v4.2.0* (или ниже). См. подробную информацию об изменениях в параметрах, которые вызывают ошибки в более новых версиях QGroundControl.

QGroundControl

В QGroundControl откройте раздел Firmware. **После** этого подключите полетный контроллер по USB.

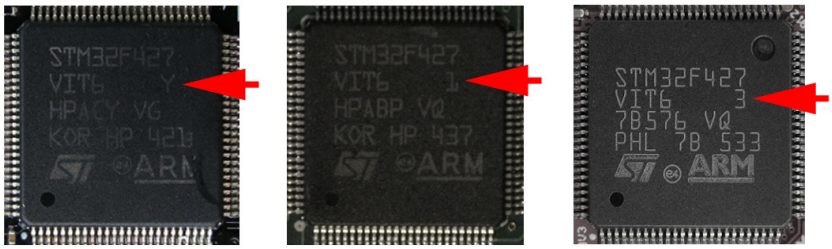
Выберите PX4 Flight Stack. Для скачивания и загрузки стандартной прошивки (вариант с EKF2 для Pixhawk) выберите пункт меню "Standard Version", для загрузки собственного файла прошивки выберите пункт "Custom firmware file...", затем нажмите OK.

> Не отключайте USB-кабель до окончания процесса прошивки.

Варианты прошивок

В названии файла прошивки кодируется информации о целевой плате и варианте сборки. Примеры:

- `px4fmu-v4_default.px4` — прошивка для COEX Pix и Pixracer с EKF2 и LPE (**Клевер 3** / **Клевер 4**).
- `px4fmu-v2_lpe.px4` — прошивка для Pixhawk с LPE (**Клевер 2**).
- `px4fmu-v2_default.px4` — прошивка для Pixhawk с EKF2.
- `px4fmu-v3_default.px4` — прошивка для более новых версий Pixhawk (чип ревизии 3, см. илл. + Bootloader v5) с EKF2 и LPE.



> Для загрузки `px4fmu-v3_default.px4` может понадобиться использование команды `force_upload` из командной строки.

Командная строка

PX4 может быть собран из исходников и загружен в плату автоматически из командной строки.

Для это склонируйте репозиторий PX4:

```
git clone https://github.com/PX4/PX4-Autopilot.git
```

Выберите необходимую версию (тэг) с помощью `git checkout` . Затем соберите и загрузите прошивку:

```
make px4_fmu-v4_default upload
```

Где `px4_fmu-v4_default` – требуемый вариант прошивки.

Для загрузки прошивки `v3` в Pixhawk может понадобиться команда `force_upload` :

```
make px4_fmu-v3_default force-upload
```

MAVLink

Основная документация: https://mavlink.io/en/.

MAVLink – это протокол для организации связи между автономными летательными и транспортными системами (дронами, самолетами, автомобилями). Протокол MAVLink лежит в основе взаимодействия между Pixhawk и Raspberry Pi.

В Клевер включено 2 обертки над этим протоколом: MAVROS и simple_offboard.

Код для отправки произвольного MAVLink сообщения можно найти в примерах.

Основные концепции

Канал связи

Протокол MAVLink может быть использован поверх следующих каналов связи:

- последовательное соединение (UART, USB и др.);
- UDP (Wi-Fi, Ethernet, 3G, LTE);
- TCP (Wi-Fi, Ethernet, 3G, LTE).

Сообщение

MAVLink-сообщение это отдельная "порция" данных, передаваемая между устройствами. Отдельное MAVLink-сообщение содержит информацию о состоянии дрона (или другого устройства) или команду для дрона.

Примеры MAVLink-сообщений:

- `ATTITUDE`, `ATTITUDE_QUATERNION` – ориентация квадрокоптера в пространстве;
- `LOCAL_POSITION_NED` – локальная позиция квадрокоптера;
- `GLOBAL_POSITION_INT` – глобальная позиция квадрокоптера (широта/долгота/высота);
- `COMMAND_LONG` – команда для квадрокоптера (взлететь, сесть, переключить режим и т. д.).

Полный список MAVLink-сообщений можно посмотреть в документации MAVLink.

Система, компонент системы

Каждое устройство (дрон, базовая станция и т. д.) имеет ID в сети MAVLink. В PX4 MAVLink ID меняется с помощью параметра `MAV_SYS_ID`. Каждое MAVLink сообщение содержит поле с ID системы-отправителя. Кроме того,

некоторые сообщения (например, `COMMAND_LONG`) содержат также ID системы-получателя.

Помимо ID систем, сообщения могут содержать ID компонента-отправителя и компонента-получателя. Примеры компонентов системы: полетный контроллер, внешняя камера, управляющий бортовой компьютер (Raspberry Pi в случае Клевера) и т. д.

Пример пакета

Пример структуры MAVLink-пакета с сообщением `COMMAND_LONG`:

	Поле	Длина	Имя	Ко
Заголовок	`magic`	1 байт	Метка начала	0xFD 2.0
	`len`	1 байт	Размер данных	
	`incompat_flags`	1 байт	Обратно несовместимые флаги	На да не ис
	`compat_flags`	1 байт	Обратно совместимые флаги	На да не ис
	`seq`	1 байт	Порядковый номер сообщения	
	`sysid`	1 байт	ID системы-отправителя	
	`compid`	1 байт	ID компонента-отправителя	
	`msgid`	3 байта	ID сообщения	
Данные (пример)	`target_system`	1 байт	ID системы-получателя	
	`target_component`	1 байт	ID компонента–получателя	
	`command`	2 байта	ID команды	
	`confirmation`	1 байт	Номер для подтверждения	
	`param1`	4 байта	Параметр 1	Число плава одина точно
	`param2`	4 байта	Параметр 2	
	`param3`	4 байта	Параметр 3	
	`param4`	4 байта	Параметр 4	
	`param5`	4 байта	Параметр 5	
	`param6`	4 байта	Параметр 6	
	`param7`	4 байта	Параметр 7	
	`checksum`	2 байта	Контрольная сумма	

	`signature`	13 байт	Сигнатура (опционально)	Позво убеди пакет скомп Обыч испол

Желтым цветом выделены поля данных (полезной нагрузки). Для каждого типа сообщения существует свой набор таких полей.

Использование мультиметра

Проверка цепей (прозвонка)

> Проверяемый объект должен быть отключен от питания (обесточен)!

С помощью мультиметра проверить отсутствие короткого замыкания (прозвонить):

- Выставить мультиметр в режим прозвона.
- Проверить работу мультиметра путем замыкания щупов между собой. При корректной работе прибор издаст характерный звук.
- Попарно красный щуп прикладывается к “+ ”контакту, черный к “-” / ”GND”. Если в цепи есть короткое замыкание, издается звук.

 Режим прозвонки

1. Прозвонить следующие цепи на НЕЗАМКНУТОСТЬ (отсутствие звукового сигнала мультиметра):

- “BAT+” и “BAT-”
- “12V” и “GND”
- “5V” и “GND”

2. Прозвонить следующие цепи на ЗАМКНУТОСТЬ (появление звукового сигнала мультиметра):

- “BAT-” с каждым контактом, обозначенным “-” и “GND”
- “BAT+”, с каждым контактом, обозначенным “+”

Проверка напряжения

С помощью мультиметра необходимо удостовериться, что преобразователи напряжения, расположенные на плате распределения питания, работают исправно и выдают напряжение 5В и 12В соответственно.

- Переключить мультиметр в режим "Измерение постоянного напряжения"
- Выбрать верхнюю границу измеряемого напряжения (в нашем случае, не более 20 В)
- Убедиться, что АКБ подключено
- Провести следующие измерения:
    1. Замерить напряжение АКБ (между BAT+ и BAT-). Оно должно лежать в пределах от 14.0В до 16.8В
    2. Замерить напряжение на выходе 5В. Оно не должно превышать 5.5В
    3. Замерить напряжение на выходе 12В. Оно не должно превышать 12.5В

После проведенных измерений:

- отключите АКБ
- выключите мультиметр

Неисправности радиоаппаратуры

Пульт заблокирован

Если пульт заблокирован, то на ЖК Экране будет отображено предупреждение: *Warning. Place all switches in their up position and lower the throttle*.

Для разблокировки пульта необходимо привести все стики и переключатели в исходное положение:

1. Левый стик (1) в центральной нижней позиции.
2. Переключатели A, B, C, D (2) в положение "от себя".
3. Правый стик (3) в центре.



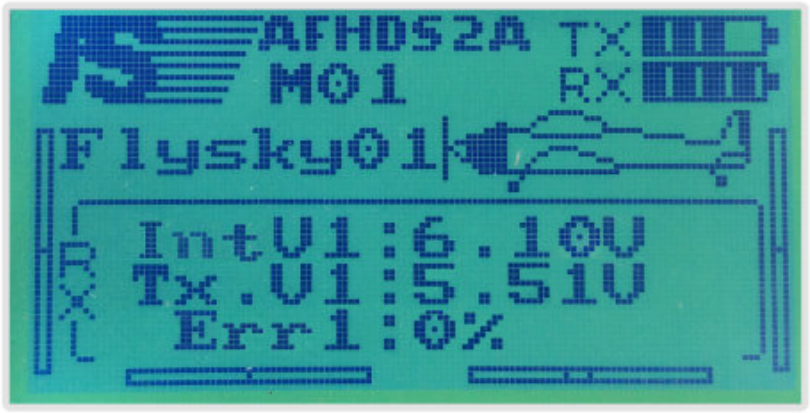
Нет связи с приемником

Для проверки соединения пульта с приемником, включите пульт и обратите внимание на индикацию на ЖК экране.

1. Соединение с приемником отсутствует:



2. Соединение с приемником установлено:

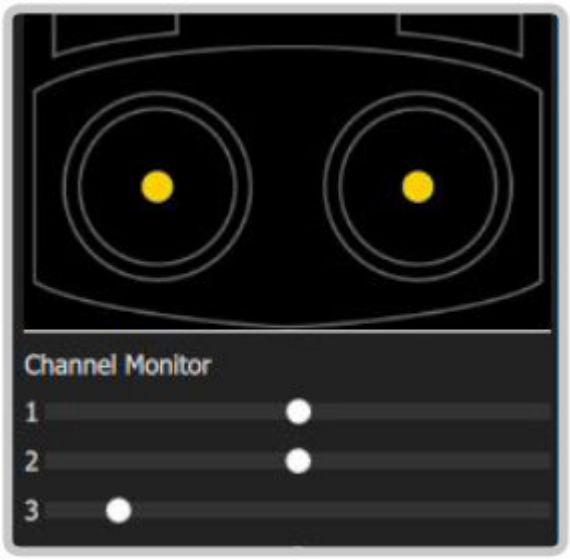


Если соединение отсутствует, то:

1. Проверьте, что приемник включен (моргает красный светодиод). Если светодиод горит непрерывно красным, то значит связь установлена с другим пультом.
2. Проведите процедуру сопряжения пульта и приемника.

Нет связи с полетным контроллером

Если нет связи с полетным контроллером, то на экране монитора компьютера в окне *Channel Monitor* не будут отображаться изменения положения слайдеров при перемещении стиками пульта.



1. Зайдите в меню (удерживайте нажатой кнопку *OK*).
2. Выберите меню *System setup* (Кнопки Up/Down - для навигации, кнопка *OK* — подтверждение выбора.
3. Выберите *RX setup* > *PPM OUTPUT* > *“On*.
4. Сохраните изменения (удерживайте нажатой кнопку *CANCEL*).

Прошивка ESC регуляторов с помощью BLHeliSuite

Хорошая статья, которая объясняет принцип работы ESC (Electric speed controller) регуляторов: http://www.avmodels.ru/engines/electric/esc.html

Зачем перепрошивать?

Иногда требуется поменять один из параметров регулятора, например направление вращения мотора, минимальная и максимальная скважности PPM сигнала на входе контроллера, уровень громкости звуковых сигналов, издаваемых мотором или время, через которое регулятор начинает напоминать, что он включён.

Программа для прошивки регуляторов

Для прошивки самых разнообразных ESC регуляторов существует программа BLHeliSuite (для Windows).

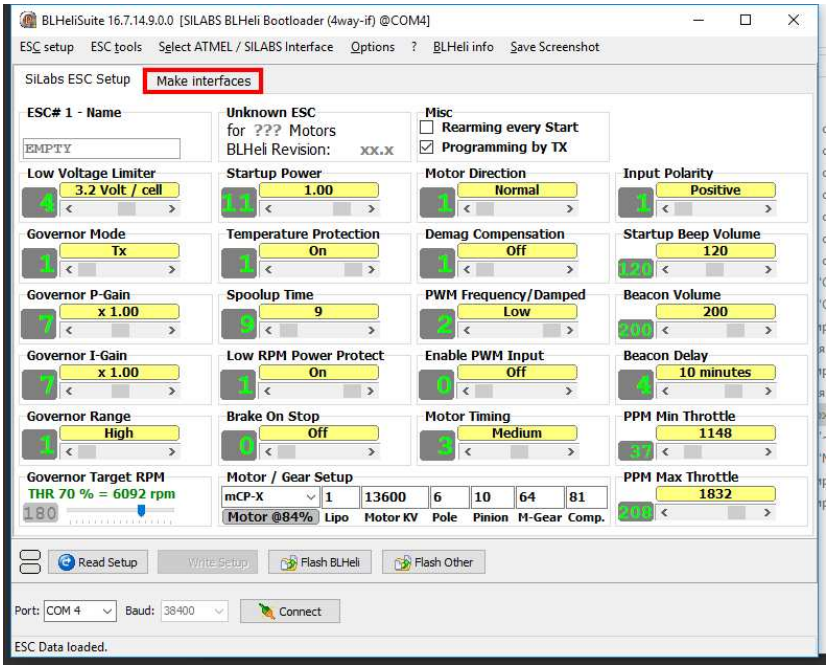
Для запуска программы (BLHeliSuite.exe) необходимо распаковать архивы BLHeliAtmelHEX.zip и BLHeliSilabsHEX.zip в папку с программой.

Программатор для прошивки регуляторов

Чтобы прошить регулятор, необходим программатор, который умеет общаться с контроллером регулятора по 1-wire протоколу. Один из способов добыть программатор - взять подвернувшуюся под руку ардуинку и прошить её специальной прошивкой. В BLHeliSuite есть инструмент для создания интерфейсов программаторов.

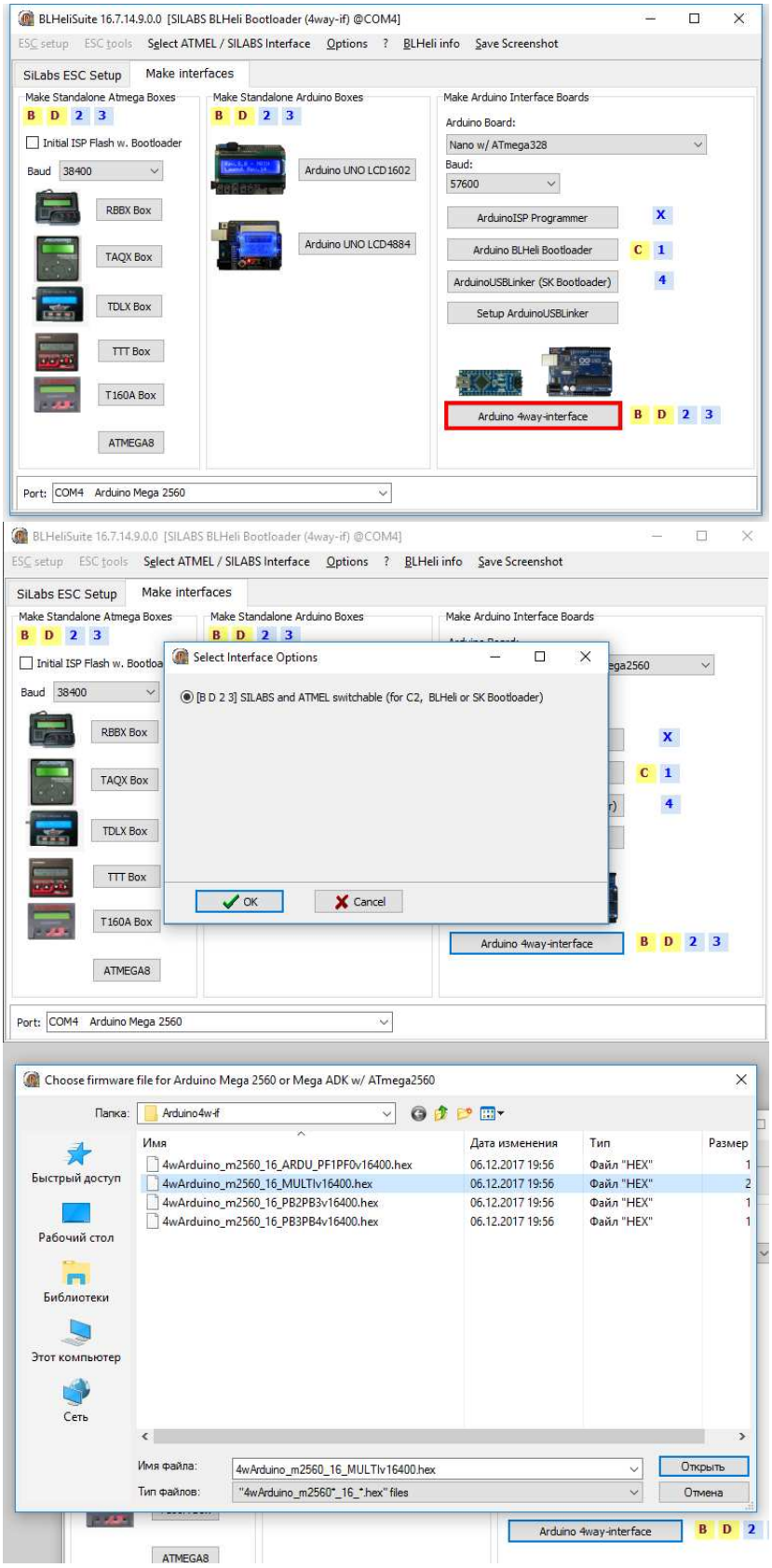
Создание программатора на примере Arduino Mega.

1. Запустите программу BLHeliSuite и выберите вкладку Make interfaces.

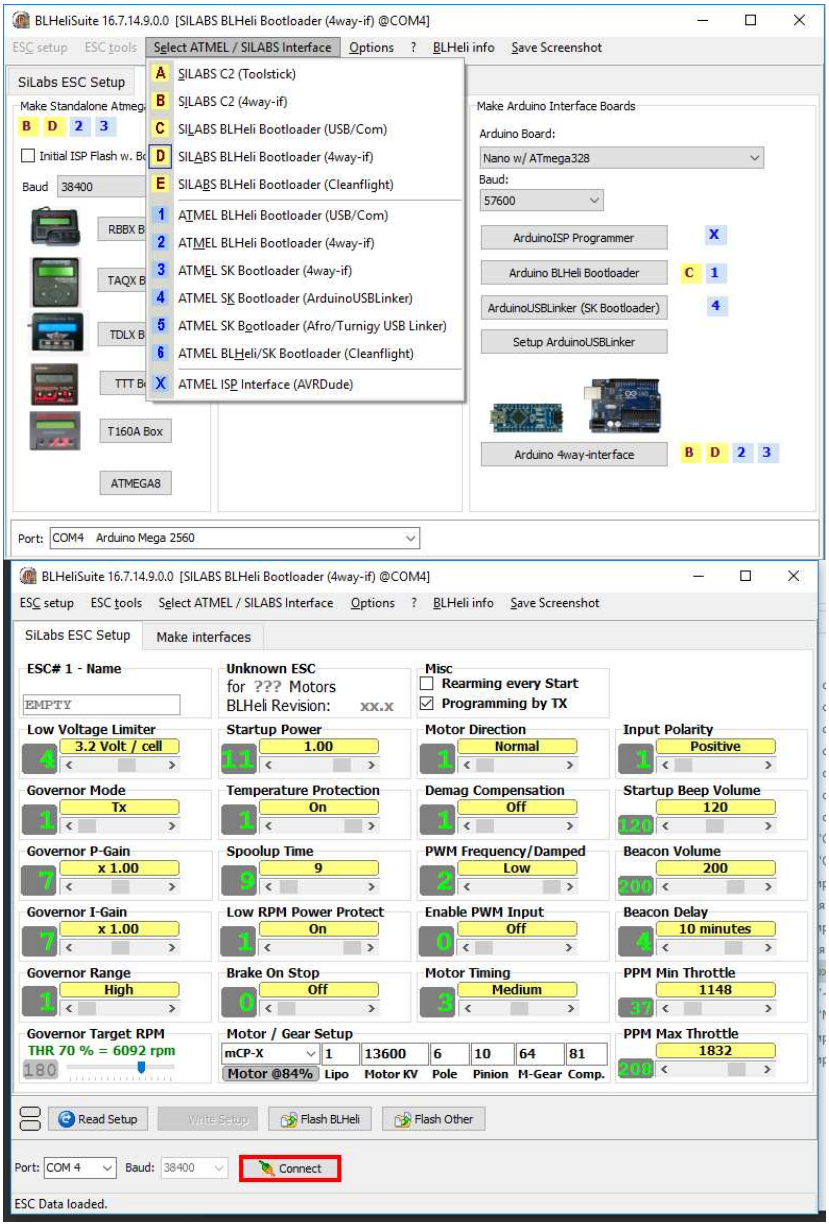


2. Подключите Arduino к компьютеру, при необходимости посмотрите в диспетчере устройств номер COM порта, к которому подключена плата.

3. Нажмите Arduino 4way-interface в разделе Make Arduino Interface Boards и выберите файл прошивки. После выбора файла начнётся прошивка контроллера.

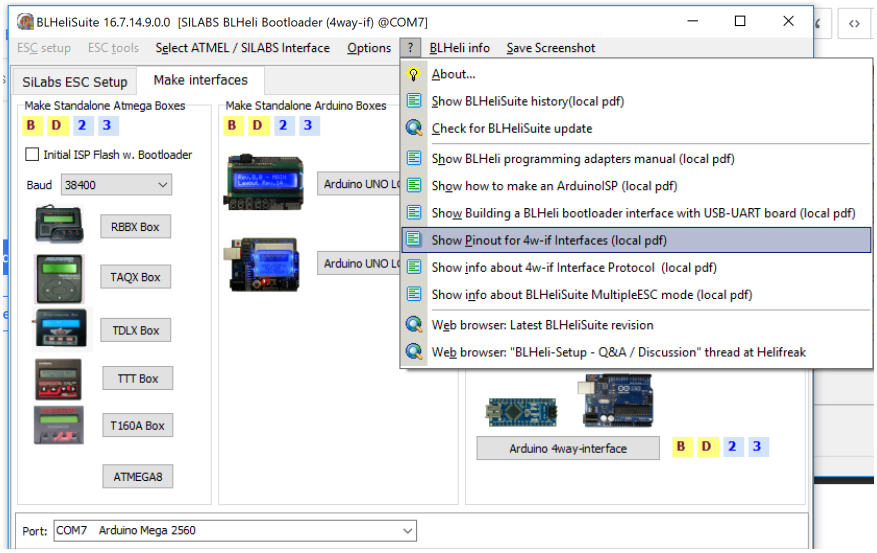


4. После прошивки Arduino вернитесь на вкладку Silabs ESC Setup и подключитесь к Arduino, предварительно выбрав интерфейс программатора 4way-if и COM порт Arduino.



Подключение ESC регуляторов к Arduino

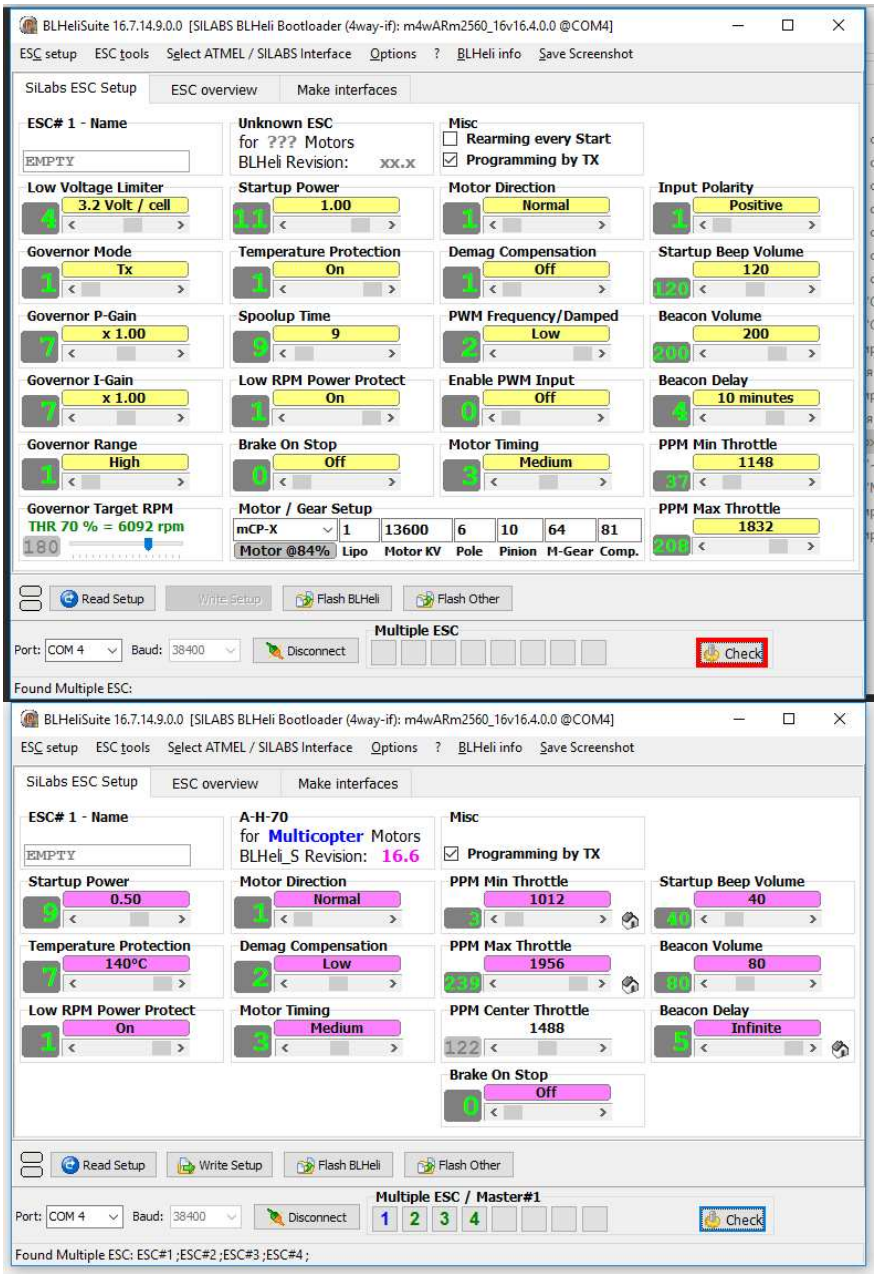
Для прошивки или изменения настроек регуляторов необходимо подключить сигнальные порты (обычно белого цвета) ESC регуляторов к портам Arduino, предварительно посмотрев в мануале (см. рисунок ниже), какие порты используются для соединения с регуляторами. Так же нужно соединить GND Arduino с землёй одного из регуляторов (обычно черного цвета). Регуляторы должны быть подключены к питанию, а если к регуляторам подключены моторы, **на них не должно быть винтов**.



В случае с Arduino Mega, сигнальные порты регуляторов подключаются к портам D43-D49 и D51.

Изменение настроек ESC регуляторов

Для загрузки информации о версии прошивки и настроек регуляторов нужно нажать на кнопку Check.

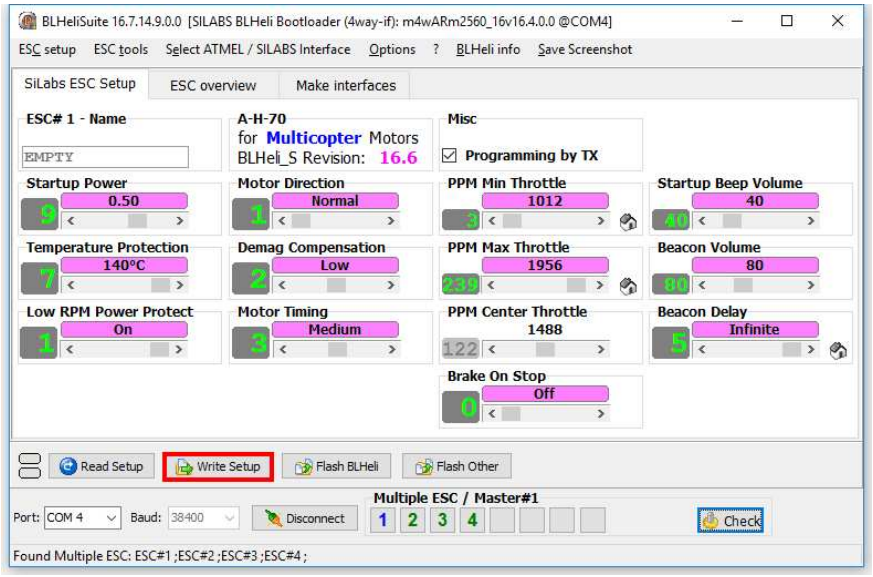


Основные параметры, которые нас интересуют, это:

- Motor Direction (Normal или Reversed) - отвечает за направление вращения моторов. Удобно настраивать, если нет желания перепаивать неправильно припаянный мотор.
- PPM Min и Max Throttle - отвечает за минимальный и максимальный сигнал газа
- Startup Beep Volume - громкость стартового сигнала. В версии прошивки 16.65 добавлена возможность изменения стартовой мелодии. Подробнее об этом написано здесь. Например, можно в качестве мелодии запуска установить имперский марш из Звёздных Войн или главную тему Игры престолов.
- Beacon Volume - громкость обнаруживающего сигнала. Когда моторы не крутятся некоторое время и регулятор не используется, он начинает напоминать о себе писком моторов.

- Beacon Delay - время бездействия, после которого включается обнаруживающий сигнал. При разработке он может хорошенько надоедать, поэтому его можно выставлять в бесконечность.

Самый левый мотор в списке моторов (Multiple ESC) считается главным (мастер). Нажимая на номера моторов, можно включать/выключать возможность записи в них настроек. После изменения необходимых параметров можно записать в нужные моторы настройки, нажав на кнопку Write Setup.



Для отображения настроек со всех регуляторов одновременно можно воспользоваться вкладкой ESC Overview.

Прошивка ESC регуляторов

Файлы с прошивками регуляторов находятся здесь.

Для перепрошивки регуляторов нужно нажать на кнопку Flash BLHeli и выбрать файл прошивки с типом контроллера, название которого указано в рамке названия прошивки и находится сверху во вкладке Silabs ESC Setup (в случае контроллера, который используется в конструкторе Клевер 2, это A-H-70).

Для перепрошивки отдельного регулятора нужно сделать все остальные неактивными.

Видеоинструкция по перепрошивке ESC регуляторов

Для лучшего понимания того, что описано в статье, рекомендуем посмотреть наглядное руководство по подключению электроники и прошивке регуляторов на английском языке на youtube.

Управление коптером с Arduino

Для взаимодействия с ROS-топиками и сервисами на Raspberry Pi можно использовать библиотеку rosserial_arduino. Эта библиотека предустановлена на образе для Raspberry Pi.

Основной туториал по rosserial: http://wiki.ros.org/rosserial_arduino/Tutorials

Arduino необходимо установить на Клевер и подключить по USB-порту.

Настройка Arduino IDE

Для работы с ROS Arduino необходимо понимать формат сообщений установленных пакетов. Для этого на Raspberry Pi необходимо собрать библиотеку ROS-сообщений:

```
rosrun rosserial_arduino make_libraries.py .
```

Полученный каталог `ros_lib` необходимо скопировать в `<папку скетчей>/libraries` на компьютере с Arduino IDE.

Настройка Raspberry Pi

Для запуска `rosserial` создайте файл `arduino.launch` в каталоге `~/catkin_ws/src/clover/clover/launch/` со следующим содержимым:

```
<launch>
    <node pkg="rosserial_python" type="serial_node.py" name="serial_node" outpu
        <param name="port" value="/dev/serial/by-id/usb-1a86_USB2.0-Serial-if0
    </node>
</launch>
```

Чтобы единоразово запустить программу на Arduino, можно будет воспользоваться командой:

```
roslaunch clover arduino.launch
```

Чтобы запускать связку с Arduino при старте системы автоматически, необходимо добавить запуск созданного launch-файла в основной launch-файл Клевера (`~/catkin_ws/src/clover/clover/launch/clover.launch`). Добавьте в конец этого файла строку:

```
<include file="$(find clover)/launch/arduino.launch"/>
```

При изменении launch-файла необходимо перезапустить пакет `clover` :

```
sudo systemctl restart clover
```

Задержки

При использовании `rosserial_arduino` микроконтроллер Arduino не должен быть заблокирован больше чем на несколько секунд (например, с использованием функции `delay`); иначе связь между Raspberry Pi и Arduino будет разорвана.

При реализации долгих циклов `while` обеспечьте периодический вызов функции `hn.spinOnce`:

```
while(/* условие */) {
  // ... Произвести необходимые действия
  nh.spinOnce();
}
```

Для организации долгих задержек используйте задержки в цикле с периодическим вызовом функции `hn.spinOnce()`:

```
// Задержка на 8 секунд
for(int i=0; i<8; i++) {
  delay(1000);
  nh.spinOnce();
}
```

Работа с Клевером

Набор сервисов и топиков аналогичен обычному набору в simple_offboard и mavros.

Пример программы, контролирующей коптер по позиции, с использованием сервисов `navigate` и `set_mode`:

```
// Подключение библиотек для работы с rosserial
#include <ros.h>

// Подключение заголовочных файлов сообщений пакета clover и MAVROS
#include <clover/Navigate.h>
#include <mavros_msgs/SetMode.h>

using namespace clover;
using namespace mavros_msgs;

ros::NodeHandle nh;

// Объявление сервисов
ros::ServiceClient<Navigate::Request, Navigate::Response> navigate("/navigate")
ros::ServiceClient<SetMode::Request, SetMode::Response> setMode("/mavros/set_mo

void setup()
{
  // Инициализация rosserial
  nh.initNode();

  // Инициализация сервисов
  nh.serviceClient(navigate);
  nh.serviceClient(setMode);

  // Ожидание подключение к Raspberry Pi
  while(!nh.connected()) nh.spinOnce();
  nh.loginfo("Startup complete");

  // Пользовательская настройка
  // <...>

  // Тестовая программа
  Navigate::Request nav_req;
  Navigate::Response nav_res;
  SetMode::Request sm_req;
  SetMode::Response sm_res;

  // Взлет на 2 метра:
  nh.loginfo("Take off");
  nav_req.auto_arm = false;
  nav_req.x = 0;
  nav_req.y = 0;
  nav_req.z = 2;
  nav_req.frame_id = "body";
  nav_req.speed = 0.5;
  navigate.call(nav_req, nav_res);

  // Ждем 5 секунд
  for(int i=0; i<5; i++) {
      delay(1000);
      nh.spinOnce();
  }

  nav_req.auto_arm = false;

  // Пролет вперед на 3 метра:
  nh.loginfo("Fly forward");
  nav_req.auto_arm = true;
  nav_req.x = 3;
  nav_req.y = 0;
  nav_req.z = 0;
  nav_req.frame_id = "body";
  nav_req.speed = 0.8;
  navigate.call(nav_req, nav_res);
```

```
  // Ждем 5 секунд
  for(int i=0; i<5; i++) {
    delay(1000);
    nh.spinOnce();
  }

  // Полет в точку 1:0:2 по маркерному полю
  nh.loginfo("Fly on point");
  nav_req.auto_arm = false;
  nav_req.x = 1;
  nav_req.y = 0;
  nav_req.z = 2;
  nav_req.frame_id = "aruco_map";
  nav_req.speed = 0.8;
  navigate.call(nav_req, nav_res);

  // Ждем 5 секунд
  for(int i=0; i<5; i++) {
    delay(1000);
    nh.spinOnce();
  }

  // Посадка
  nh.loginfo("Land");
  sm_req.custom_mode = "AUTO.LAND";
  setMode.call(sm_req, sm_res);
}

void loop()
{
}
```

Получение телеметрии

С Arduino можно использовать сервис `get_telemetry`. Для этого надо объявить его по аналогии с сервисами `navigate` и `set_mode`:

```
#include <ros.h>

// ...

#include <clover/GetTelemetry.h>

// ...

ros::ServiceClient<GetTelemetry::Request, GetTelemetry::Response> getTelemetry(

// ...

nh.serviceClient(getTelemetry);

// ...

GetTelemetry::Request gt_req;
GetTelemetry::Response gt_res;


// ...

gt_req.frame_id = "aruco_map"; // фрейм для значений x, y, z
getTelemetry.call(gt_req, gt_res);

// gt_res.x - положение коптера по x
// gt_res.y - положение коптера по y
// gt_res.z - положение коптера по z
```

Проблемы

При использовании Arduino Nano может не хватать оперативной памяти (RAM). В таком случае в Arduino IDE будут появляться сообщения, типа:

```
Глобальные переменные используют 1837 байт (89%) динамической памяти, оставляя
Недостаточно памяти, программа может работать нестабильно.
```

Можно сократить использование оперативной памяти уменьшив размер выделяемых буферов для передачи и приема сообщений. Для этого **в самое начало** программы следует поместить строку:

```
#define __AVR_ATmega168__ 1
```

Можно уменьшить количество занятой памяти еще сильнее, если вручную настроить количество publisher'ов и subscriber'ов, а также размеры буферов памяти, выделяемой для сообщений, например:

```
#include <ros.h>

// ...

typedef ros::NodeHandle_<ArduinoHardware, 3, 3, 100, 100> NodeHandle;

// ...
NodeHandle nh;
```

Подключение GPS

При подключении GPS появляются следующие возможности:

- удерживание коптером позиции при полете на улице;
- программирование автономных миссий в программе QGroundControl;
- полеты на глобальные точки в автономном режиме при помощи модуля simple_offboard.

Полезные ссылки:

- https://docs.px4.io/en/assembly/quick_start_pixhawk.html
- http://ardupilot.org/copter/docs/common-pixhawk-wiring-and-quick-start.html
- http://ardupilot.org/copter/docs/common-installing-3dr-ublox-gps-compass-module.html

Подключение

GPS-модуль подключается к разъемам "GPS" и "I2C" (компас) полетного контроллера.

При подключении GPS необходимо заново откалибровать магнитометры в программе QGroundControl, подключившись по Wi-Fi или USB.

Далее, необходимо включить GPS в параметре `EKF2_AID_MASK` (при использовании EKF2) или `LPE_FUSION` (при использовании LPE). При использовании LPE вес компаса должен быть больше нуля (`ATT_W_MAG` = 0.1).

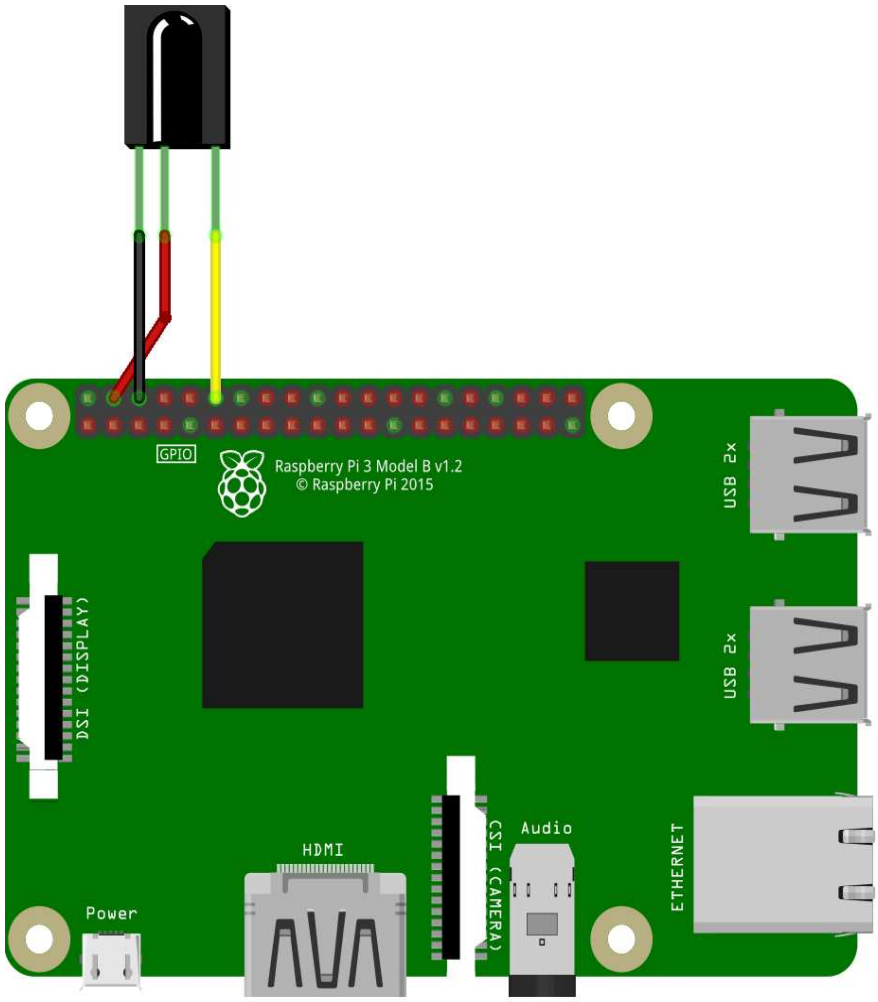
Работа с ИК-датчиками на Raspberry Pi 3

> Данная статья не актуальна для последних версий образа *clover* и работает только на версиях *clover_v0.16-clover_v0.17*.

Инфракрасные датчики – удобный инструмент для передачи каких-либо команд на коптер. Они гибки в настройке и взаимодействие с ними возможно из языка Python.

Подключение ИК-приемника

Большинство ИК-приемников работают и подключаются одинаково. У таких приемников есть 3 пина для подключения: G/GND – земля, V/VCC – питание 5В, S/OUT – сигнал.



> Сигнальный порт не обязательно должен подключаться к порту GPIO 17, данный пин можно изменить во время настройки портов in/out.

Настройка ИК-приемника и работа с модулем LIRC

LIRC (Linux Infrared Remote Control) – стабильная и проверенная библиотека с открытым кодом, которая позволяет отправлять и получать команды по инфракрасному порту. LIRC поддерживается Raspbian.

Для установки LIRC и сопутствующих модулей нужно подключить вашу Raspberry Pi к интернету и выполнить консольные команды:

```
sudo apt-get update
sudo apt-get install lirc
sudo apt-get install python-lirc
pip install py-irsend
```

> Для корректного редактирования системных файлов требуется иметь права доступа администратора, используйте `sudo` при вызове редактора текста.

После установки модуля нужно отредактировать файл `/etc/modules` и добавить в него строки:

```
lirc_dev
lirc_rpi gpio_in_pin=18 gpio_out_pin=17
```

Где:

- `gpio_in_pin` – пин входа от приемника
- `gpio_out_pin` – пин выхода для передатчика

Обновите следующую строку в файле `/boot/config.txt`:

```
dtoverlay=lirc-rpi,gpio_in_pin=18,gpio_out_pin=17
```

Добавьте следующие строки в файл `/etc/lirc/hardware.conf`. При отсутствии данного файла создайте его вручную.

```
LIRCD_ARGS="--uinput --listen"
LOAD_MODULES=true
DRIVER="default"
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"
```

Обновите следующие строки в файле `/etc/lirc/lirc_options.conf`

```
driver    = default
device    = /dev/lirc0
```

Все требуемые настройки сделаны, теперь нужно перезагрузить ваше устройство Raspberry Pi для завершения установки. Для этого выполните:

```
sudo reboot
```

После перезагрузки проверьте его статус, вызвав команду:

```
sudo /etc/init.d/lircd status
```

Если вы все сделали правильно, то статус работы должен быть `active`. Чтобы проверить, что установленный модуль LIRC работает, выключите демон `lircd` и вызовите соответствующую команду:

```
sudo /etc/init.d/lircd stop
mode2 -d /dev/lirc0
```

Теперь, направив ИК-передатчик на ваше устройство и нажав несколько кнопок, вы должны увидеть что-то похожее на это:

```
space 402351
pulse 135
space 7085
pulse 85
space 2903
pulse 560
space 1706
pulse 535
```

> В ситуации, если вы используете ИК-передатчик (TV-пульт, пульт от кондиционеров, и т. д.) и во время проверки вы не получаете сигнал, возможно ваш пульт использует другую частоту передачи сигнала. При использовании приемников типа TSOP 22XX рабочая частота приема сигнала будет в диапазоне от 30 до 50 кГц.

Запись своей конфигурации ИК-пульта

В случае, если вы хотите использовать свой ИК-передатчик, вам нужно записать его характерные настройки с помощью прилагающегося модуля `irrecord`. Для этого вам нужно выключить демон `lircd` и вызвать соответствующую команду. Во время калибровки пульта точно выполняйте все написанные инструкции.

> Обратите внимание, что на последнем шаге калибровки вам нужно будет задать наименования кнопок, которые вы захотите расшифровать программно. Для того чтобы посмотреть список доступных имен, вызовите команду `irrecord --list-namespace`.

```
irrecord -d /dev/lirc0 ~/lircd.conf
```

Если у вас успешно получилось записать конфигурацию вашего пульта, в папке `/home/pi/` должен был появиться файл `ваше-имя.lircd.conf`. Теперь вам нужно перенести записанный конфигурационный файл в рабочую папку `lirc` и перезагрузить демон:

```
sudo cp ~/ваше-имя.lircd.conf /etc/lirc/lircd.conf
sudo /etc/init.d/lircd restart
```

Для того чтобы проверить, распознается ли записанная вами конфигурация, вызовите соответствующий модуль. Теперь при нажатии на кнопки, которые вы записали в ранее созданной конфигурации, в терминал будет выводиться отладочная информация о том, какая кнопка была нажата.

```
irw
```

> В случае работы с некоторыми пультами бывают ситуации, когда битовые описания кнопок являются излишними и в таком случае у вас может не работать команда `irw`. Что бы исправить эту ошибку откройте файл `etc/lirc/lircd.conf` и проверьте как выглядит описание ваших клавиш, если оно похоже на `KEY_1 0x00FF6897 0x7EE0CF2C` и во всех строках у вас второе число совпадает, то вам нужно его удалить, что бы строки с назначением кнопок выглядели таким образом `KEY_1 0x00FF6897` и все числа в них были уникальны. После выполнения этих действий закройте файл и перезагрузите демон.

Если вы все сделали правильно, то при нажатии кнопки вы увидите выход похожий на:

```
0000000000ff6897 00 KEY_1 pult
0000000000ff6897 01 KEY_1 pult
0000000000ff9867 00 KEY_2 pult
0000000000ff9867 01 KEY_2 pult
```

Это значит, что ваша конфигурация корректно распознается программой и теперь вы можете запрограммировать нужные вам действия в случае нажатия определенной клавиши.

Работа с ИК-датчиками в Python

Чтобы иметь возможность использовать сигналы с ИК-приемника в Python программах, вам потребуется пакет `python-lirc`. Установите его при необходимости.

Для корректного получения информации в собственном скрипте нужно создать файл `lircrc`, в котором будут храниться настройки ваших кнопок и программный ответ в случае их вызова.

Данный файл создается в той же папке, из которой будет вызван ваш скрипт, по умолчанию это `/home/pi/`.

Для того чтобы создать требуемый файл используйте любой текстовый редактор:

```
sudo nano .lircrc
```

Формат данного файла должен быть примерно таким:

```
begin
prog = myprogram
button = KEY_1
config = one
end

begin
prog = myprogram
button = KEY_2
config = two
end
```

Где:

- `prog` – имя программы, которое вы будете вызывать в своем скрипте
- `button` – наименование клавиши, которое вы вводили во время настройки пульта
- `config` – информация, которая будет передана вашей программе в случае нажатия указанной клавиши

Все настройки выполнены и теперь можно переходить непосредственно к программированию ИК-сигналов.

Для этого нужно создать Python скрипт, который будет принимать значения нажатых клавиш и выполнять в соответствии с ними требуемые действия. Пример такого скрипта:

```
import lirc
import fly_module

# ...

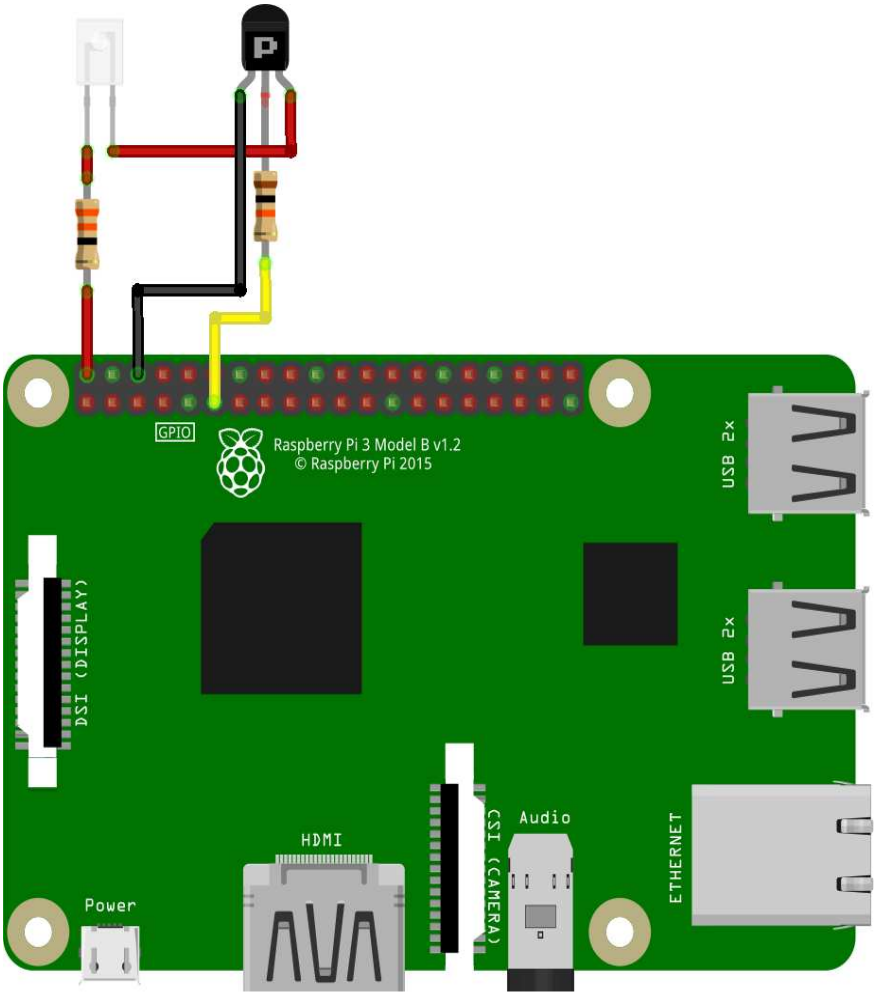
sockid = lirc.init('myprogram')

inf = lirc.nextcode()
if inf[0] == 1:
    print('You pressed key 1')
elif inf[0] == 2:
    print('You pressed key 2')

lirc.deinit()
```

Работа с ИК-передатчиком

Для того, что бы работать с ИК-передатчиком, подключите его к тем портам, которые указывали при настройке.





> В случае, если вы используете готовую плату ИК-передатчика, подключите ее к нужным пинам Raspberry в соответствии с маркировкой пинов, точно так же как и с приемником.

Если все правильно подключено, то можно отправлять сигналы заданные на моменте настройки пульта, используя команду:

```
irsend SEND_ONCE deviceName keyName
```

Где:

- SEND_ONCE - параметр отвечающий за то, посылаете вы один сигнал или он передается как от зажатой кнопки
- deviceName - имя пульта, которое вы давали во время его настройки
- keyName - имя одной из кнопок, которые были вами заданы во время настройки пульта

Для того чтобы работать с `irsend` внутри вашего скрипта, вам потребуется модуль `python-irsend`, при необходимости установите его.

Чтобы использовать `irsend` импортируйте библиотеку и вызовите соответствующую команду:

```
from py_irsend import irsend


irsend.send_once('YourRemote', ['YourKey'])
```

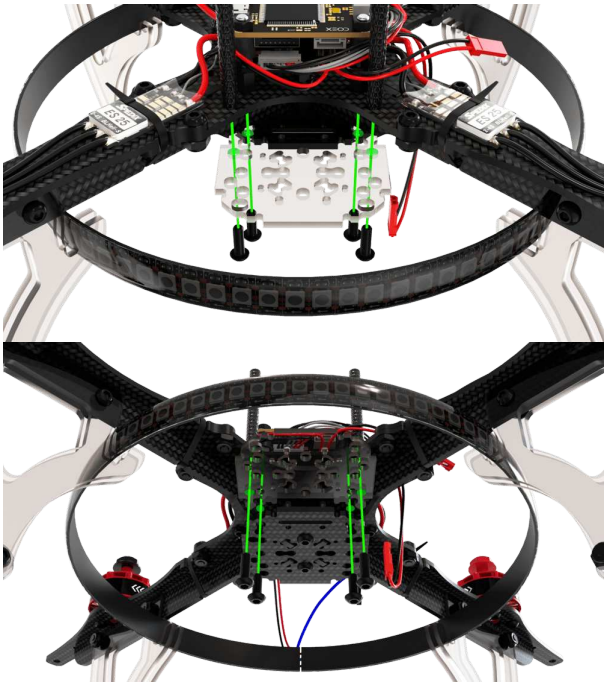
Где:

- YourRemote - название вашего пульта указанное при настройке
- YourKey - имя одной из заданных при настройке кнопок

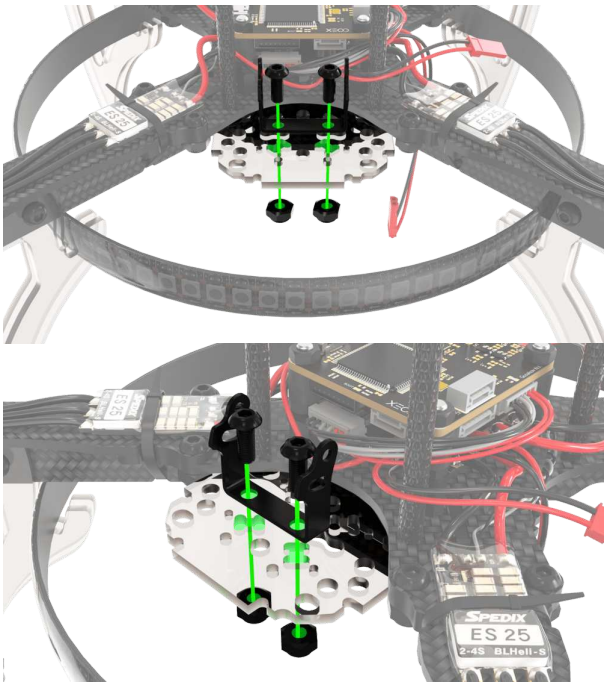
Установка и настройка FPV-оборудования

Подготовка и установка камеры и передатчика

1. Установите малую монтажную деку на основную раму.



2. Установите скобу для крепления камеры в соответствующие отверстия.



3. Обрежьте трехпиновый комплектный кабель камеры.





4. Залудите провода



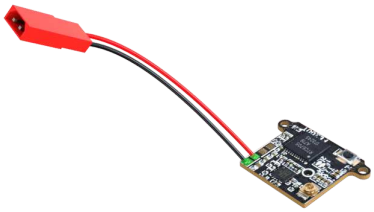
5. К силовым проводам камеры припаяйте разъем JST-папа.

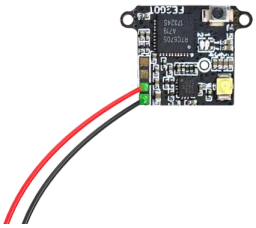




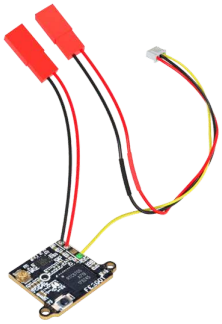
Перед спаиванием проводов не забудьте надеть термоусадку на провода.

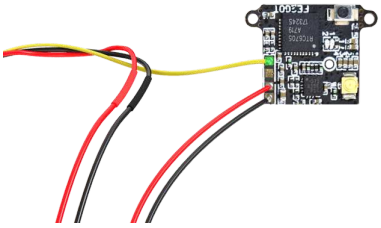
6. К передатчику припаяйте разъем JST-папа.



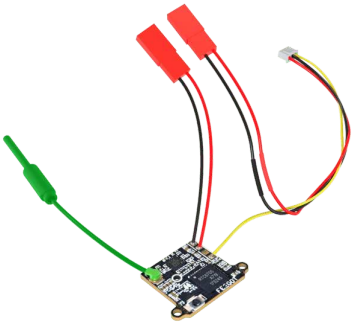


7. К передатчику припаяйте желтый сигнальный кабель камеры.



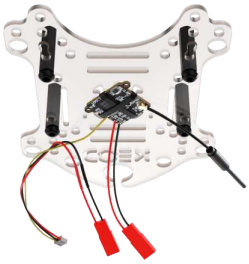


8. Подключите антенну к передатчику.

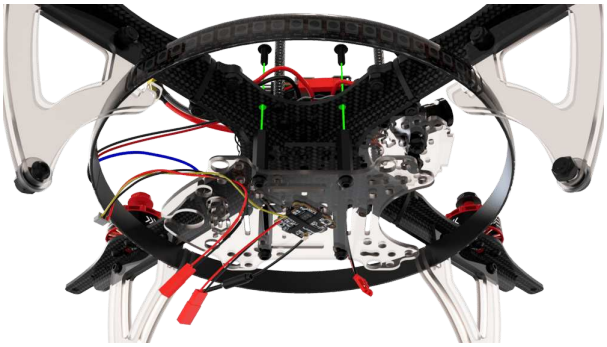


> Если на передатчик без антенны подать напряжение есть большая вероятность, что он сгорит.

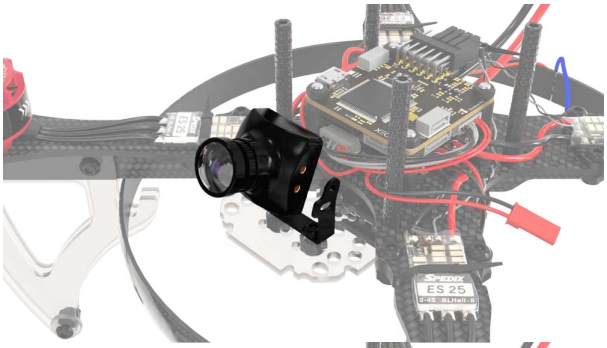
9. На монтажную деку установите приемник, закрепив его стяжками.

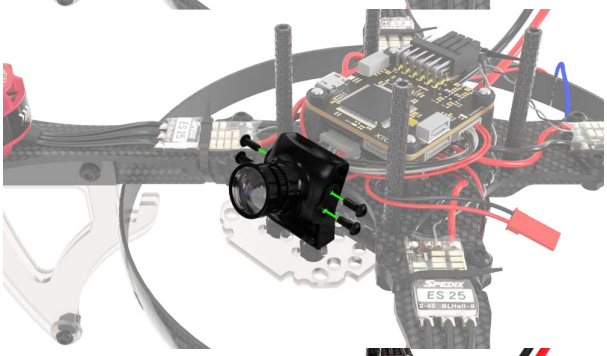


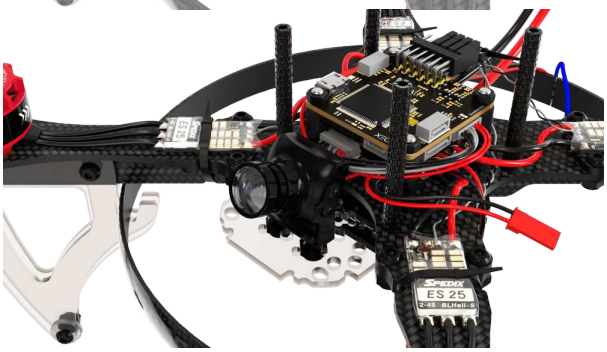
10. Установите монтажную деку вместе с приемником снизу коптера.



11. Установите камеру в скобу и закрепите ее с помощью 4-х комплектных болтов. Камера должна быть под углом 15°-20° относительно плоскости коптера.





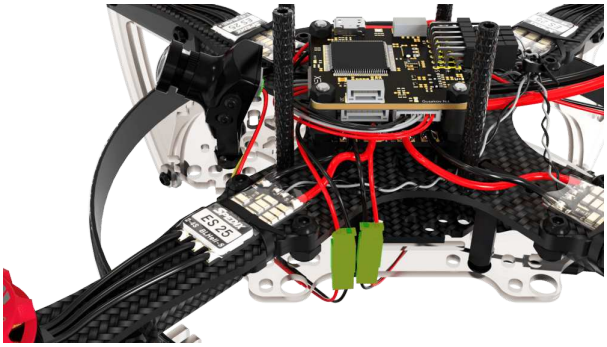




12. Подключите сигнальный кабель к камере.



13. Подключите кабель питания камеры к силовому JST, припаянному к площадкам *BAT+* и *GND* на плате распределения питания.
14. Подключите кабель питания передатчика к JST на 5В.



Настройка и подключение FPV-очков

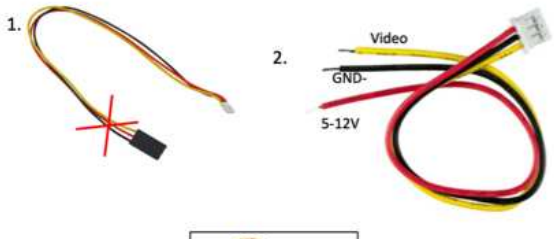
1. Установите на очки две комплектные антенны.
2. Включите очки удерживая кнопку питания 3–4 секунды.
3. Включите коптер и убедитесь, что светодиод передатчика светится синим цветом.
4. Нажмите на очках кнопку *Auto Search*, для автоматического поиска доступного радиоканала.

Установка FPV

Подготовка курсовой камеры

1. Взять провод-коннектор от камеры и откусить ЧЁРНУЮ сторону 3-х пинового разъема.
2. Подготовить провода провода к подключению:
    i. Укоротить провода до нужной длины *.
    ii. Зачистить (снять 2мм термоизоляции с конца провода, не повредив жилы).
    iii. Скрутить провода.
    iv. Залудить, используя пинцет.

* Длину нужно определить заранее, между платой распределения питания и предположительным место установки камеры!



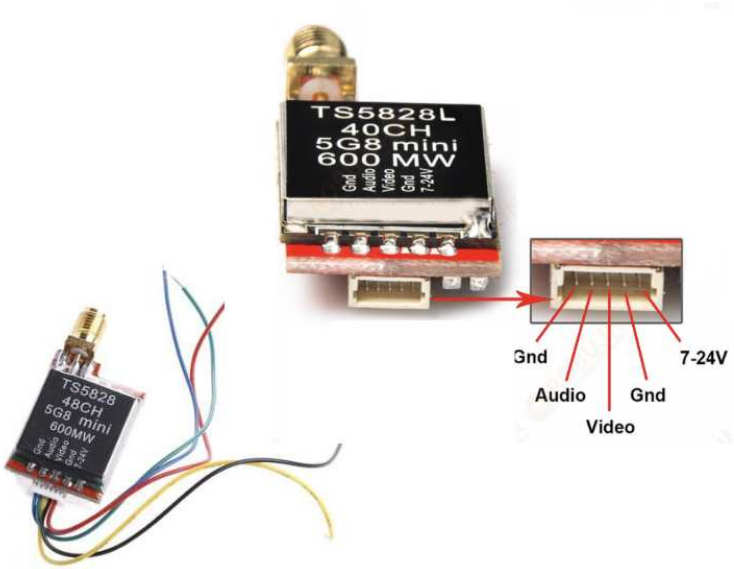


Курсовая камера
- Питание: DC5-12V
- Видео Выход: 1.0Vp-p, 75 Ом

Подготовка передатчика

Аналогичную процедуру проводим и здесь:

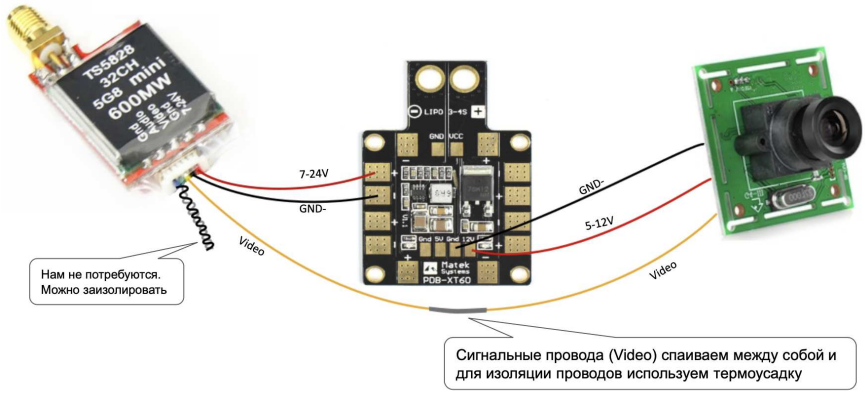
1. Взять провод-коннектор от передатчика и откусить ЧЁРНУЮ сторону 5-ти пинового разъема.
2. Подготовить провода провода к подключению:
    i. Укоротить провода до нужной длины *.
    ii. Зачистить (снять 2мм термоизоляции с конца провода, не повредив жилы).
    iii. Скрутить провода.
    iv. Залудить, используя пинцет.
3. Длину нужно определить заранее, между платой распределения питания и предположительным место установки передатчика!



Передатчик
Питание: DC7-24V

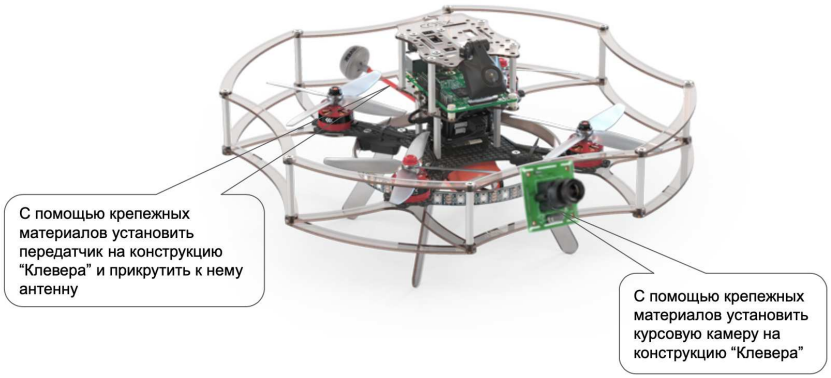
Подключение FPV

Готовые коннекторы вставить в соответствующие разъёмы и запаять провода питания на плату распределения питания согласно схеме:



> В данной схеме питание камеры идёт на 12V (Однако возможно использовать 5V). Питание передатчика идёт на питание регулятора (однако возможно использовать 12V).

Установка компонентов FPV

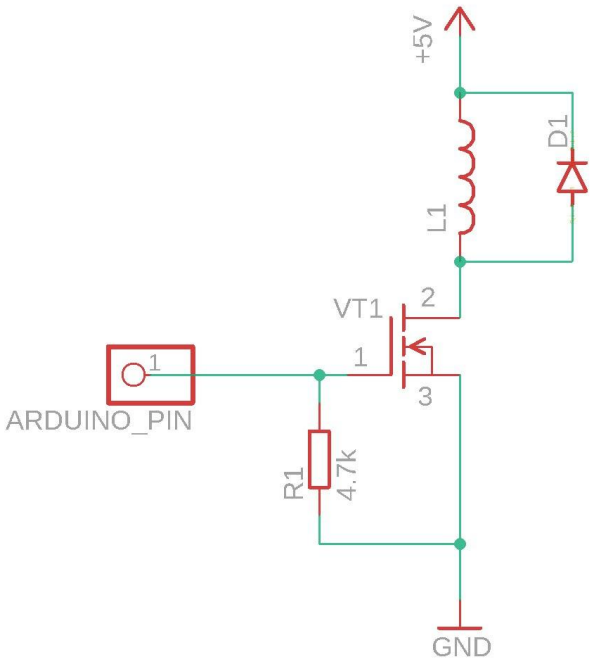


В качестве крепежных материалов можно использовать:

1. Термоклей;
2. изоленту;
3. стяжки (хомуты);
4. двусторонний скотч.

Сборка и настройка электромагнитного захвата

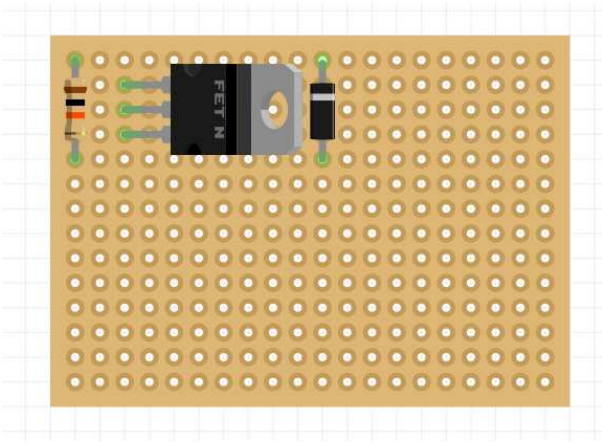
Магнитный захват можно собрать различными способами в соответствии с электрической схемой.



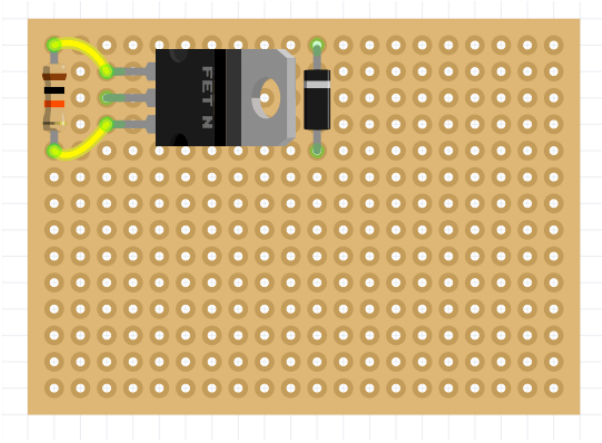
Ниже представлен пример сборки схемы электромагнитного захвата на макетной плате.

> Рекомендуется проложить проводку между элементами с обратной стороны платы (на дальнейших изображениях проводка сделана поверх схемы, для наглядности).

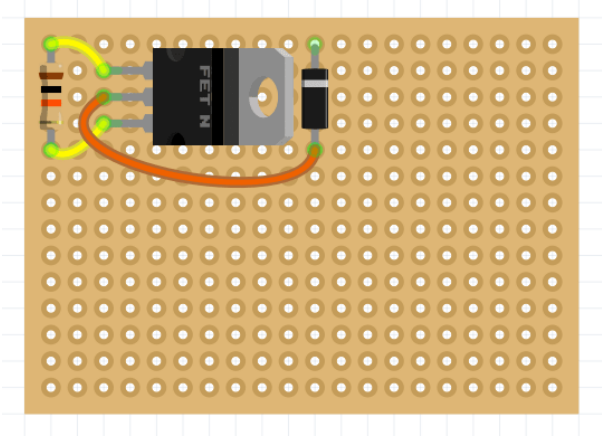
1. На паечной плате разместите диод Шоттки, резистор на 10 кОм и транзистор.



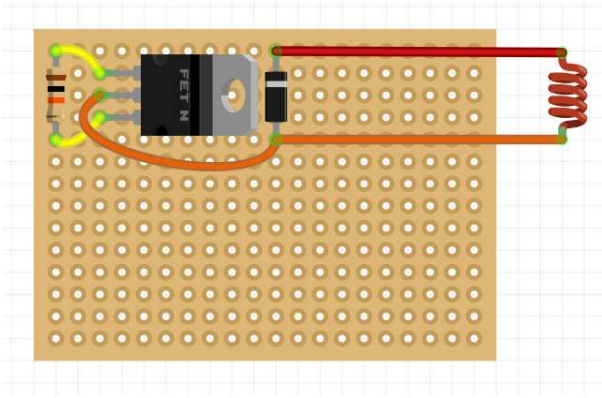
2. Припаяйте контакты с другой стороны платы и откусите оставшиеся ножки элементов.
3. Соедините контакты резистора и двух крайних ножек транзистора.



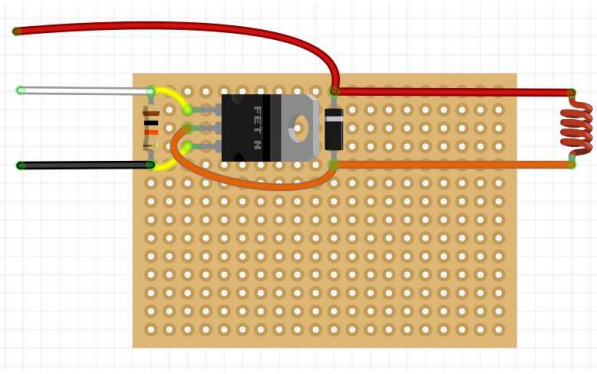
4. Соедините центральную ножку транзистора и ножку диода Шоттки (противоположную серой маркировочной полоске).



5. Обрежьте необходимое количество провода магнитного захвата и припаяйте его к контактам диода Шоттки.

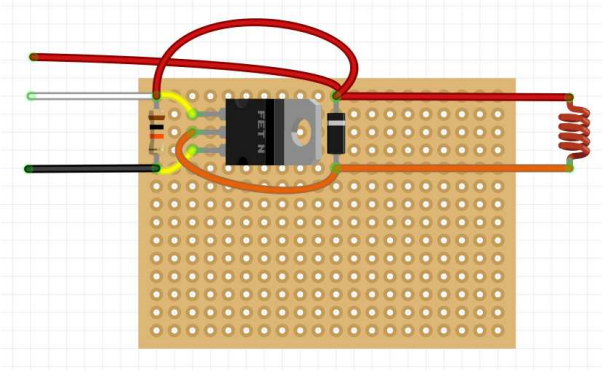


6. Припаяйте провода *Dupont*-папа к ножке транзистора и диода (красный, черный провода), а также провод *Dupont*-мама на противоположную ножку транзистора (белый провод).



Проверка работы электромагнитного захвата

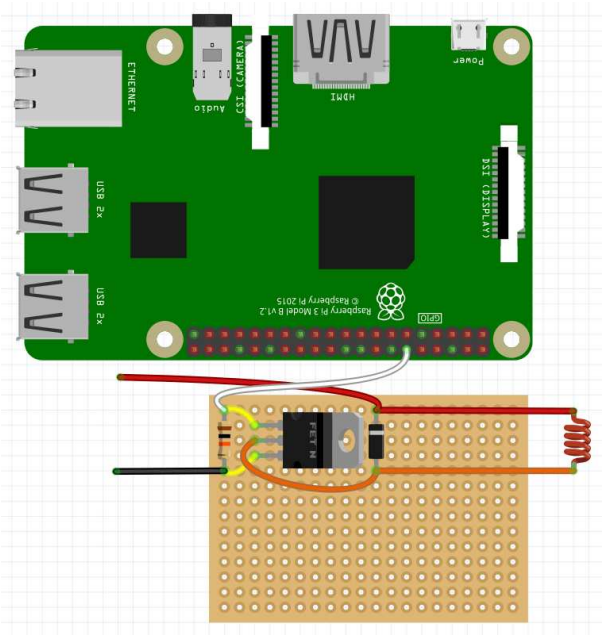
Для того, чтобы проверить работу захвата, подайте на сигнальный провод напряжение 5В. Для этого можно использовать провод *Dupont* папа-папа.



После подачи напряжения магнит должен включиться.

Подключение к Raspberry Pi

Подключите магнитный захват к плате Raspberry Pi для программного использования

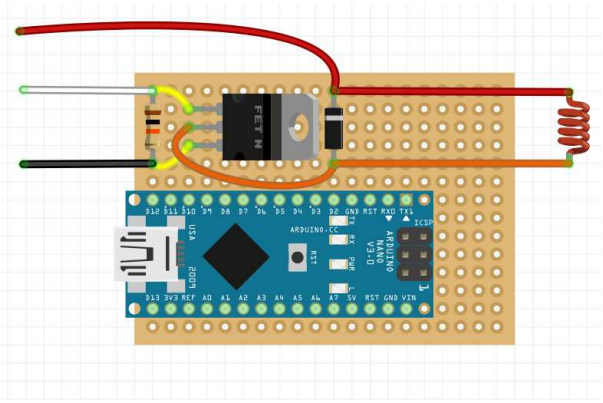


Пример кода, активирующего магнитный захват, можно посмотреть тут.

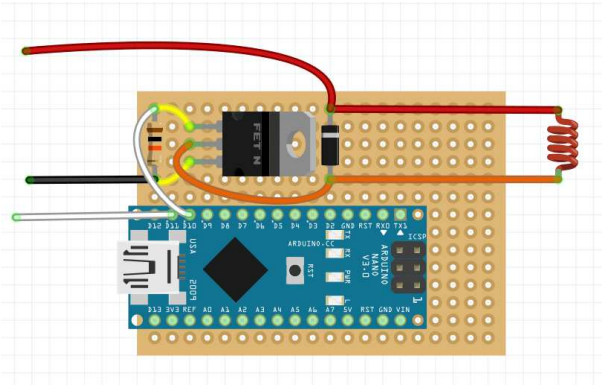
Подключение к Arduino

Подключите захват плате Arduino Nano, чтобы использовать его в ручном режиме.

Удобно ее располагать на той же паечной плате -- вставьте ее в подходящие отверстия и припаяйте с обратной стороны к плате.



Затем подключите сигнальный выход схемы к выбранному порту и припаяйте провод *Dupont*-мама к выбранному сигнальному порту на плате.



Установка электромагнитного захвата

1. В центральное отверстие на деке захвата установите электромагнит.
2. Стяжкой притяните собранную схему к обратной стороне деки.
3. Сигнальный вывод Arduino *D11* вставьте в один из выводов *AUX* на полетном контроллере.
4. Вставьте силовой вывод электромагнитного захвата в JST 5B.

Настройка электромагнитного захвата

Для управления магнитом через плату Arduino Nano, используйте код ниже:

```
void setup() {
  pinMode(11, INPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  if (int duration = pulseIn(11, HIGH) > 1200) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
}
```

Для однозначного определения статуса магнитного захвата, можно подключить светодиодную ленту типа *ws281x* (входит в наборы "Клевер"). Подключите ее к питанию +5v – 5v, земле GND – GND и сигнальный контакт DIN – Arduino D12.

Для управления магнитным захватом со светодиодной лентой через плату Arduino Nano используйте код ниже:

```
#include <Adafruit_NeoPixel.h>
#define NUMPIXELS 72
#define PIN 12
int pin = 11;
int led = 13;

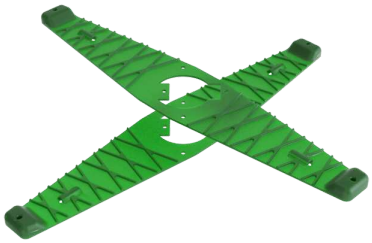
unsigned long duration;
Adafruit_NeoPixel strip (NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin();
  strip.setBrightness(10);
  Serial.begin(9600);
  pinMode(pin, INPUT);
  pinMode(led, OUTPUT);
}

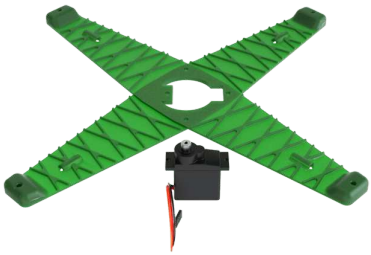
void loop() {
  duration = pulseIn(pin, HIGH);
  Serial.println(duration);
  delay(100);
  if (duration >= 1500) {
    digitalWrite(led, HIGH);
    for (int i = -1; i < NUMPIXELS; i++) {
      strip.setPixelColor(i, strip.Color(255, 0, 0));
      strip.show();
    }
  } else {
    digitalWrite(led, LOW);
    for (int i = -1; i < NUMPIXELS; i++) {
      strip.setPixelColor(i, strip.Color(0, 255, 0));
      strip.show();
    }
  }
}
```

Сборка и настройка механического захвата

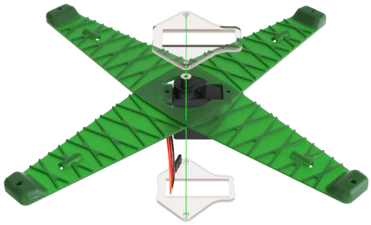
1. Совместите главные пластины захвата.



2. Установите сервопривод в соответствующий паз в центре пластин, таким образом, чтобы осевая шестерня находилась посередине захвата.



3. Прижмите пластины захвата небольшими проставками.

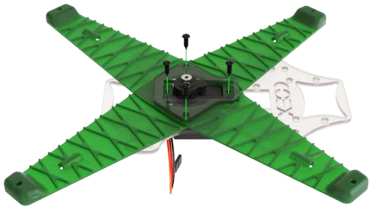




4. Установите деку захвата таким образом, чтобы крепежные отверстия в захвате совпадали с отверстиями для саморезов в пластине.



5. Закрепите конструкцию захвата саморезами.



6. Поверните шестерню сервопривода в крайнее положение.

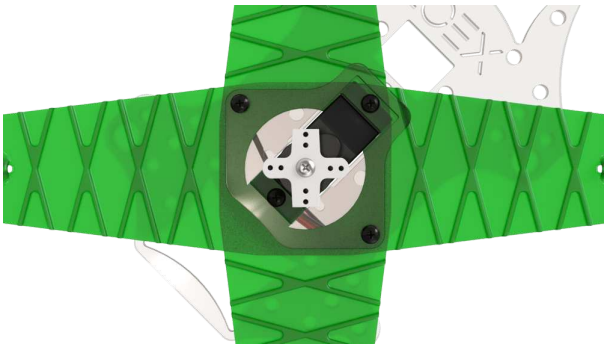


7. Установите на шестерню крестообразное крепление и закрепите его помощью винта, прилагающегося к сервоприводу.

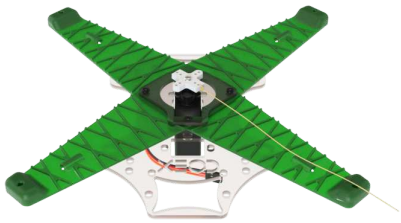


8. Обрежьте крестообразное крепление.



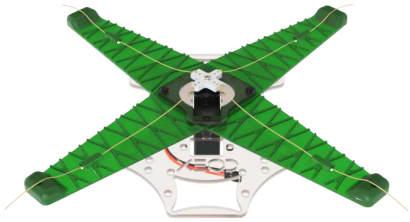


9. Завяжите сервоприводную нить таким образом, чтобы оставалось 2-3 см запаса.





10. Проденьте сервоприводную нить в соответствующие натягивающие пазы.



11. Закрепите клешни захвата маленькими саморезами таким образом, чтобы их угол составлял 25°–40°.

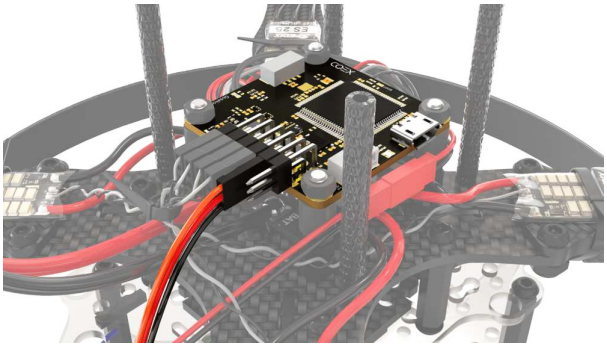


12. Установите собранный захват на коптер снизу.





13. Протяните кабель сервопривода и вставьте его в выход *AUX* 1-2 на полетном контроллере.



14. Для того, чтобы настроить управление захватом с пульта, зайдите во вкладку *Radio*.
15. В параметре *AUX 1/2 Passthrough RC channel* укажите необходимый вам канал.
16. Теперь при переключении тумблера соответствующего канала захват будет закрываться или открываться.

Груз с подставкой для магнитного захвата

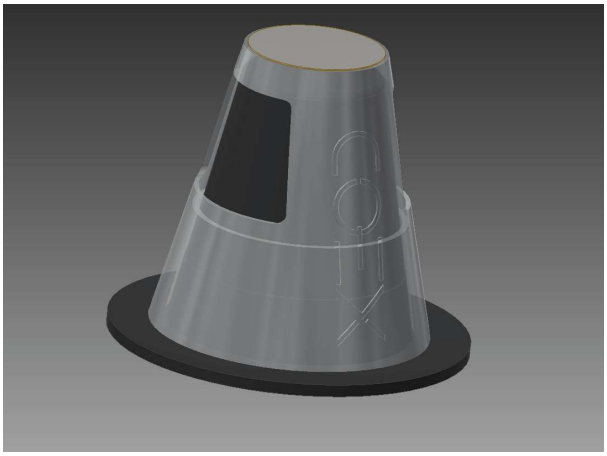
Груз с подставкой для использования в работе с квадрокоптером COEX Клевер 4 WorldSkills Russia.

Грузом для магнитного захвата является некая геометрическая фигура, которая имеет внутренний и внешний конус и имеет форму стакана с вырезами для облегчения.

Технические характеристики

Груз в сборе с подставкой и пластиной для зацепа магнитного захвата

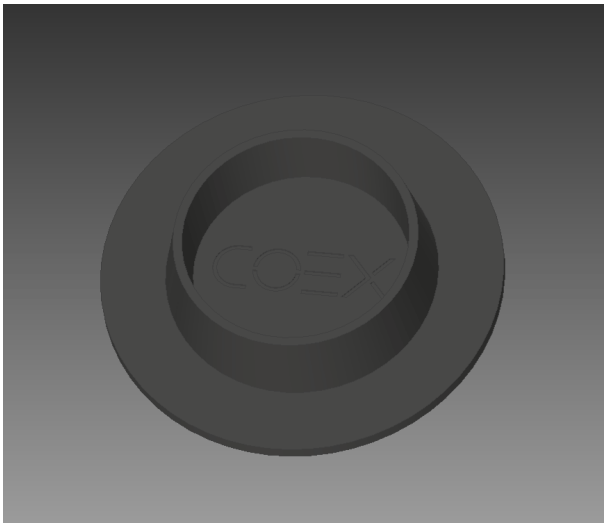
Груз в сборе с подставкой и пластиной для зацепа магнитного захвата выглядит следующим образом:



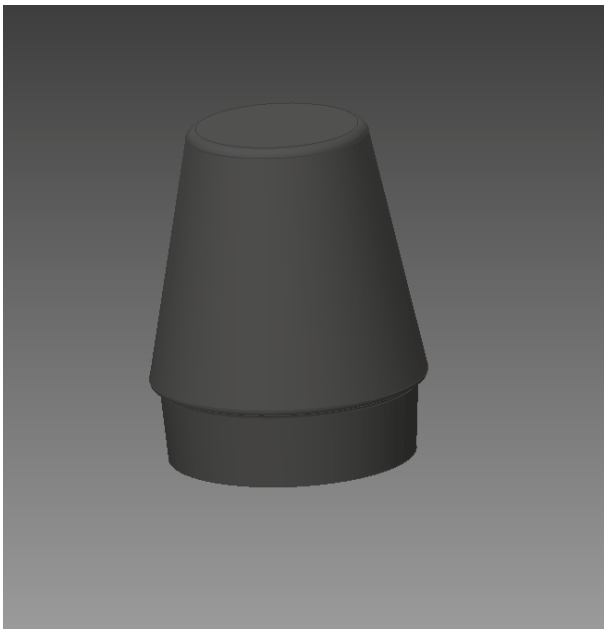
Подставка под груз для магнитного захвата

Подставка под груз для магнитного захвата имеет разборную конструкцию и состоит из двух частей.

Первая - это низ:



Вторая - это дополнение, чтобы добрать необходимую высоту для точного позиционирования груза на подставке:

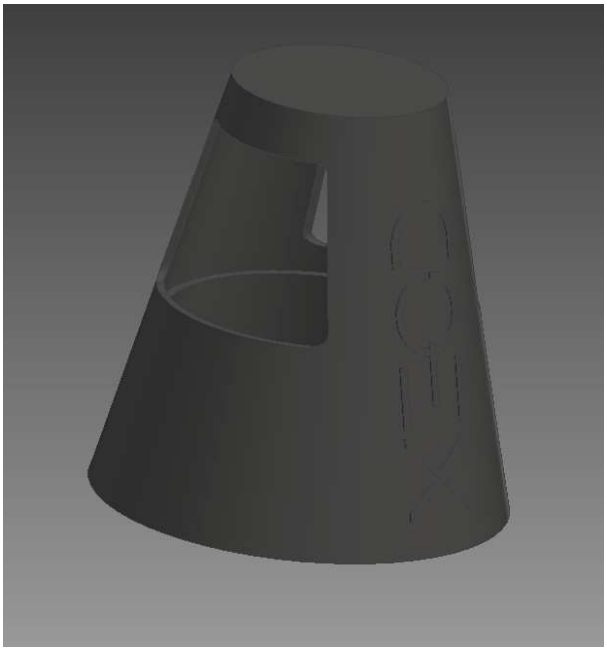


В сборе это выглядит так:



Груз для подставки

Сам груз отдельно от подставки выглядит следующим образом:



Вес при 100% заполнении из материала PETG при печати на 3D принтере составляет 35 грамм.

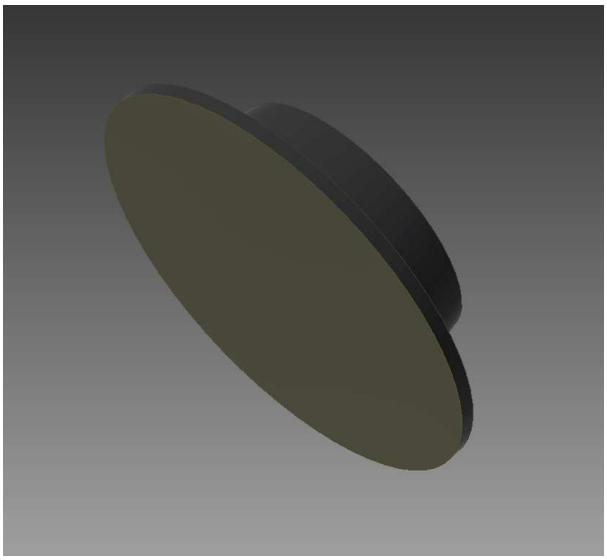
Высота груза 72.5 миллиметров.

Фиксация груза на захвате

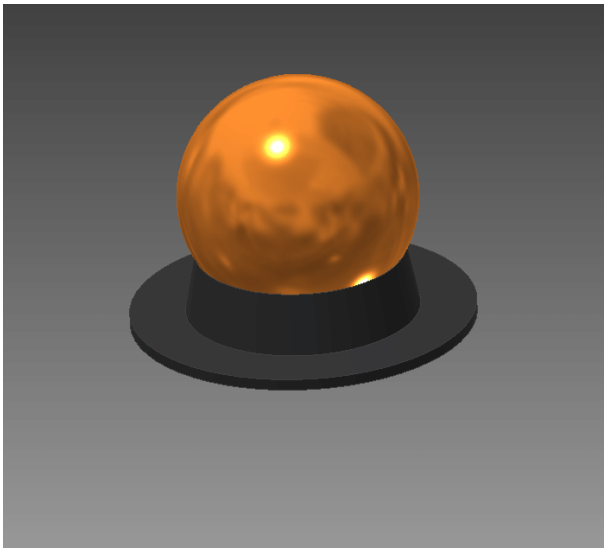
Для того, чтобы груз держался на магнитном захвате, на его верхнюю часть приклеивается комплектная пластина, и в сборке это выглядит следующим образом:



На дно подставки необходимо приклеить двусторонний скотч для дальнейшей фиксации подставки на используемой в дальнейшем поверхности:

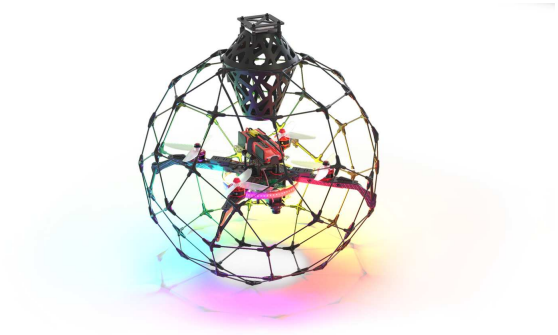


В случае использования нижней части с механическим захватом также необходимо приклеить к её основанию двусторонний скотч, чтобы подставку не сдувало воздушным потоком от пропеллеров. В сборке с теннисным мячиком это выглядит следующим образом:



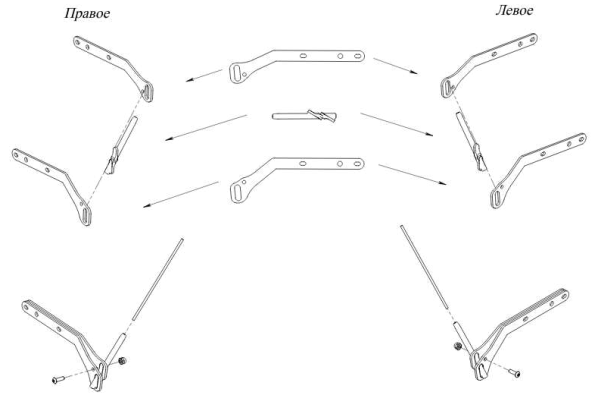
Ссылки для скачивания файлов для 3D-печати находятся в статье CAD-модели.

Сборка сферической защиты

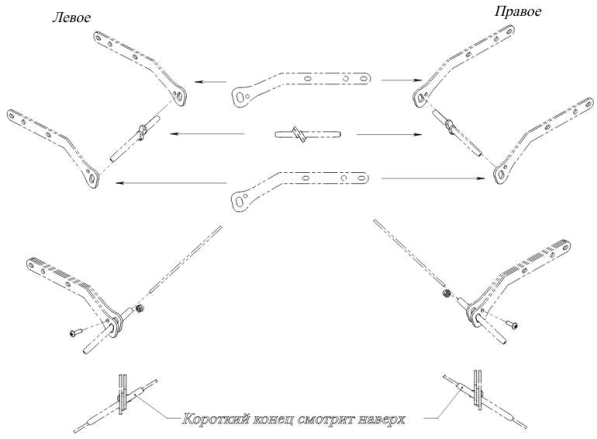


Сборка пояса сферы

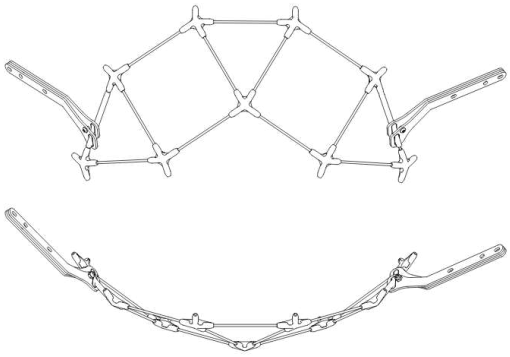
1. Соберите передние крепления коптера.



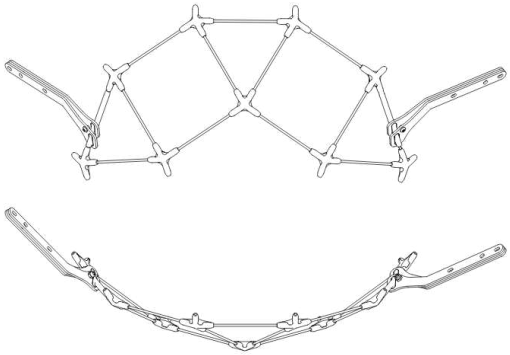
2. Соберите задние крепления коптера.



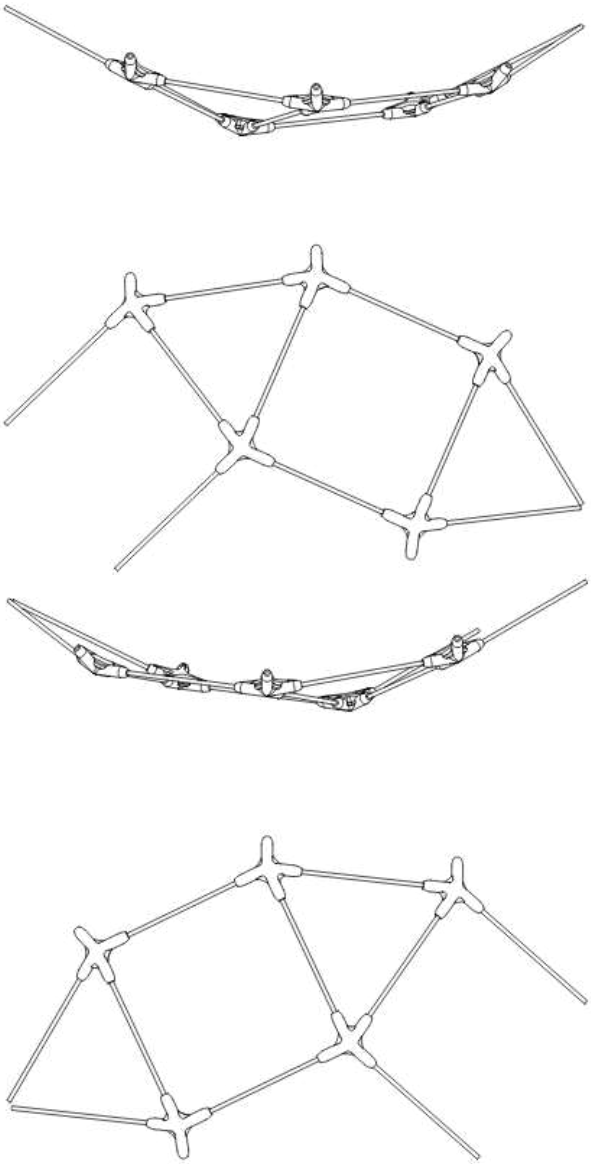
3. Соберите переднюю часть пояса.



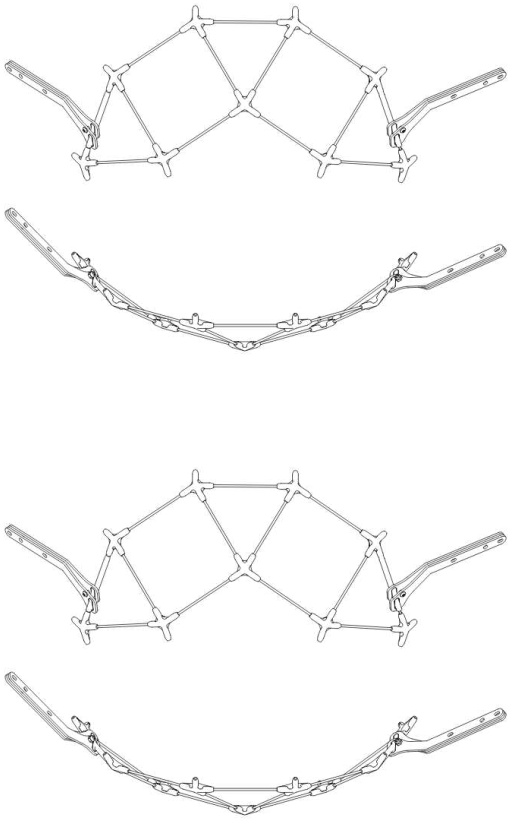
4. Соберите заднюю часть пояса.



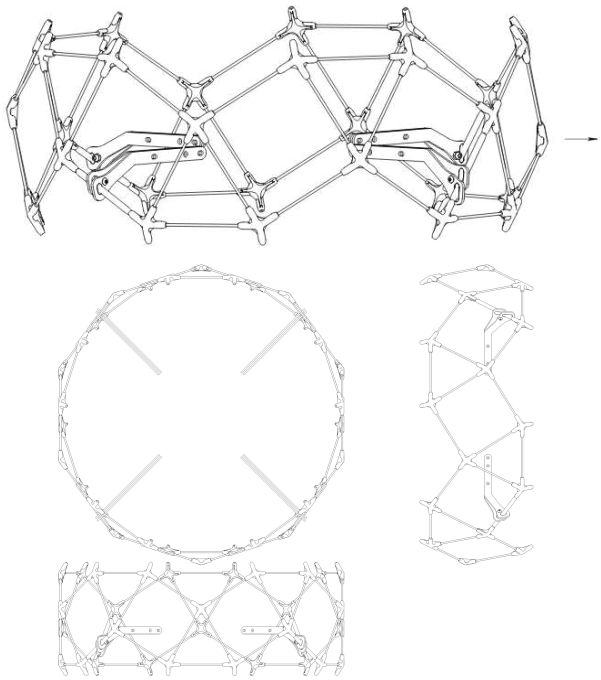
5. Соберите правую и левую части пояса.



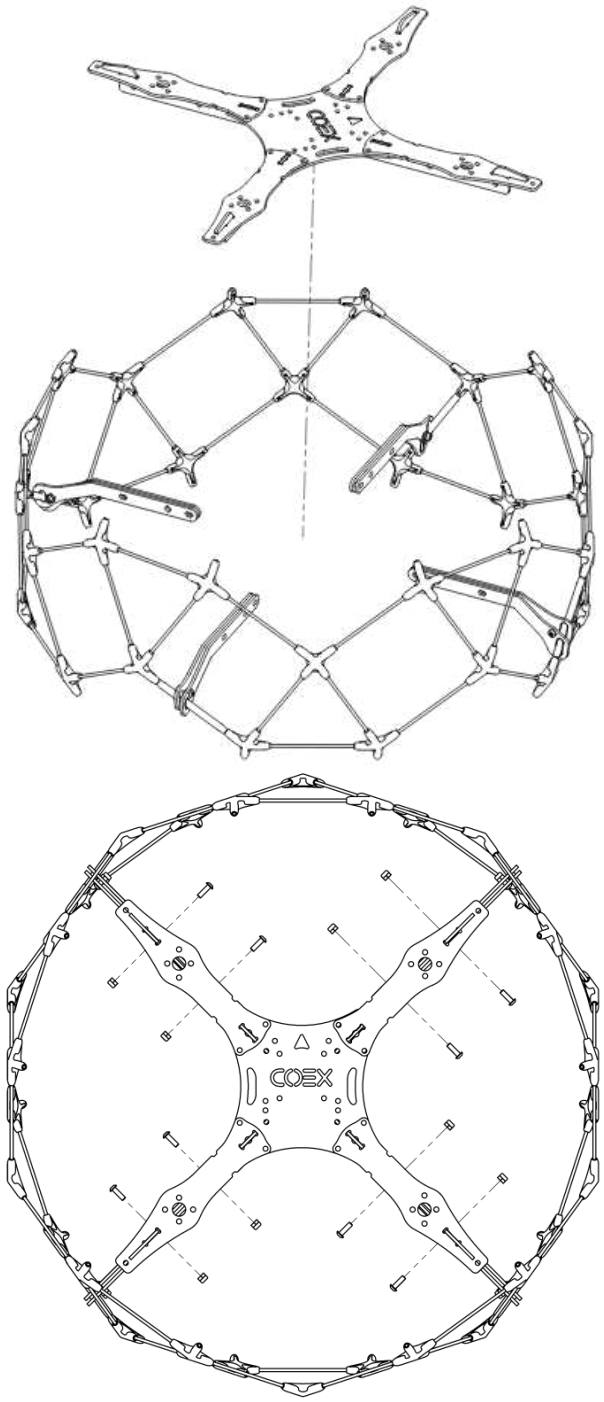
6. Соберите пояс сферы (виды сверху и сбоку).



7. Пояс сферы в сборе.

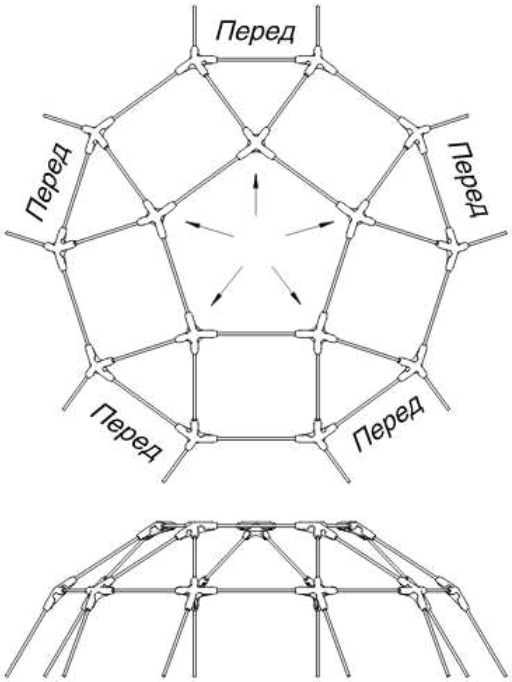


8. Закрепите коптер внутри пояса.



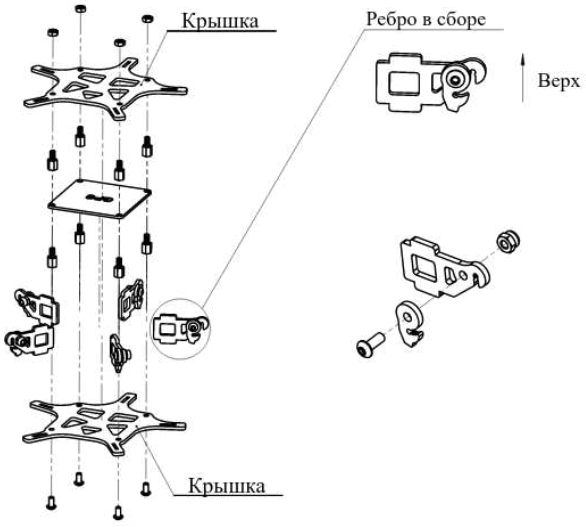
Сборка верхней части сферы

1. Соберите верхнюю часть сферы.

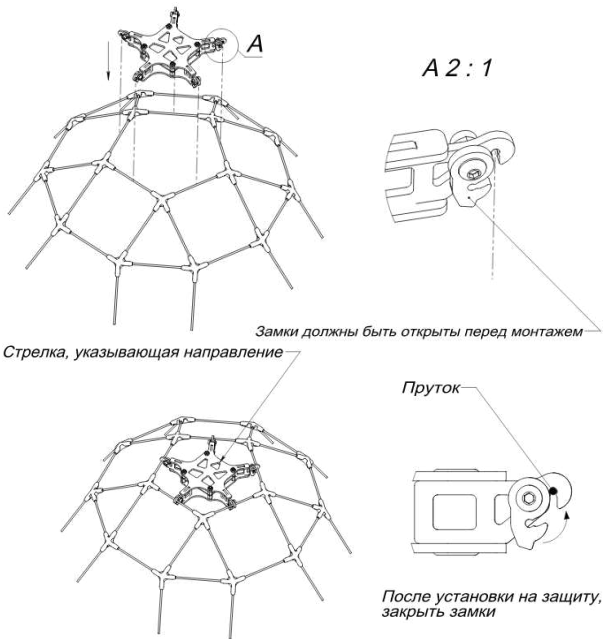


Установка GPS модуля

1. Соберите крепление GPS.

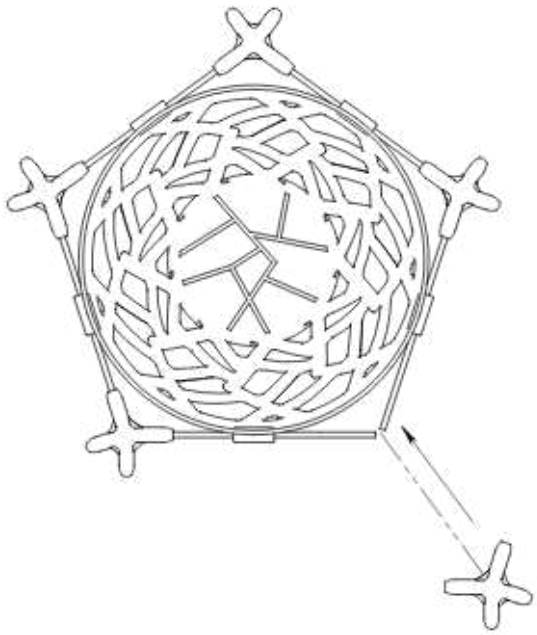


2. Установите крепление GPS на верхнюю часть сферы.

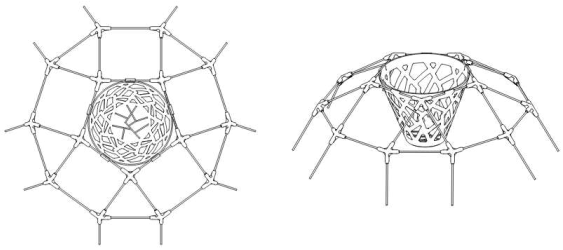


Установка корзинки

1. Проденьте прутки в специальные пазы в корзинке и соедините их коннекторами.

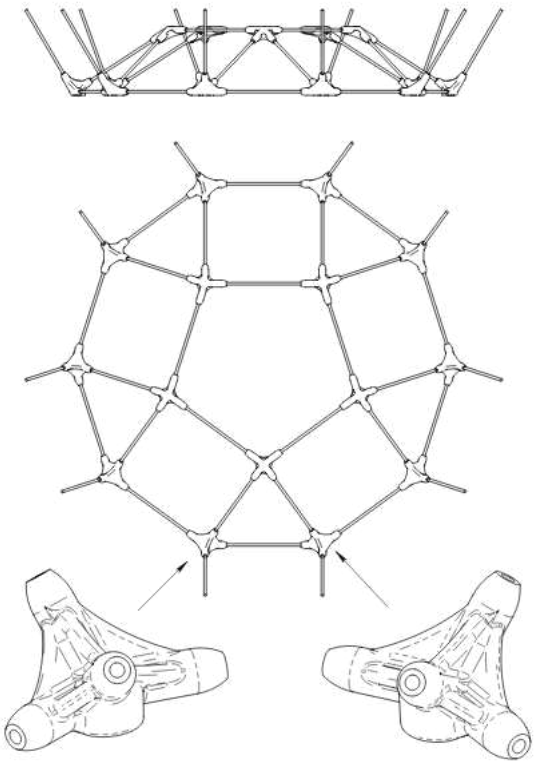


2. Установите корзинку в середину верхней части сферы.



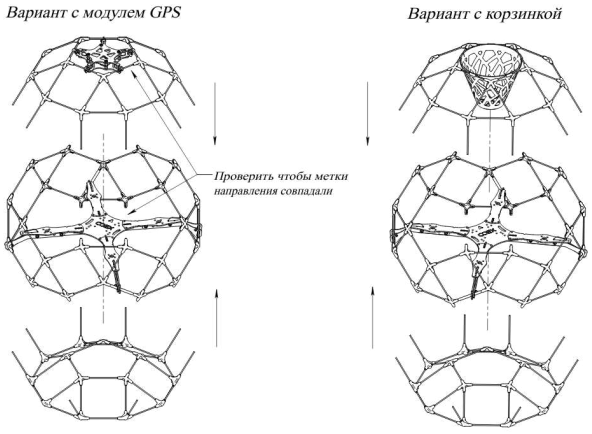
Сборка нижней части сферы

1. Соберите нижнюю часть сферы.



Зеркальные нижние коннекторы

2. Соберите шаровую защиту из 3-х частей.

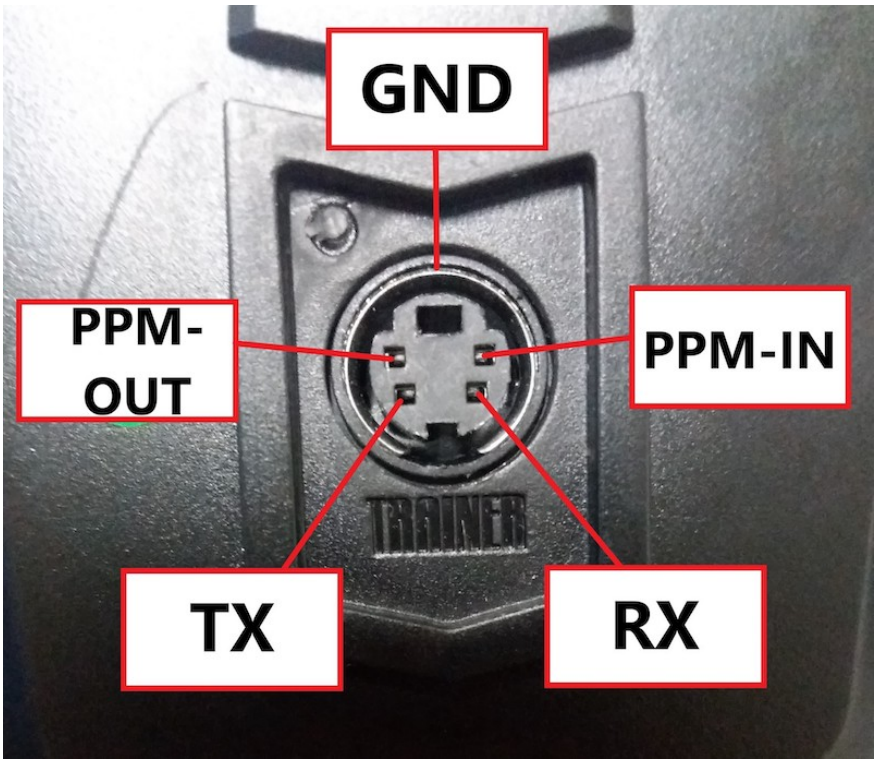

Вариант с модулем GPS
Вариант с корзинкой
Проверить чтобы метки направления совпадали

Настройка режима тренера на пульте FlySky

Режим тренера используют для обучения пилотированию. Чтобы во время обучения ученик ничего не повредил, на перехвате стоит опытный пилот, чтобы при необходимости взять управление на себя.

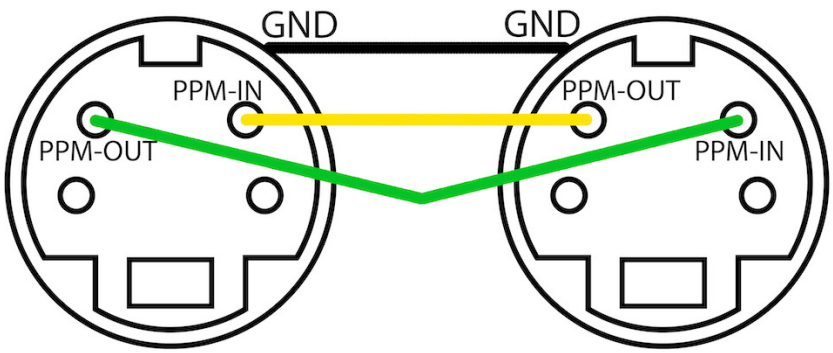
Для этого при помощи провода соединяют два пульта: один из них - пульт тренера, а другой – ученика.

Схема провода



Для создания канала общения между пультами используется разъём сзади корпуса (S-Video). Три контакта в разъёме отводятся для приёма, передачи информации и заземления. Контакт PPM-OUT (передача) должен быть соединён с PPM-IN (приём) и наоборот. Во избежание помех внешней среды контакты ground должны быть соединены между собой.

Таким образом нам нужно припаять провода к разъёму.



Настройка пульта учителя

Зайдите в настройки (зажмите кнопку OK). Далее зайдите в системные настройки (System setup) и найдите (Up/Down) режим тренера (Trainer mode).

Активируйте режим: для этого в строке Mode должен стоять параметр On. Чтобы изменить параметр, используйте кнопки Up/Down. Чтобы сохранить параметр, нажмите OK.

Теперь выбираем переключатель для передачи управления.

Это можно сделать в меню режима (Trainer mode). В строке Switch выберите (менять можно с помощью Up/Down) любую удобную клавишу (SwA, SwB, SwC, SwD). С помощью этой клавиши вы сможете перехватить управление. Убедитесь, что на эту клавишу не назначены другие функции.

Чтобы сохранить настройки зажмите Cancel.

> На пульте учителя должен стоять тот же режим полёта, что и на пульте ученика.

Настройка пульта ученика

Зайдите в настройки, System setup и выберите режим ученика (Student mode). Далее нажмите OK и при помощи Up/Down выберите подтверждение (Yes).

Зажмите Cancel, чтобы сохранить настройки.

Если всё настроено правильно, на главном экране появится буква S.

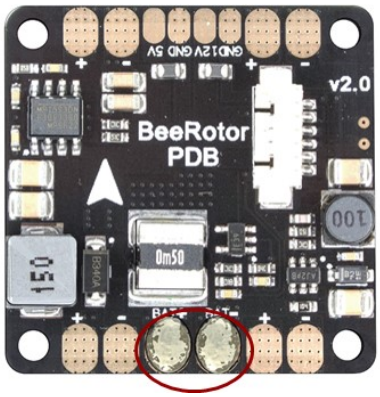
Лужение

> Перед пайкой и лужением убедитесь, что провода и платы отключены от питания (обесточены)!

Лужение контактных площадок

Залудить контактную площадку значит:

1. Нанести флюс на контактную площадку.
2. Покрыть припоем контактную площадку.



1. Залудить контактные площадки BAT+ и BAT-
2. Залудить остальные контактные площадке

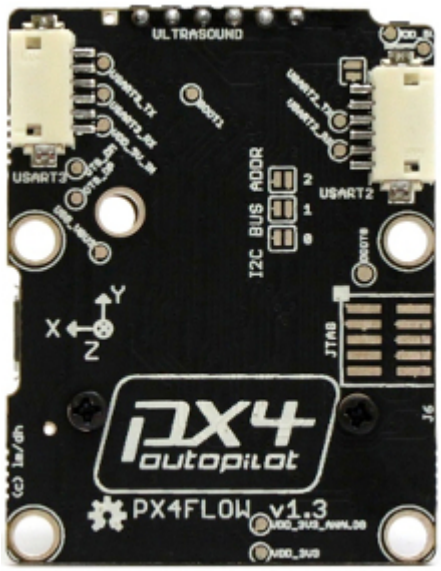
Лужение проводов

Залудить провод значит:

1. Зачистить провод — снять слой изоляции.
2. Скрутить оголенные провода.
3. Нанести флюс на скрученные оголенные провода.
4. Покрыть слоем припоя.



Смарт-камера PX4FLOW





PX4FLOW – смарт-камера, реализующая алгоритм вычисления "оптического потока" (optical flow) и способная передать полученные данные в полётный контроллер. Optical flow часто применяется для полётов коптера в помещении.

Спецификации

Модуль PX4FLOW содержит:

- микропроцессор Cortex M4F (168 MHz, 128 + 64 KB RAM);
- светочувствительную матрицу MT9V034 (разрешение: 752x480 точек);

- трёхосевой гироскоп L3GD20.

Для светочувствительной матрицы используется объектив 16мм M12 с установленным ИК-фильтром.

Габариты модуля составляют 45.5x35x25 мм (в зависимости от настройки объектива).

Модуль использует питание 5 В и потребляет до 115 мА.

Интерфейсы:

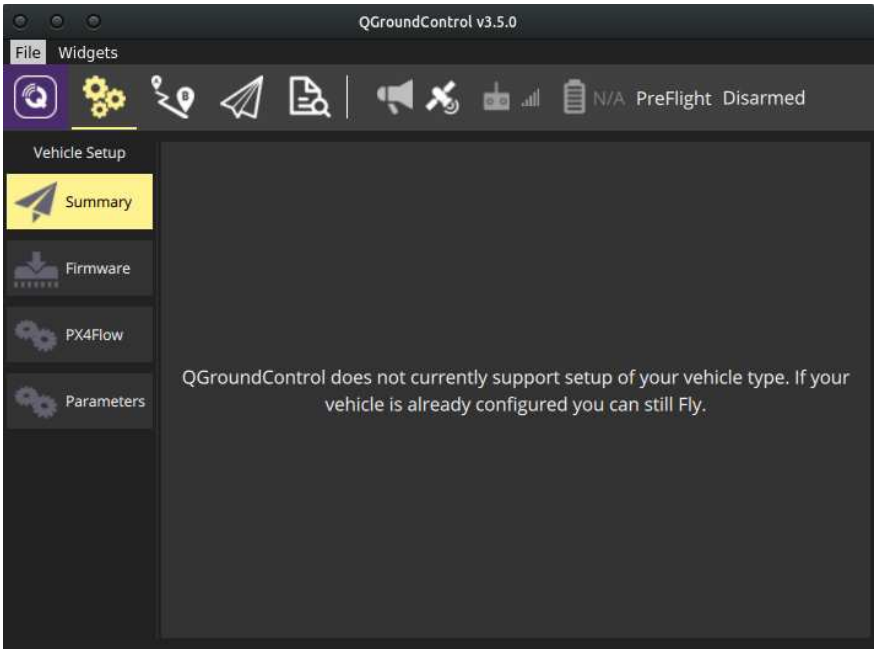
- USB (разъём microUSB) – используется для программирования и первоначальной настройки;
- I2C (разъём Hirose DF13 4 pos) – используется для передачи данных на полётный контроллер;
- USART2, USART3 (разъёмы Hirose DF13 6 pos) – могут использоваться для подключения к полётному контроллеру и последовательного соединения нескольких модулей.

> Использование Optical Flow требует наличия дальномера!

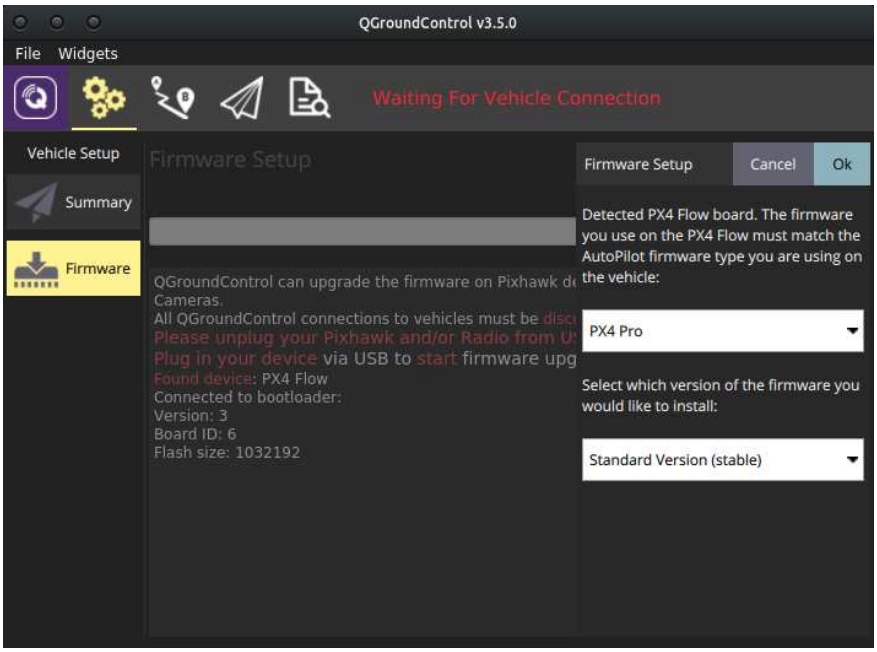
На некоторых моделях PX4FLOW может быть установлен сонар Maxbotix LZ-EZ4. В этом случае его показания будут пересылаться вместе с показаниями камеры.

Первоначальная настройка

Настройка камеры производится с помощью программы QGroundControl. Запустите её и подключите модуль PX4FLOW к компьютеру по USB. Откройте режим `Vehicle setup` (иконка с шестерёнками); окно QGroundControl при этом будет выглядеть примерно так:



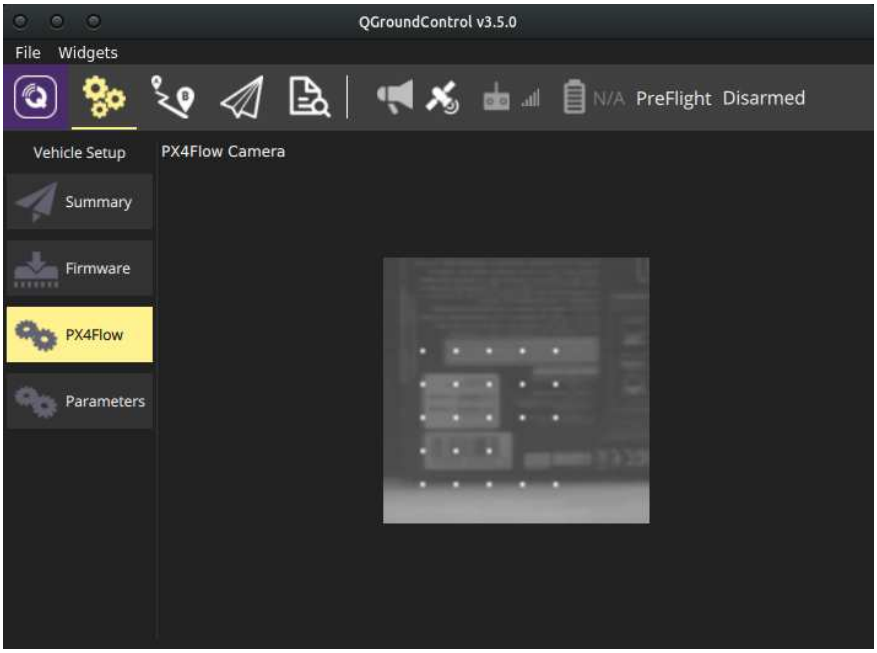
Если это первое включение камеры, то вам надо будет произвести обновление её прошивки по аналогии с перепрошивкой полётного контроллера.



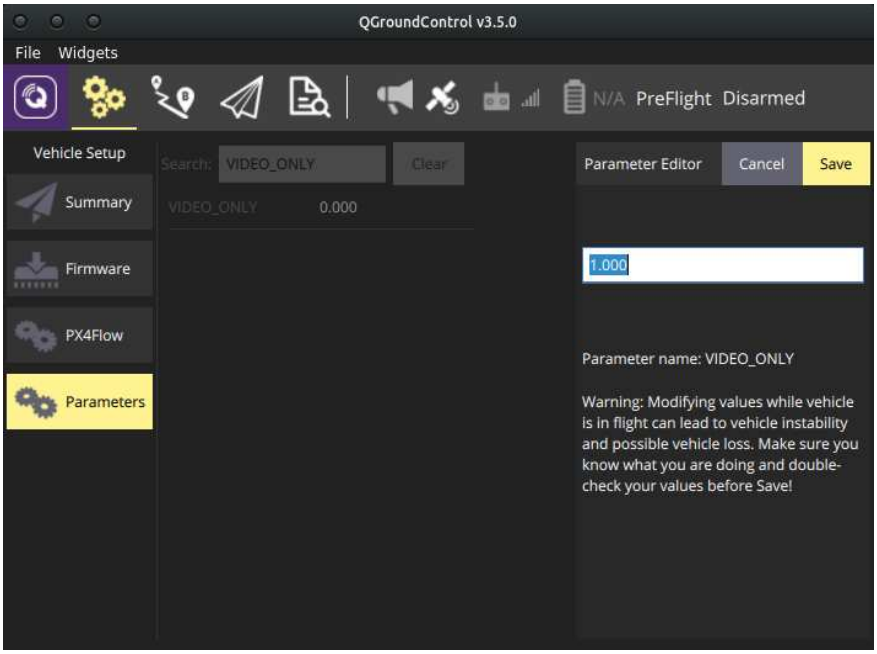
> При перепрошивке PX4FLOW следует выбрать прошивку `PX4 Pro` при использовании полётного контроллера на базе `PX4` и `ArduPilot` при использовании полётного контроллера с прошивкой `ArduPilot`.

Фокусировка камеры

Для оптимальной работы модуля PX4FLOW следует настроить фокус его камеры на предполагаемое расстояние от пола, при котором будут осуществляться полёты. Это можно сделать в QGroundControl в режиме `Vehicle setup` во вкладке `PX4Flow`. Там вы сможете увидеть изображение с камеры:



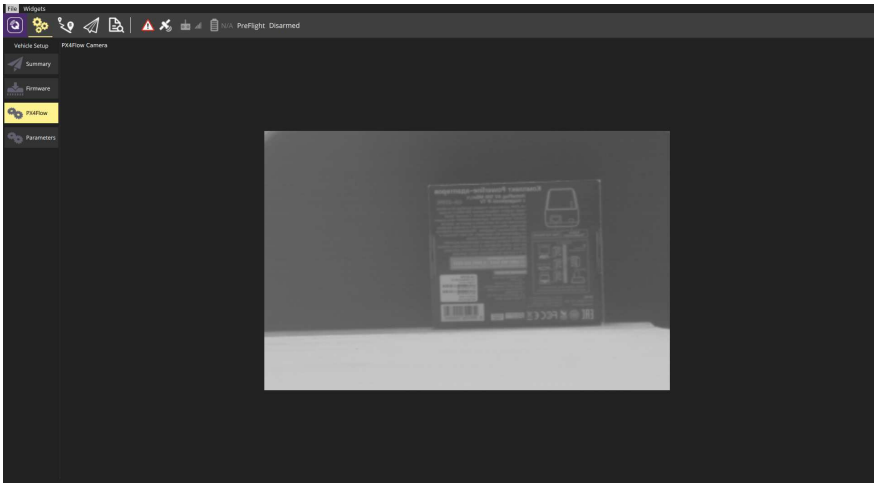
Для калибровки камеры лучше использовать специальный режим `Video only`. Для его включения зайдите во вкладку `Parameters`, найдите параметр `VIDEO_ONLY` и установите его в значение `1`



Фокусировку следует проводить в хорошо освещённом месте, модуль камеры при этом следует закрепить. Для фокусировки рекомендуется использовать книгу или коробку с текстом. Расположите объект, на котором вы будете фокусироваться, на таком расстоянии, на котором вы предполагаете летать над полом.

В режиме калибровки камера будет выдавать изображение большего разрешения, чем при штатной работе, но с меньшей частотой кадров.

При работе в режиме калибровки рекомендуется развернуть окно QGroundControl на весь экран, чтобы лучше видеть изображение:



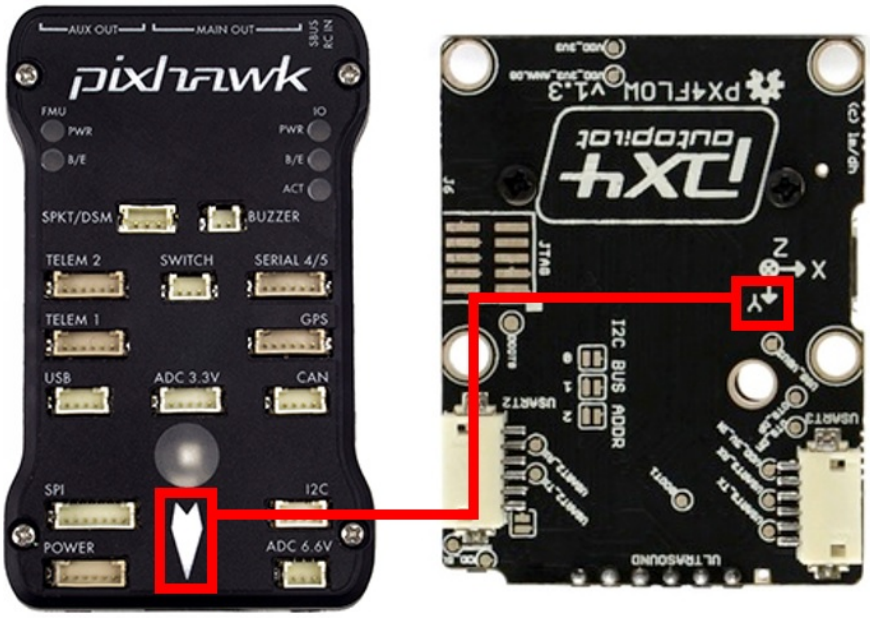
> Изображение с камеры будет отражено по вертикальной оси. Это является нормальным поведением и не должно быть поводом для беспокойства.

Отпустите контргайку на объективе и медленно вращайте его, пока изображение не станет резким. После этого аккуратно закрепите объектив с помощью контргайки.

> Остальные параметры в данный момент не сохраняются, поэтому настройка камеры ограничивается её фокусировкой.

Установка и подключение

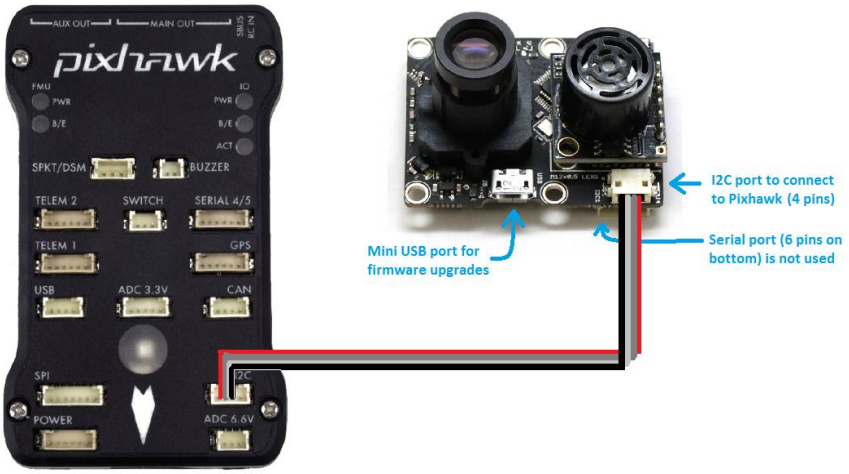
По умолчанию предполагается, что модуль PX4FLOW будет расположен так, что направление оси `y` будет совпадать со стрелкой на полётном контроллере:



Поворот модуля PX4FLOW относительно полётного контроллера указывается параметром `SENS_FLOW_ROT` в PX4. Его значение по умолчанию (`Yaw 270°`) соответствует размещению на иллюстрации.

Подключение к Pixhawk

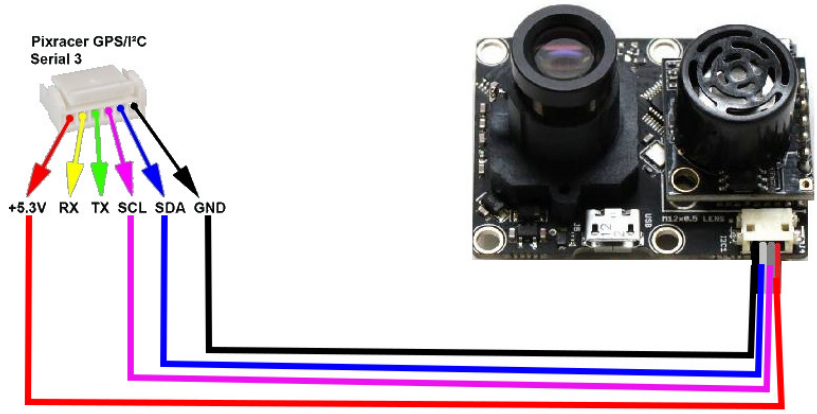
Подключите PX4FLOW с помощью поставляемых кабелей к разъёму I2C на Pixhawk:



Подключение к Pixracer

В полётном контроллере Pixracer используется совмещённый порт UART/I2C (помечен как GPS).

На более ранних версиях PX4FLOW разъём I2C может быть перевёрнут; ориентируйтесь на кабель, идущий в комплекте – на нём красным отмечен провод питания.



Для подключения PX4FLOW к Pixracer рекомендуется использовать GPS/I2C сплиттер

Настройка полётного контроллера

Для того, чтобы полётный контроллер использовал данные с PX4FLOW, следует установить соответствующие флаги в настройках estimator'а:

- Для **LPE** (`SYS_MC_EST_GROUP` = `local_position_estimator`): в `LPE_FUSION` включены флажки `fuse optical flow` и `flow gyro compensation` .
- Для **EKF2** (`SYS_MC_EST_GROUP` = `ekf2`): в `EKF_AID_MASK` включен флажок `use optical flow` .

Остальные настройки PX4FLOW описаны в статье по Optical Flow. Значения по умолчанию должны обеспечить оптимальную работу PX4FLOW.

Типы силовых разъемов

XT-60

Один из самых надёжных силовых разъёмов, которые стараются применять на силовых аккумуляторах. Именно такие аккумуляторы используются на Li-Po аккумуляторах для коптеров.



T-plug

Аналог XT-60. Имеет различные вариации для упрощения разъединения.



JST-XH или балансировочный разъем

Разъёмы данного типа часто применяются для балансировки отдельных элементов в составе сборки из нескольких литий-полимерных (Li-Pol), литий-ионных (Li-ion) или литий-фосфатных (LiFePO4) аккумуляторов. Подобные разъёмы с разным количеством штырьков устанавливаются в большинство современных зарядных устройств для балансировки литиевых элементов при заряде. Может использоваться в сочетании с Buzzer (пищалкой) для контроля заряда аккумулятора.



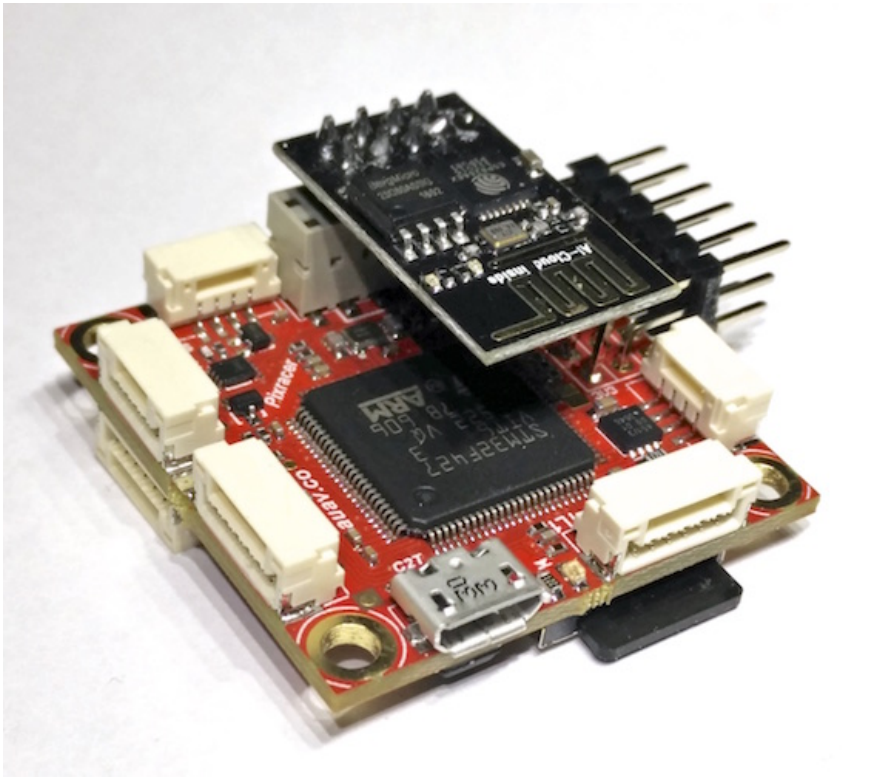
Gold Bullet Connector или "Бананы"

Существует великое множество штырьковых разъёмов типа Gold Bullet Connector. Разъёмы данного типа отличаются друг от друга диаметром и размером. Наиболее распространены разъёмы с диаметром коннектора 2 мм, 3 мм и 4 мм. Часто используется для создания беспаечных соединений на PDB и моторах.



Модуль ESP8266

> Более подробную информацию можно найти в основной статье в официальной документации и в репозитории с прошивкой для ESP8266.



Полётные контроллеры семейства Pixracer поддерживают подключение Wi-Fi модулей ESP8266. Эти модули можно использовать для подключения к полётному контроллеру с компьютера или планшета, даже если бортовой компьютер (например, Raspberry Pi) отсутствует или неисправен.

> Для Клевера предпочтительнее связываться с полётным контроллером через Raspberry Pi.

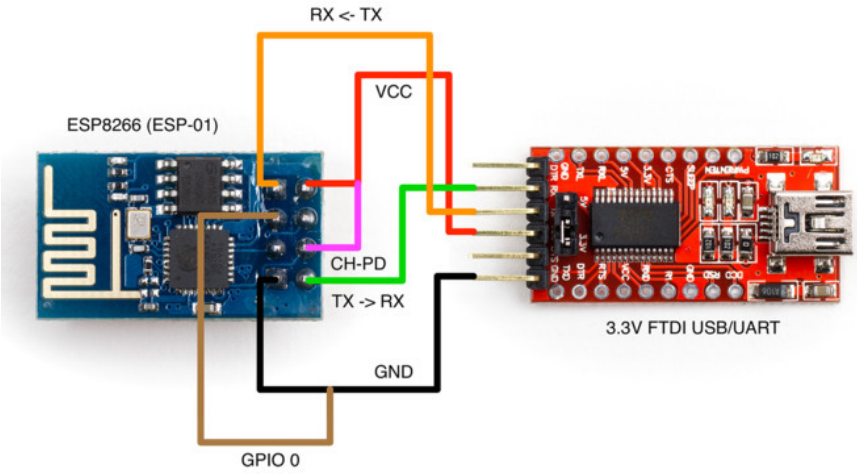
Подготовка модуля

Перед началом работы в модуль ESP8266 следует загрузить прошивку с поддержкой MAVLink. Для этого потребуется:

- сама прошивка;
- компьютер с ОС на базе GNU/Linux;
- USB-UART адаптер;
- утилита esptool.

> Убедитесь в том, что на вашем USB-UART адаптере установлено напряжение **3.3 В**!

Подключите ваш модуль к USB-UART, как показано на схеме:



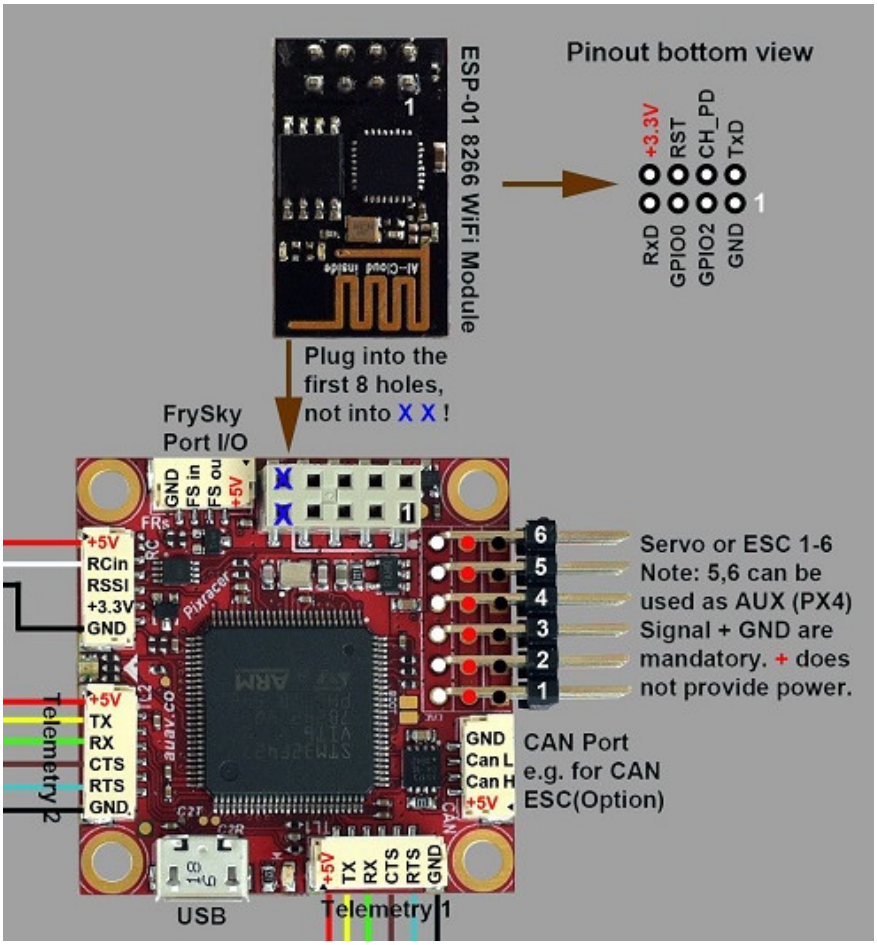
Скачайте прошивку для модуля. Подключите USB-UART к компьютеру и посмотрите, какое устройство соответствует переходнику. Убедитесь, что у вас установлена утилита esptool (её можно поставить командой `pip install esptool`). Запустите процесс загрузки прошивки командой:

```
esptool.py --baud 921600 --port /dev/ttyUSB0 write_flash 0x00000 firmware-1.2.2
```

Вместо `/dev/ttyUSB0` укажите устройство, соответствующее вашему переходнику, а вместо `firmware-1.2.2.bin` - путь к прошивке.

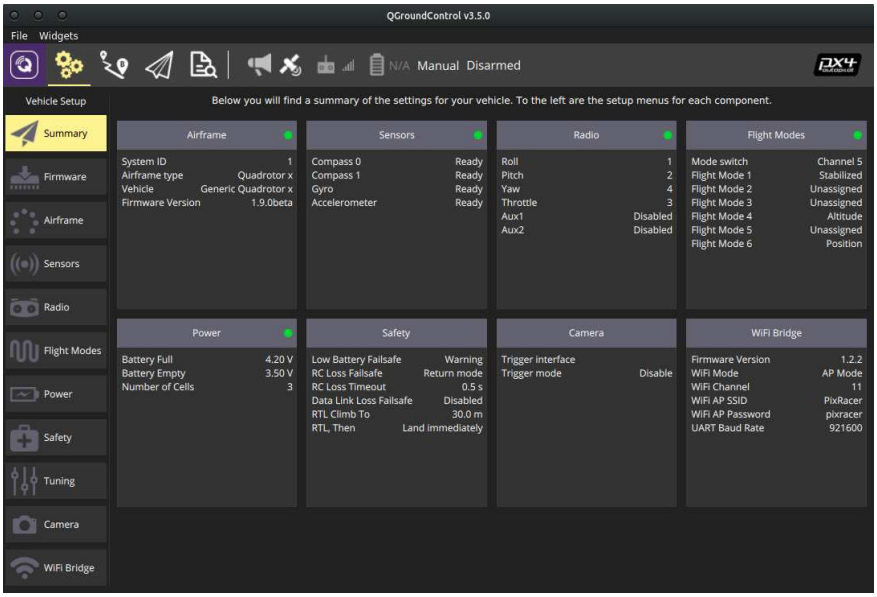
> Если в процессе загрузки прошивки возникли проблемы, вы можете запустить процесс заново.

Работа с ESP8266

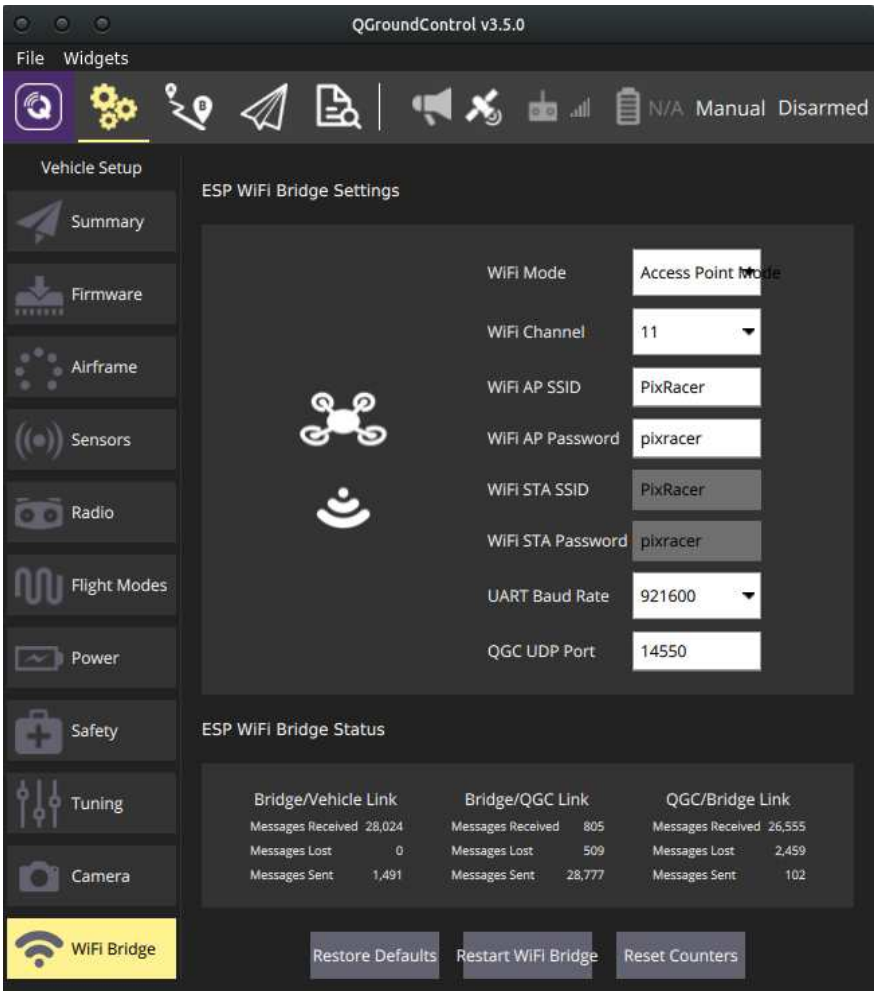


Подключите ESP8266 к Pixracer так, как показано на схеме, и включите полётный контроллер. В списке доступных Wi-Fi сетей появится сеть `PixRacer` с паролем по умолчанию `pixracer`. Подключитесь к этой сети и запустите QGroundControl. Программа должна автоматически установить соединение с полётным контроллером.

> Если автоматическое подключение не происходит, проверьте, что в настройках QGroundControl включено автоматическое подключение по UDP.

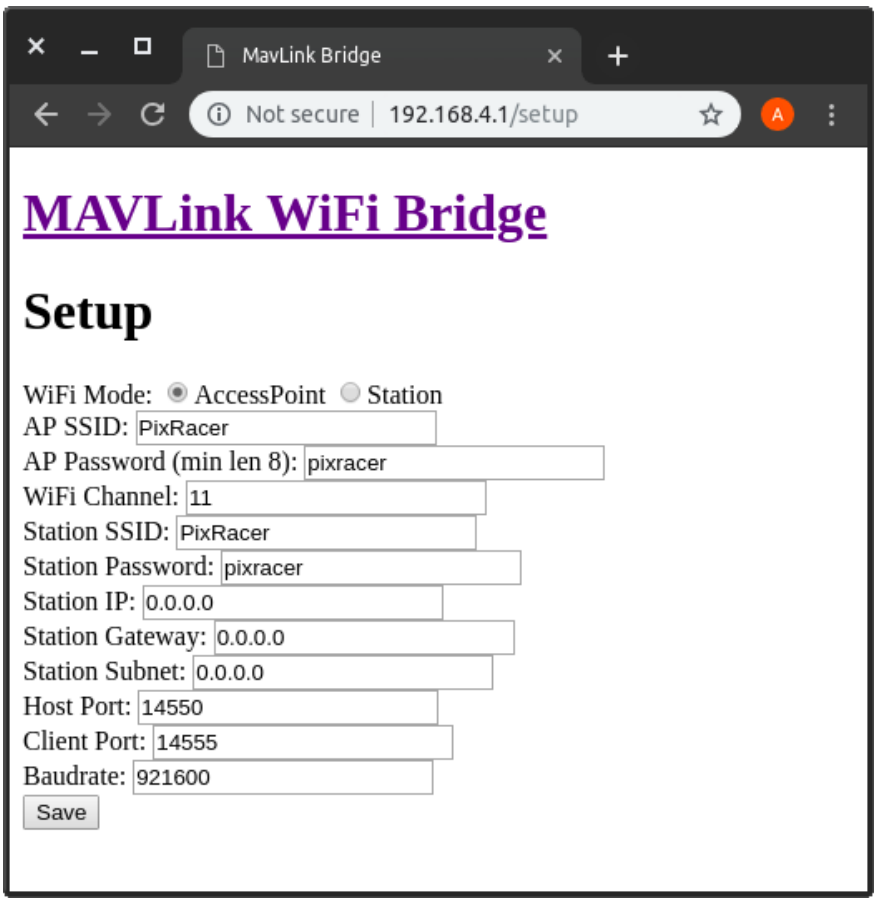


В меню настроек полётного контроллера появится вкладка **WiFi Bridge**. В ней можно изменить некоторые параметры модуля, например, режим работы (точка доступа/клиент), название и пароль Wi-Fi сети.



> Настоятельно рекомендуется поменять стандартные параметры сети!

Также эти параметры можно поменять в веб-интерфейсе модуля, доступном по умолчанию по адресу http://192.168.4.1/setup.



Автоматическая сборка и модификация образа Клевера

Иногда возникает необходимость в сборке модифицированного образа системы, например для своего проекта на базе Клевера. За основу можно взять, например, чистый образ Raspbian Stretch и модифицировать его с нуля, пройдя те же этапы, через который проходит сборка образа Клевера, добавив свои модификации. Однако на данный момент времени сборка образа Клевера занимает чуть больше часа, что превышает ограничения бесплатной сборки в Travis (50 минут). Соответственно для проектов на базе Клевера имеет смысл брать за основу уже готовый образ и кастомизировать его. Концепция и основные этапы для автоматизированной сборки изложены ниже.

Концепция

Имеется Docker образ, который содержит инструментарий для выполнения скриптов, копирования файлов и увеличения/сжатия размера образа системы на требуемой платформе для сборки (например сборка для Raspberry Pi 3 осуществляется через qemu-arm-static, пример Docker образа для сборки находится здесь). При запуске Docker образа выполняется скрипт `builder/image-build.sh`, в котором описан процесс сборки (например скачивание опорного образа - увеличение свободного места на образе - установка необходимого софта - сжатие образа), в результате которого создаётся файл образа системы. Триггер сборки, запуск Docker образа для сборки, выкладка образа осуществляется с помощью CI (continuous integration) системы Travis.

Добавление скриптов сборки

1. Для осуществления сборки образа добавьте в свой проект build скрипты, модифицирующие исходный образ. За основу можно взять скрипты из репозитория Клевера (папка builder) или из репозитория шоу дронов на основе Клеверов (тоже папка builder). Опорный скрипт, который исполняется безусловно Docker образом в этих проектах - `builder/image-build.sh`.
2. Для автоматического запуска сборки в облаке добавьте в свой проект `.travis.yml` файл, описывающий последовательность этапов выполнения сборки и правила для выкладки образов. Пример из репозитория Клевера, пример из репозитория шоу дронов. Документация по составлению `.travis.yml` файла находится здесь.

Настройка инструмента сборки travis-ci.com

1. Войдите в Travis через свой GitHub аккаунт.
2. Проверьте, что файл `.travis.yml` добавлен правильно: выберите свой проект, нажмите Trigger build из выпадающего меню справа сверху. Сборка должна начаться и успешно завершиться через некоторое время, если всё правильно.
3. Настройте проект. Основные настройки можно оставить по умолчанию. Если необходимы ключи авторизации (токены) для доступа к репозиторию (например для того, чтобы выложить образ прикреплённым файлом в релиз), нужно сгенерировать их в своём аккаунте и добавить под названием переменной, которая используется для передачи токена.

Запуск сборки в облаке travis-ci.com

По умолчанию скрипт сборки из `.travis.yml` файла выполняется автоматически при любом изменении GitHub репозитория. Есть возможность добавить скрипты, которые будут выполняться только при создании релиза (публикации тега), пример здесь.

Запуск сборки на локальной машине

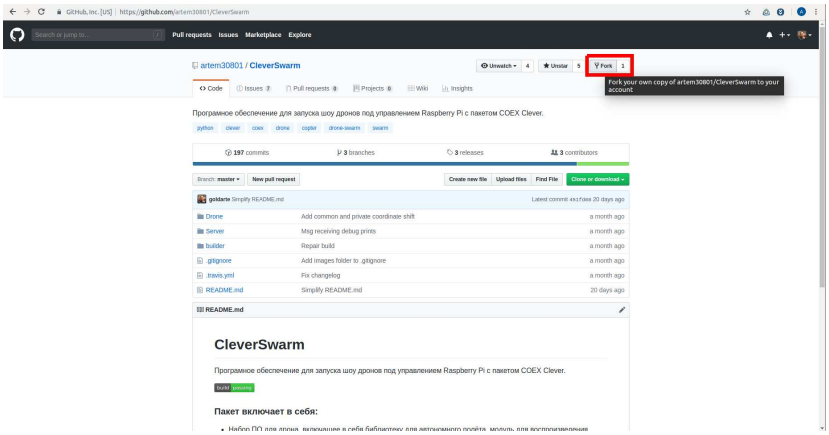
Если есть необходимость собрать образ быстрее, чем в облаке, или поэкспериментировать со сборкой локально, можно запустить Docker образ на локальной машине. Для этого необходимо в консоли перейти в папку с репозиторием, где прописаны скрипты автоматической сборки, и запустить оттуда Docker, например (подробнее здесь):

```
cd repo-w-instructions

docker run --privileged -it --rm -v /dev:/dev -v $(pwd):/mnt goldarte/img-tool
```

Пример запуска автоматической сборки образа из форка репозитория шоу дронов

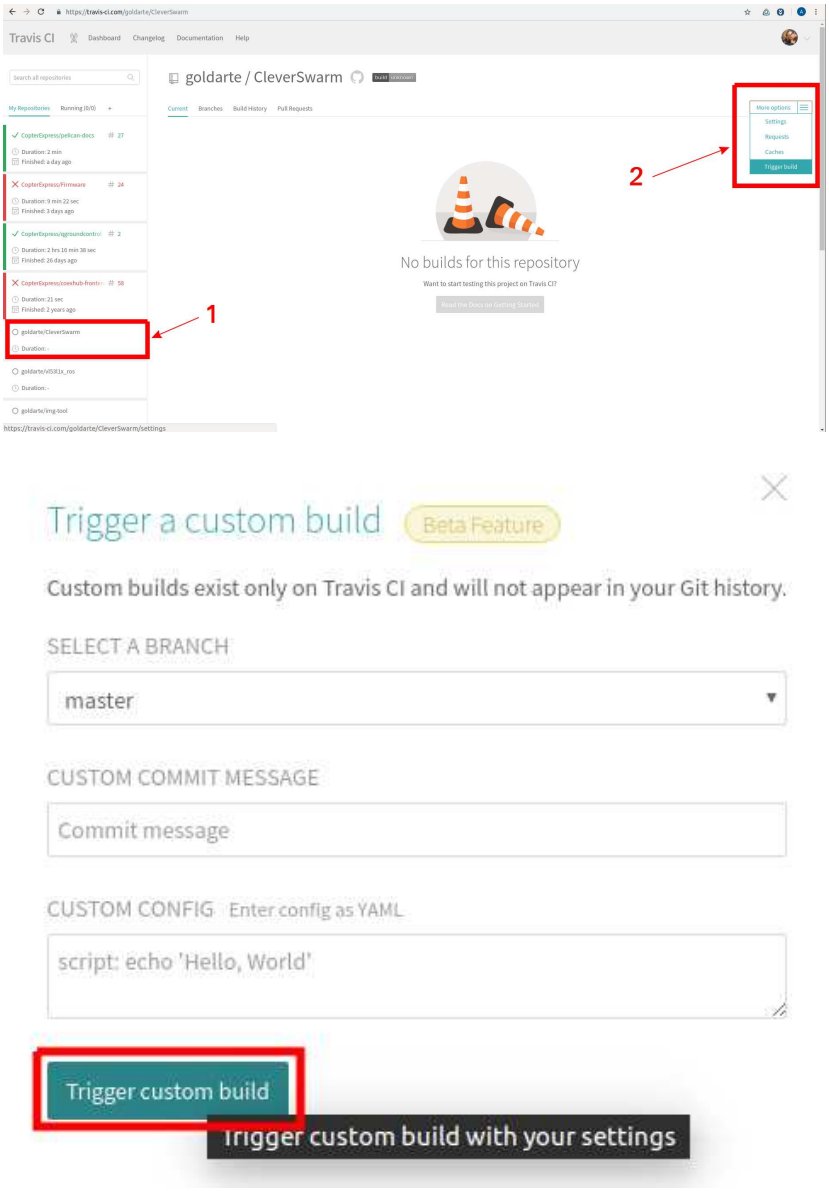
- Сделайте форк репозитория:



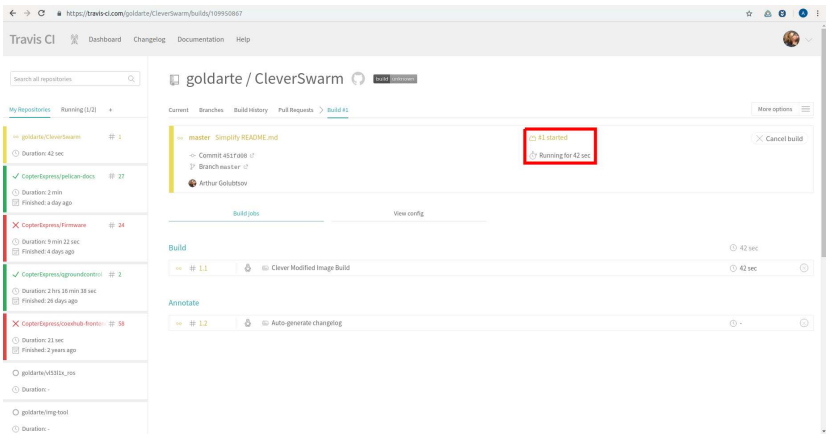
- Склонируйте репозиторий к себе на компьютер:

```
git clone <адрес репозитория>
```

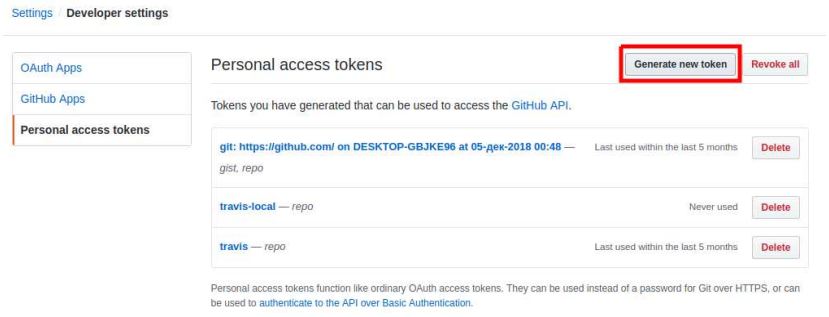
- Зайдите на travis-ci.org под своим аккаунтом GitHub.
- Выберите среди проектов форк репозитория шоу дронов и запустите тестовую сборку, нажав Trigger build из выпадающего меню:

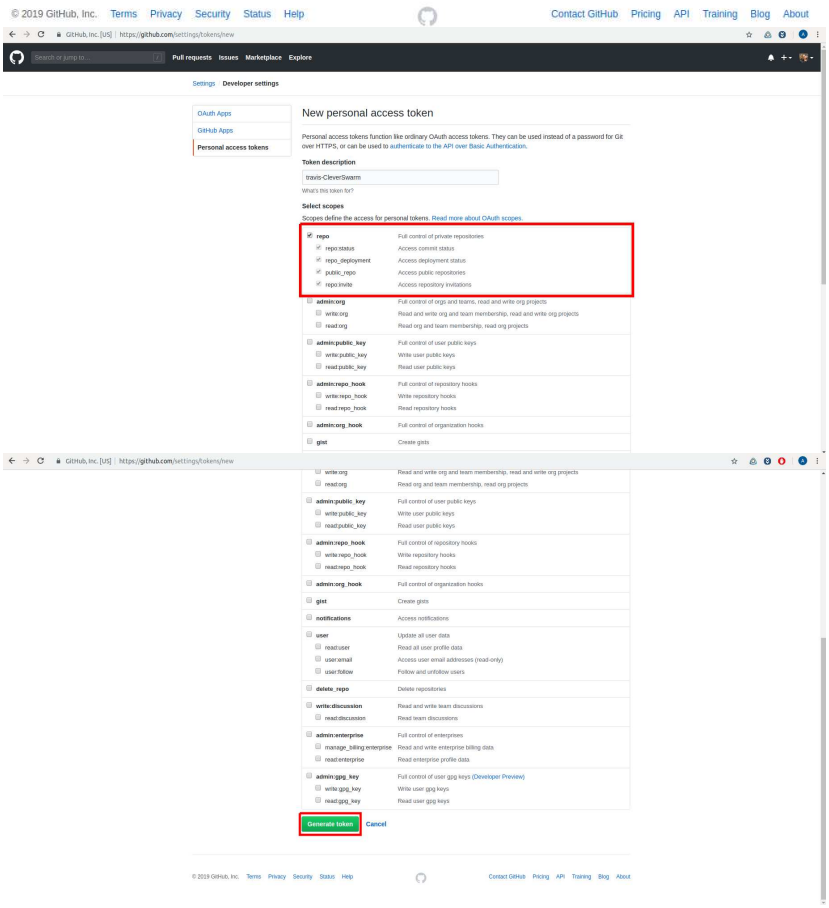


- Проверьте, что сборка запустилась:

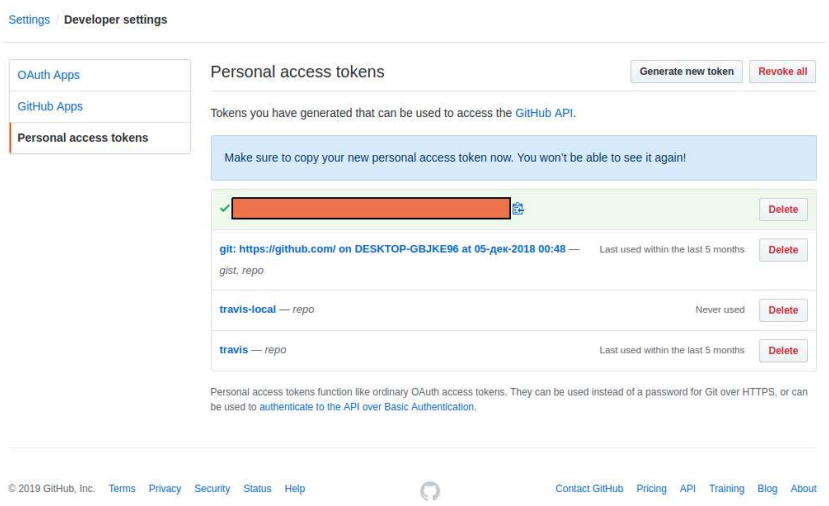


- Добавьте ключ аутентификации к вашему репозиторию для прикрепления файла образа к релизу. Зайдите в настройки токенов своего аккаунта и сгенерируйте новый токен для доступа к вашему репозиторию:

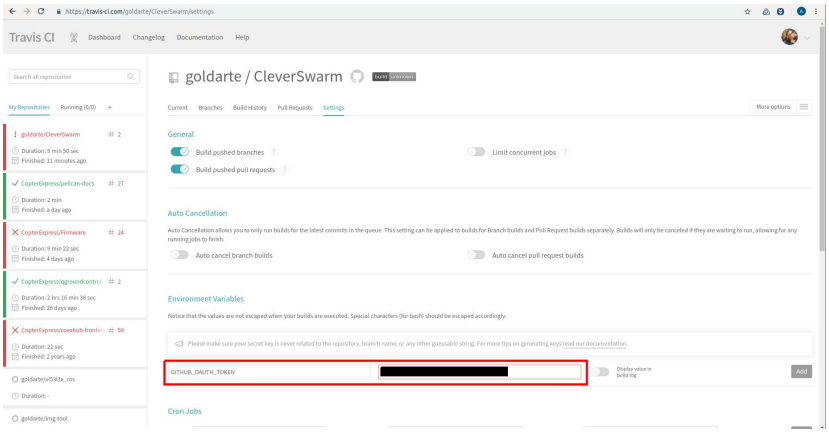




- Скопируйте получившийся токен:



- Перейдите в настройки сборки travis-ci.com и добавьте скопированный токен под именем `GITHUB_OAUTH_TOKEN`:

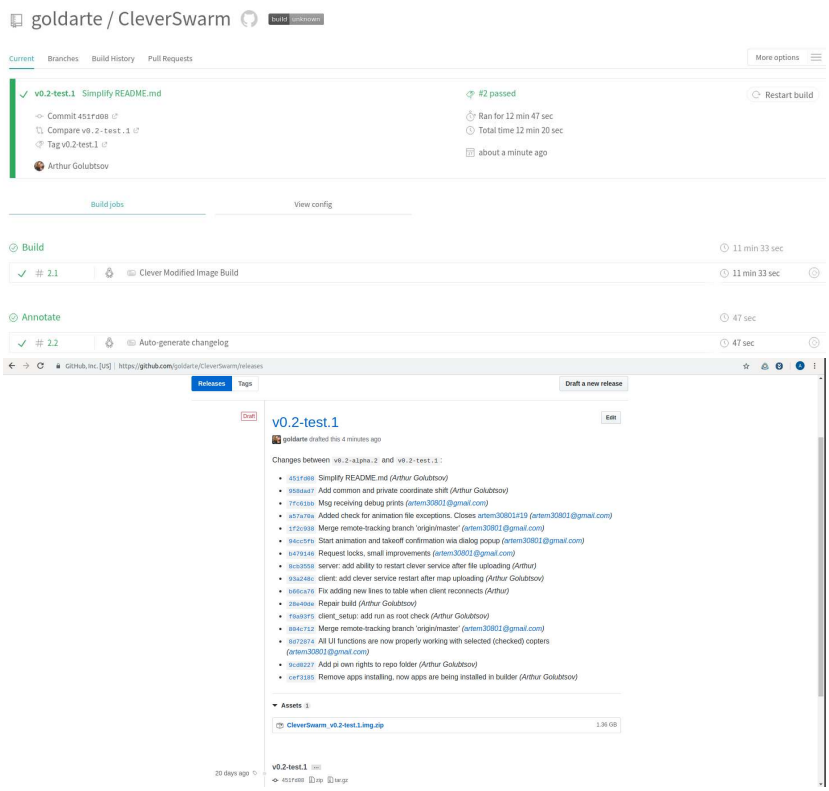


- В терминале перейдите в папку со скопированным репозиторием, создайте и опубликуйте тег для создания пре-релиза, автоматической сборки и выкладки образа на GitHub:

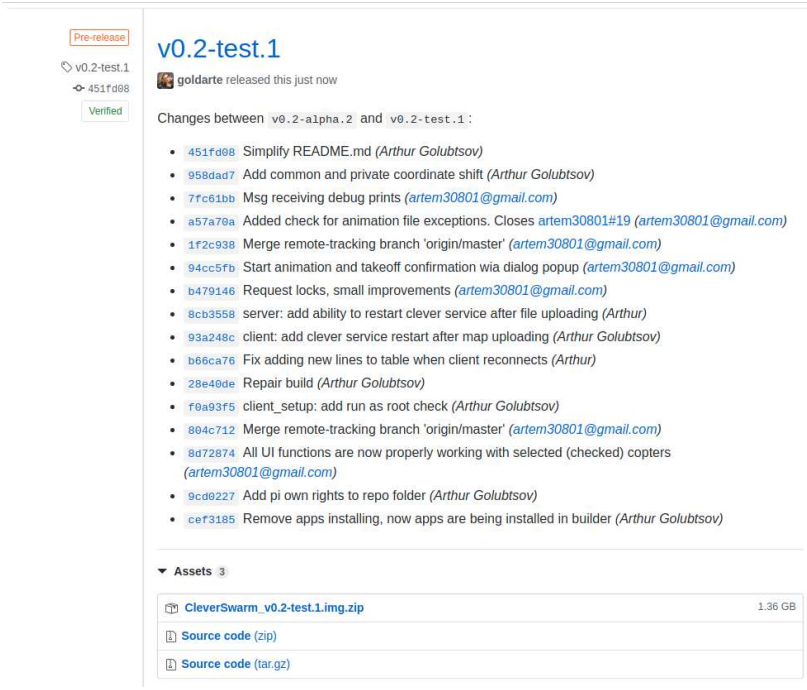
```
git tag <имя тега>

git push --tags
```

- Дождитесь окончания сборки образа и проверьте раздел Releases в вашем репозитории:



- Нажмите на кнопку Draft a new release и выпустите pre-release или release собранного образа и исходным кодом:



Pre-release

v0.2-test.1

451fd08

Verified

v0.2-test.1

goldarte released this just now

Changes between `v0.2-alpha.2` and `v0.2-test.1`:

- `451fd08` Simplify README.md *(Arthur Golubtsov)*
- `958dad7` Add common and private coordinate shift *(Arthur Golubtsov)*
- `7fc61bb` Msg receiving debug prints *(artem30801@gmail.com)*
- `a57a70a` Added check for animation file exceptions. Closes artem30801#19 *(artem30801@gmail.com)*
- `1f2c938` Merge remote-tracking branch 'origin/master' *(artem30801@gmail.com)*
- `94cc5fb` Start animation and takeoff confirmation wia dialog popup *(artem30801@gmail.com)*
- `b479146` Request locks, small improvements *(artem30801@gmail.com)*
- `8cb3558` server: add ability to restart clever service after file uploading *(Arthur)*
- `93a248c` client: add clever service restart after map uploading *(Arthur Golubtsov)*
- `b66ca76` Fix adding new lines to table when client reconnects *(Arthur)*
- `28e40de` Repair build *(Arthur Golubtsov)*
- `f0a93f5` client_setup: add run as root check *(Arthur Golubtsov)*
- `804c712` Merge remote-tracking branch 'origin/master' *(artem30801@gmail.com)*
- `8d72874` All UI functions are now properly working with selected (checked) copters *(artem30801@gmail.com)*
- `9cd0227` Add pi own rights to repo folder *(Arthur Golubtsov)*
- `cef3185` Remove apps installing, now apps are being installed in builder *(Arthur Golubtsov)*

Assets 3

CleverSwarm_v0.2-test.1.img.zip	1.36 GB
Source code (zip)	
Source code (tar.gz)	

Подключение регуляторов 4in1

Распиновка платы регуляторов 4in1

Одним цветом выделены соответствующие фазные провода (рис. 1a) и управляющий ими сигнал (рис. 1b).

Например, оранжевый цвет -> нижний правый мотор -> S1 - оранжевый провода.

Распиновка полетного контроллера Pixracer

На рис. 2a указана распиновка гребенки:

- **SIGNAL** – подключение регуляторов. Каждый пин имеет свой собственный сигнал. На 5 и 6 сигнал можно получать ШИМ сигнал (Например, можно подключить сервопривод).
- **GND** – земля полетного контроллера. Единая шина на всех пинах GND (отмечены черным).
- 1, 2, 3, 4 – порты для подключения ESC.
- 1, 2 - порты расширения выходного ШИМ сигнала (настраиваются в QGroundControl, также могут использоваться для управления гексакоптером).

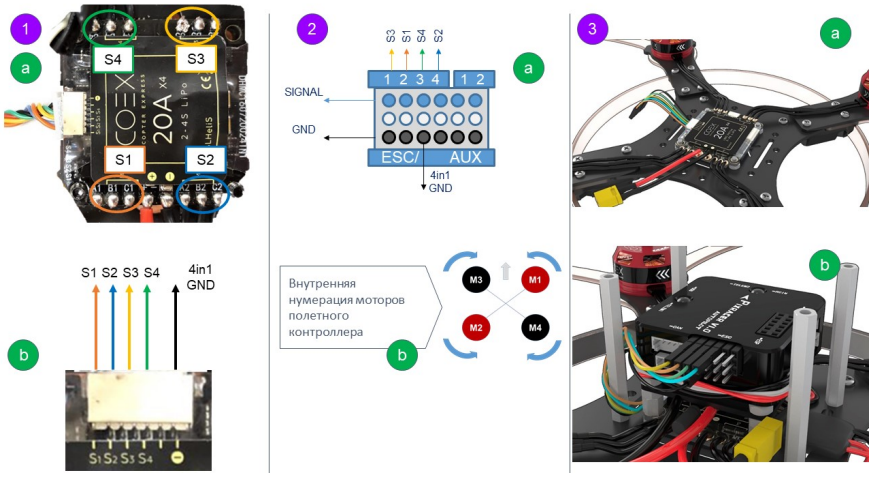
  На рис. 2b указана нумерация моторов полетного контроллера Pixracer.

- Стрелочка – ориентация полетного контроллера.
- Черные M3, M4 – моторы, вращающиеся по часовой стрелке.
- Красные M1, M2 – моторы, вращающиеся против часовой стрелке.

Иллюстрация подключения, исходя из текущей ориентации платы регуляторов 4in1

Используя рис. 1a, 1b, 2a, 2b необходимо сопоставить каждому мотору свой сигнал управления и подключить в соответствии с порядком нумерации моторов Pixracer.

Например, мотор M3, вращающийся против часовой стрелки (верхний левый угол) управляется сигналом S4 (зеленый провод). Подключается в порт 3.

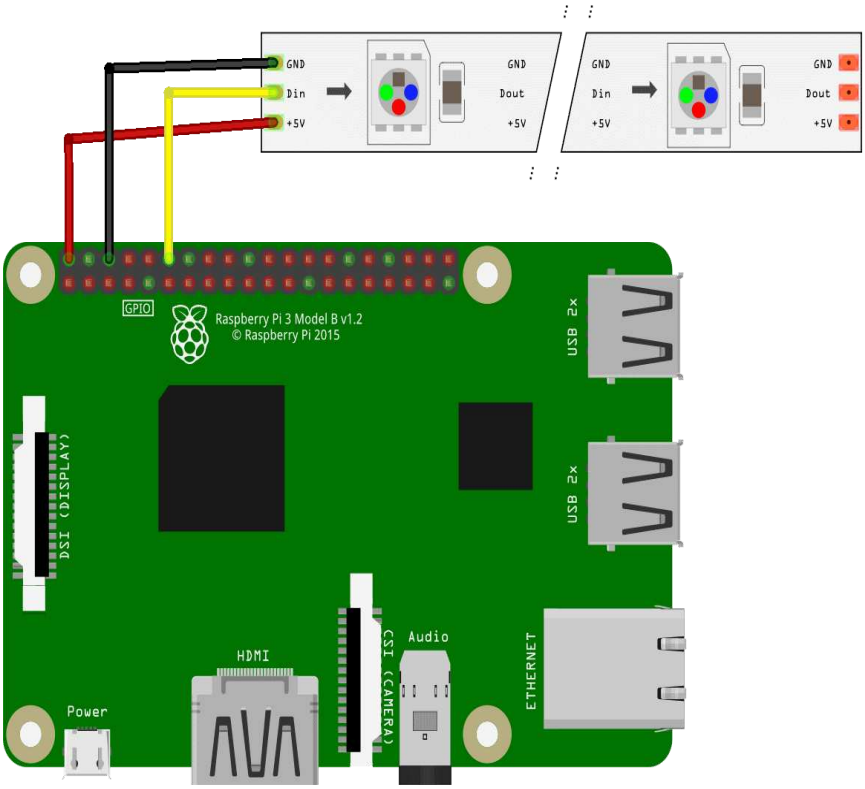

1
a
S4
S3
S1
S2
20A
2-4S LiPo
LiHe/iS
b
S1 S2 S3 S4
4in1
GND
2
S3 S1 S4 S2
1 2 3 4
1 2
SIGNAL
GND
ESC/
AUX
4in1
GND
a
Внутренняя нумерация моторов полетного контроллера
M3
M1
M2
M4
b
3
a
b

Работа со светодиодной лентой на Raspberry 3

Подключение и определение типа ленты

> Документация для версии образа, начиная с 0.14. Для более ранних версий см. документацию для версий 0.13

Подключите светодиодную ленту к питанию +5v - 5v, земле GND - GND и сигнальному порту DIN - GPIO30, **GPIO21**, GPIO18. По умолчанию нужно подключать к **GPIO21**.



> Обратите внимание, что светодиодную ленту нужно питать от стабильного источника энергии. Если вы подключите питание напрямую к Raspberry, то это создаст слишком большую нагрузку на ваш микрокомпьютер. Для снятия нагрузки с Raspberry можно подключить питание к преобразователю `BEC` .

> При работе с GPIO следует подключать ленту к пину GPIO21. В противном случае управление LED-лентой будет работать некорректно.

Совместимость с ROS и Python

Чтобы корректно работать со светодиодной лентой вам нужно добавить в окружение необходимые пути к библиотекам Python и пакетам ROS, для этого необходимо добавить в файл `/etc/sudoers` следующие строки:

```
Defaults        env_keep += "PYTHONPATH"
Defaults        env_keep += "PATH"
Defaults        env_keep += "ROS_ROOT"
Defaults        env_keep += "ROS_MASTER_URI"
Defaults        env_keep += "ROS_PACKAGE_PATH"
Defaults        env_keep += "ROS_LOCATIONS"
Defaults        env_keep += "ROS_HOME"
Defaults        env_keep += "ROS_LOG_DIR"
```

Пример программы для светодиодной ленты на RPI3

Для проверки работоспособности ленты можете использовать приведенный ниже код, данный код поочередно зажжет первые 10 диодов 3 цветами и в конце их погасит.

```
import time

from rpi_ws281x import Adafruit_NeoPixel
from rpi_ws281x import Color


LED_COUNT      = 10      # Количество светодиодов в ленте

LED_PIN        = 21      # GPIO пин, к которому вы подсоединяете светодиодную л
LED_FREQ_HZ    = 800000  # Частота несущего сигнала (обычно 800 кГц)
LED_DMA        = 10      # DMA-канал для генерации сигнала (обычно 10)
LED_BRIGHTNESS = 255     # Яркость: 0 - наименьшая, 255 - наибольшая
LED_INVERT     = False   # True для инвертирования сигнала (для подключения чер
LED_CHANNEL    = 0       # '1' для GPIO 13, 19, 41, 45 или 53

strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT)

strip.begin()


def colorWipe(strip, color, wait_ms=50):
    """Заполнение ленты цветом по одному светодиоду."""
    for i in range(strip.numPixels()):
        strip.setPixelColor(i, color)
        strip.show()
        time.sleep(wait_ms/1000.0)


print('Color wipe animations.')
colorWipe(strip, Color(255, 0, 0), wait_ms=100)  # Заполнение красным
colorWipe(strip, Color(0, 255, 0), wait_ms=100)  # Заполнение зелёным
colorWipe(strip, Color(0, 0, 255), wait_ms=100)  # Заполнение синим
colorWipe(strip, Color(0, 0, 0), wait_ms=100)    # Выключение ленты
```

> Вы так же можете использовать тестовый код разработчиков данного модуля. Вы можете его скачать из репозитория разработчика.

Сохраните программу в ваш скрипт и запустите его используя права администратора:

```
sudo python YourScriptName.py
```

Основные функции используемые для работы со светодиодной лентой

Для подключения библиотеки и её корректной работы требуется подключить следующие модули: `Adafruit_NeoPixel` и `Color` - для работы ленты и `time` – для управления задержками.

```
from rpi_ws281x import Adafruit_NeoPixel
from rpi_ws281x import Color
import time
```

Для работы с лентой необходимо создать объект типа `Adafruit_NeoPixel` и инициализировать библиотеку:

```
# Создание объекта NeoPixel с заданной конфигурацией
strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT
# Инициализация библиотеки, должна быть выполнена перед другими функциями
strip.begin()
```

Основные функции, которые используются для управления лентой:

- `numPixels()` - возвращает количество пикселей в ленте. Удобно для цикличного управления всей лентой целиком.
- `setPixelColor(pos, color)` – устанавливает цвет пикселя в позиции `pos` в цвет `color`. Цвет должен быть 24 битным значением, где первые 8 бит - красный цвет (red), следующие 8 бит - зелёный цвет (green) и последние 8 бит - голубой (blue). Для получения значения `color` можно использовать функцию `Color(red, green, blue)`, которая составляет это значение из 3х компонент. Каждый компонент должен находиться в диапазоне 0-255, где 0 – отсутствие цвета, а 255 – наибольшая доступная яркость компонента в светодиодном модуле.
- `setPixelColorRGB(pos, red, green, blue)` – устанавливает цвет пикселя в позиции pos в цвет, состоящий из компонент `red`, `green`, `blue`. Каждый компонент должен находиться в диапазоне 0–255, где 0 – отсутствие цвета, а 255 – наибольшая доступная яркость компонента в светодиодном модуле.
- `show()` – обновляет состояние ленты. Только после её использования все программные изменения перемещаются на светодиодную ленту.

Почему именно так и можно ли по-другому?

Основной тип ленты, который используется для Клевера 3 управляются по принципу: для массива светодиодов в ленте отправляется пакет данных по 24 бита на светодиод; каждый светодиод считывает первые 24 бита из пришедших к нему данных и устанавливает соответствующий цвет, остальные данные он отправляет следующему светодиоду в ленте. Нули и единицы задаются разными сочетаниями длительностей высокого и низкого уровня в импульсе.

Используемый тип ленты поддерживаются для управления библиотекой rpi_ws281x, при этом для управления используется модуль DMA (direct memory access) процессора Raspberry Pi и один из каналов передачи данных: PWM, PCM или SPI, что гарантирует отсутствие задержек в управлении (а управляется всё на многозадачной операционке, это важно).

Есть некоторые особенности работы с каналами, например при передаче данных с помощью PWM (ШИМ) перестаёт работать встроенная аудиосистема Raspberry Pi, при передаче данных по PCM блокируется использование подключенных цифровых аудиоустройств (при этом встроенная система работает), а при использовании SPI (кстати, требуется специальная настройка размера буфера и частоты GPU Raspberry Pi для правильной работы) лента блокирует все остальные устройства, подключенные по этому каналу.

Есть некоторые особенности выбора канала DMA для управления лентой: некоторые каналы используются системой, поэтому их использование может привести к неприятным последствиям, например использование 5 канала рушит файловую систему Raspberry, т.к. этот канал используется при чтении-записи на SD карту. Безопасный канал – 10, он же установлен по умолчанию в приведённой выше библиотеке.

Поэтому сценарии использования LED-ленты следующие:

1. Если нам не важна работоспособность встроенного аудио на Raspberry Pi (и мы его не используем, т. к. аудио и лента будут выдавать белиберду в этом случае), то можно использовать PWM канал (для этого требуется подключить вход ленты к одному из следующих GPIO портов Raspberry Pi: 12, 18, 40, или 52 для PWM0 канала и 13, 19, 41, 45 или 53 для PWM1 канала).
2. Если нам не важно наличие на шине SPI других устройств, то можно управлять лентой по каналу SPI (GPIO на Raspberry Pi 10 или 38). Здесь требуется произвести следующие настройки (только для Raspberry Pi 3):
   - увеличить размер буфера передачи данных для поддержки длинных лент, добавив стройку `spidev.bufsiz=32768` в файл `/boot/cmdline.txt`;
   - установить частоту GPU для правильной частоты работы SPI, добавив строку `core_freq=250` в файл `/boot/config.txt`.
   - перезагрузить вашу Raspberry, используя команду `sudo reboot`

3. Если нам важна и работа аудио, и подключение к SPI устройств кроме лед ленты, то можно управлять лентой по каналу PCM (GPIO 21 или 31). При этом никаких дополнительных манипуляций с Raspberry не требуется.

Исходя из вышеперечисленных способов управления лентой, наилучшим вариантом, позволяющим управлять лентой, сохранить работоспособность встроенной аудиосистемы и возможность подключения всяческих устройств и датчиков по SPI, является управление по каналу PCM (GPIO 21) с использованием 10 канала DMA.

Вклад в Клевер

Клевер – это, по большей части, open source и open hardware проект, который ставит своей целью уменьшение порога входа в разработку проектов, связанных с летающей робототехникой. Вы можете внести свой вклад, предлагая исправления и улучшения в документацию и ПО Клевера.

> Для внесения предложений по изменению документации или ПО Клевера необходимо иметь аккаунт на GitHub.

Markdown

Вся документация Клевера написана в широко распространенном формате Markdown. В Интернете существует множество руководств по нему.

На русском: https://guides.hexlet.io/markdown/.

На английском: https://www.markdownguide.org/getting-started, https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet.

Для удобного редактирования текста, вы можете использовать текстовые редакторы с поддержкой Markdown: Typora, Dillinger (веб), VSCode с плагином Markdown Editor.

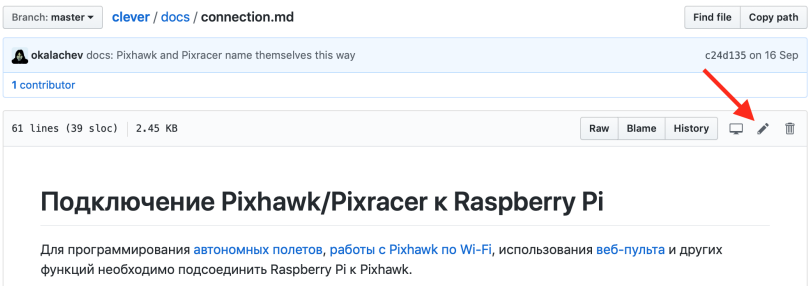
Для VSCode также рекомендуется использование плагина Code Spell Checker (словарь для русского языка).

Для локальной сборки статического сайта документации необходимо использовать утилиту `gitbook-cli`.

Исправление ошибок в документации

Если вы нашли ошибку в документации или хотите ее улучшить, используйте механизм **Pull Request**'ов.

1. Найдите файл с интересующей вас статьей в репозитории – https://github.com/CopterExpress/clover/tree/master/docs.
2. Нажмите кнопку "Редактировать".



3. Внесите необходимые изменения.

4. Нажмите кнопку "Propose file change".
5. Опишите ваше изменение и нажмите кнопку "Create Pull Request".
6. Ожидайте принятия ваших изменений :)

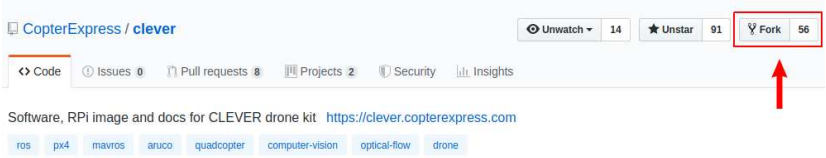
Более подробную информацию о Pull Request'ах смотрите на GitHub (англ.) или в документации по git (русск.).

Добавление статьи в GitBook

> Если вы реализовали собственный интересный проект на Клевере, вы можете добавить статью о нем в раздел "Проекты на базе Клевера".

Подготовьте вашу статью и пришлите Pull Request с ней в репозиторий Клевера.

1. Сделайте форк репозитория Клевера:



2. Склонируйте форк на компьютер:

```
git clone https://github.com/<USERNAME>/clover.git
```

3. Перейдите в директорию с форком и создайте новую ветку с названием вашей статьи (например `new-article`):

```
git checkout -b new-article
```

4. Напишите новую статью в разделе `docs/ru` или `docs/en` в формате Markdown (например `docs/ru/new_article.md`). Не забудьте указать контактную информацию (e-mail / Telegram /...) для авторских статей.
5. Поместите дополнительные визуальные материалы в папку `docs/assets` и оформите на них ссылки в вашей статье.
6. Добавьте статью в файл оглавления `SUMMARY.md` в том разделе, где вы её написали (например в `docs/ru/SUMMARY.md`):

```
...
* Дополнительные материалы
  * [Олимпиада НТИ 2019](nti2019.md)
  * [Вклад в Клевер](contributing.md)
  * [Новая статья](new_article.md)
  * [Сборка и модификация образа Клевера](image_building.md)
  * [Прошивка ESC контроллеров](esc_firmware.md)
...
```

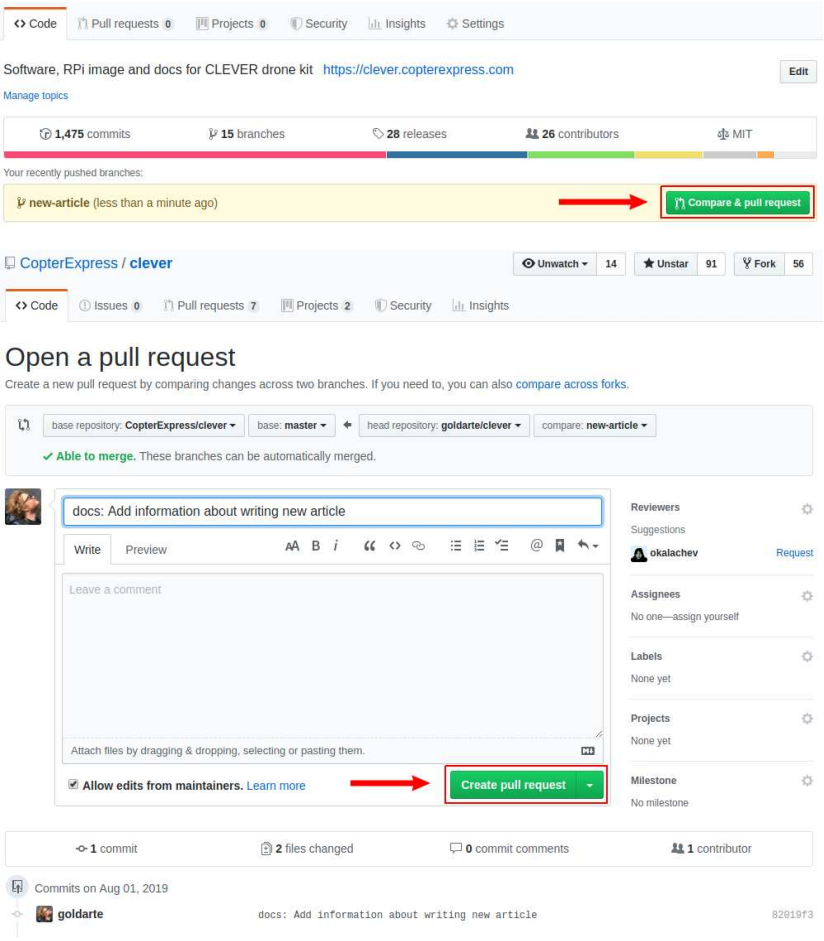
7. Сохраните состояние ваших изменений локально:

```
git add docs/
git commit -m "Add new article for Clover"
```

8. Загрузите вашу новую ветку с изменениями на ваш GitHub репозиторий с форком Клевера:

```
git push -u origin new-article
```

9. Перейдите на web страницу вашего форка и сделайте `pull request` вашей ветки в master Клевера:



10. Дождитесь комментариев на свою статью, сделайте правки, если потребуется.

11. Порадуйтесь своей новой полезной статье, опубликованной на https://clovercoex.tech !

Простой способ

Если вышеприведенные инструкции для вас оказываются слишком сложными, отправляйте правки или новые статьи по e-mail (okalachev@gmail.com) или в Telegram (пользователь @okalachev).

Публикация пакетов

Вы также можете опубликовать собственный пакет, расширяющий функциональность Клевера, в Debian-репозитории COEX.

Репозиторий пакетов COEX

COEX предоставляет открытый Debian-репозиторий с предсобранными пакетами, относящимися к ROS Noetic, для архитектуры `armhf`.

> Адрес репозитория: http://packages.coex.tech.

Репозиторий подключен в образе для RPi и может быть использован для легкой установки дополнительных ROS-пакетов.

Публикация пакетов

Вы можете прислать Pull Request в git-репозиторий с пакетами, добавляющий или обновляющий ваш пакет (файл с расширением `.deb`), относящийся с Клеверу или ROS. После принятия ваш пакет будет доступен для установки с помощью утилиты `apt`:

```
sudo apt install ros-noetic-clover-some-feature
```

Пакеты, расширяющие функциональность Клевера, рекомендуется называть с префиксом `clover_`, например `clover_some_feature`.

Использование на обычной Raspberry Pi OS

На обычной Raspberry Pi OS репозиторий может быть добавлен в список источников пакетов следующими командами:

```
wget -O - 'http://packages.coex.tech/key.asc' | apt-key add -
echo 'deb http://packages.coex.tech buster main' >> /etc/apt/sources.list
sudo apt update
```

Список тестирования

Актуальный список для ручного тестирования релизов Клевера.

Критичность: **критично**, средняя критичность, *не критично*.

Образ Клевера

Общие тесты

- **Раздача Wi-Fi**
- **Возможность подключения по SSH по IP и Hostname**
- **Успешное подключение COEX Pix по USB (по умолчанию)**
- Успешное подключение COEX Pix по UART (с настройкой)
- **Бридж для QGC – корректное подключение по TCP**
- Бридж для QGC – корректное подключение по UDP-b
- **Раздача главной страницы**
- **Раздача пользовательской документации (RU/EN), отсутствие битых изображений и т. д.**
- **Работа веб-терминала Butterfly**
- **Работа web_video_server**
- **Корректная работа драйвера камеры, корректные изображения и данные в топиках**
- **Корректная работа драйвера vl53l1x (i2c к Raspberry), в том числе топика `~data`**
- **Корректная работа optical flow и всех его топиков, полет по optical flow**
- **Полет по полю маркеров**
- **Корректная установка OpenCV – возможность использования из Python и C++**
- Работа примера с компьютерном зрением: `red_circle.py`
- **Отсутствие неожиданного жора памяти и CPU (можно контролировать с помощью `selfcheck.py` или `htop`)**
- Автоматическая перекалибровка камеры при изменении разрешения

Тесты веб-части

- Работа веб-просмотрщика топиков
- Работа веб-консоли
- Работа веб 3D-визуализации ArUco (map, detect)
- Работа веб 3D-визуализации web_rviz
- Работа веб 3D-визуализации web_visualization_aruco_map
- Работает отображение карты ArUco `/aruco_map/image` и в snapshot, и в debug
- Визуализация расположения камеры в web rviz
- Правильное отображение осей в `/aruco_map/image`

Тесты selfcheck.py

- **Корректная работа `rosrun clover selfcheck.py`, отсутствие варнингов, анализ вывода**
- **Выводит ориентацию камеры текстом**
- **Делает `commander check`**
- **Показывается, что используется наш форк прошивки и версию образа**
- **Показывает возникающие ошибки и опечатки, допущенные в .launch файлах**
- **Проверка на throttling**

Автоматизированные тесты

- **Корректная работы автоматизированных тестов**:
  - Тест автономного полета: `rosrun clover autotest_flight.py`
  - Тест автономного полета по маркерам: `rosrun clover autotest_aruco.py`
  - Тест LED-ленты: `rosrun clover autotest_led.py`

Тесты simple_offboard

- **Корректная работа simple_offboard – взлет, полет в точку в любом фрейме, отсутствие проблем с `yaw` и `yaw_rate`**
- **В фрейме `body`**
- **В фрейме `aruco_map`**
- **В фрейме `map`**
- **В фрейме `navigate_target`**
- **В фрейме `terrain`**
- Корректное выполнения флипа
- **Возможность лететь к отдельным маркерам в карте, которые вне кадра и в кадре**
- **Корректное детектирование статуса kill switch при выполнение команды с флагом `auto_arm`**
- *Корректная работа outdoor по GPS-координатам*
- Работают программы из папки `~/examples`

Тесты ArUco

- **Распознавание ArUco-маркеров, корректная работа всех топиков пакета `aruco_detect` и `aruco_map`**
- **VPE-полеты по маркерам на полу**
- *VPE-полеты по маркерам на потолке*
- Корректное распознавание ArUco-маркеров и ArUco-карты (проверка с помощью rviz или debug)
- *Работает в случае если используется слишком большой ID*
- Работают комментарии в файле карты, а также в карте используется от 4 до 8 параметров
- Полет по Optical Flow над 1 маркером

- `aruco_map` не падает в случае маленьких размеров карты и маркеров

Тесты LED-ленты

- **Работает нода LED ленты на RPi 4**
- Дополнительная проверка на RPi 3, RPi 4 Rev. 1.4
- **Корректная работа всех notify эффектов заданных в `led.launch`**
- **Низкоуровневое управление отдельными диодами**
- **Высокоуровневое управление эффектами**

Тесты pigpiod

- Корректная работа pigpiod, возможность работы с сонаром, сервой и электромагнитом по мануалу
- Одновременная работа pigpiod и rpi_ws281x (правильная работа светодиодной ленты и сервы)

Блочное программирование

- Корректная работа функционала блочного программирования
- Работа функций сохранение/загрузка/удаление
- Работа с pigpiod
- Работа всех примеров

Дополнительно

- ROS ноды не падают в случае потери всех соединений (удобно проверять с экраном)
- Работает `rosshow`
- Работает `espeak`
- Работает аргумент `rectify` в `main_camera.launch`
- *Работает LIRC*
- *Работа iOS-пульта из коробки*
- *Работа Android-пульта из коробки*

Переход на версию 0.20

Образ для RPi версии 0.20 содержит в себе значительные изменения по сравнению с версией 0.19. При переходе на новую версию обратите внимание на разъяснения, приведенные ниже.

Пакет `clever` переименован в `clover`

Необходимо заменить все импорты модуля в Python-скриптах.

Было:

```
# coding: utf8

import rospy
from clever import srv
from std_srvs.srv import Trigger

rospy.init_node('flight')

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)

# Взлет на высоту 1 м
navigate(x=0, y=0, z=1, frame_id='body', auto_arm=True)
```

Стало:

```
import rospy
from clover import srv
from std_srvs.srv import Trigger

rospy.init_node('flight')

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)

# Взлет на высоту 1 м
navigate(x=0, y=0, z=1, frame_id='body', auto_arm=True)
```

systemd-сервис `clever` переименован в `clover`

Для перезапуска платформы теперь вместо команды:

```
sudo systemctl restart clever
```

используется команда:

```
sudo systemctl restart clover
```

Путь к файлам платформы изменен

Каталог `~/catkin_ws/src/clever/` переименован в `~/catkin_ws/src/clover`. Таким образом, файлы конфигурации (`.launch`) необходимо редактировать по новому пути.

Например, файл `~/catkin_ws/src/clever/clever/launch/clever.launch` теперь называется `~/catkin_ws/src/clover/clover/launch/clover.launch`.

Настройки Wi-Fi сети

SSID Wi-Fi сети изменен на `clover-XXXX` (где X – случайная цифра), пароль изменен на `cloverwifi`.

Способ настройки ориентации камеры изменен

Подробнее читайте в статье про настройку камеры.

Переход на версию 0.22

Переход на Python 3

Python 2 был признан устаревшим, начиная с 1 января 2020 года. Платформа Клевера переходит на использование Python 3.

Для запуска полетных скриптов вместо команды `python`:

```
python flight.py
```

теперь следует использовать команду `python3`:

```
python3 flight.py
```

Синтаксис языка Python 3 имеет определенные изменения по сравнения со второй версией. Вместо *оператора* `print`:

```
print 'Clover is the best'  # this won't work
```

теперь используется *функция* `print`:

```
print('Clover is the best')
```

Оператор деления по умолчанию выполняет деление с плавающей точкой (вместо целочисленного). Python 2:

```
>>> 10 / 4
2
```

Python 3:

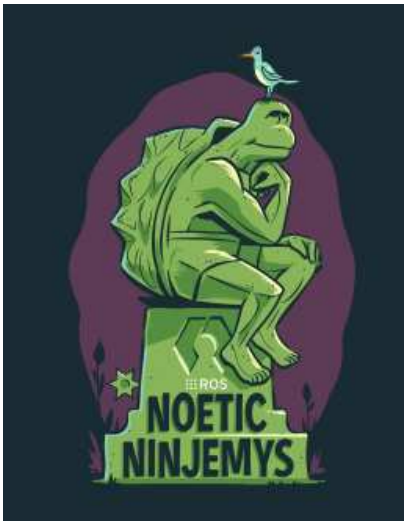
```
>>> 10 / 4
2.5
```

Для строк по умолчанию теперь используется тип `unicode` (вместо типа `str`).

Указание кодировки файла (`# coding: utf8`) перестало быть необходимым.

Полное описание всех изменений языка смотрите в соответствующей статье.

Переход на ROS Noetic



Версия ROS Melodic обновлена до ROS Noetic. Смотрите полный список изменений в официальной документации ROS.

Изменения в launch-файлах

Упрощено конфигурирование навигации с использованием ArUco-маркеров. Подробнее в статьях по навигации по маркерам и навигации по картам маркеров.

COEX DuoCam

"COEX DuoCam" – это программно-аппаратный комплекс, позволяющий получать комбинированное визуально-тепловизионное изображение.



В качестве источника визуальной картинки теоретически можно использовать любую камеру, имеющую CSI или HDMI-выход.

В качестве источника тепловизионной картинки можно использовать тепловизоры Seek Compact или Seek Compact Pro.

В качестве платформы для запуска обрабатывающего софта рекомендуется использовать одноплатный компьютер Raspberry Pi 4B.

Подключение устройств

Тепловизор подключается в один из портов USB 3.0 на RPi4. Во второй USB 3.0-порт подключается Wi-Fi адаптер для осуществления связи с наземной станцией.

Визуальная камера подключается в CSI порт напрямую или через HDMI-CSI-конвертер.

В порт USB 2.0 следует подключить USB-flash устройство, на которое будет производиться сохранение фото и видеофайлов.

Принцип работы

Программная часть осуществляет наложение тепловизионной картинки на визуальную. С помощью конфигурационного файла или виджета управления свойствами камеры в QGroundControl можно изменять результирующее изображение.

Выбор модели тепловизора

Первоначально надо определиться, какая у вас модель тепловизора (простая или Pro) и выставить соответствующую настройку в конфиг-файле (`enable: yes` у seek_thermal_pro или seek_thermal в разделе thermal_camera_models. Важно не забыть выставить `enable: no` у камеры, которую не предполагается использовать).

Управление наложением

При включенном визуальном (`show_visual: yes`) и термическом (`show_thermal: yes`) изображениях, термическое будет налагаться на визуальное в регион, заданный в свойстве `overlay_thermal_on_visual` (начальная точка с координатами `(x_start, y_start)` , ширина региона - `thermal_region_width` , высота - `thermal_region_height`).

Если отключить визуальное изображение (`show_visual: no`), то термическое развернется на весь экран.

Если отключить термическое (`show_thermal: no`), то будет транслироваться только визуальное.

Если отключить оба – изображение перестанет транслироваться.

Центровка

Для центровки результирующего изображения (из-за физического сдвига объективов камер термическое будет сдвинутого относительно центра визуального) следует использовать свойства раздела `final_frame_cropping` . Принцип аналогичен предыдущему разделу.

Детекторы контуров (edge detectors)

Для облегчения понимания термического изображения, есть возможность включения наложения на него соответствующих контуров визуального изображения. В программе реализована поддержка двух алгоритмов детектирования контуров: Sobel и Canny. В соответствующих разделах конфига (`apply_sobel` и `apply_canny`) можно поэкспериментировать с некоторыми свойствами детекторов.

Колоризация

Для большей наглядности, тепловизионное изображение можно колорифицировать. Для этого необходимо выставить свойство `apply_colormap: yes`.

Конкретную схему колорификации можно выбрать из списка, указав порядковый номер схемы в свойстве `colormap`.

Свойства транслируемого видео и сохраняемых файлов

Свойства транслируемого видео можно изменять в разделе `output_video`. Делать это стоит с четким пониманием того, что делаешь - неправильные настройки могут сломать трансляцию. Например, разрешение видео должно совпадать с настройками gstreamer'a, который осуществляет последующее кодирование.

Для настройки сохраняемого на флеш-накопитель видео следует пользоваться группой свойств `encode_video`.

Каталоги для сохранения фото и видеофайлов можно указать в свойствах `path_to_save`.

Настройки дополнительной отображаемой информации

Скрывать\показывать частоту кадров в секунду визуального и термического изображений можно при помощи свойства `show_fps`.

Показывать\скрывать перекрестие с температурой центрального пикселя можно при помощи свойства `draw_temp`.

Подстройка температуры

В зависимости от используемого тепловизора, может потребоваться корректировка температурного диапазона. В секции `temperature_calibration` Доступны две настройки:

- Сдвиг (`offset`) - линейный сдвиг температурного диапазон на указанное число.
- Масштаб (`scale`) - пропорциональное изменение шкалы.

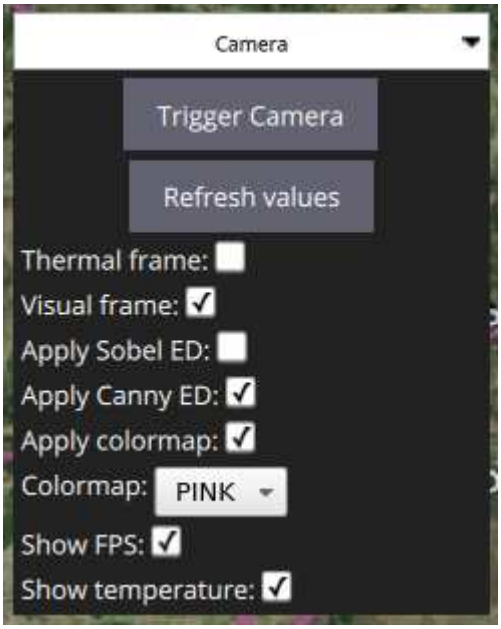
Выравнивание градиента температуры от центра к краям

Если на тепловизионном изображении наблюдается "рамка", вызванная увеличением определяемой температуры к краям, то следует применить выравнивающий фильтр (виньетирование). Настройки располагаются в секции `vignette`:

- `enable` - включить-выключить наложение фильтра.
- `show_center` - показать точку "центра" для отладки ее положения.
- `center_x` - X-координата "центра".

- `center_y` - Y-координата "центра".
- `max_decrement_relative` - максимальное уменьшение температуры к краям.

Виджет управления камерой для QGroundControl



Где взять

QGC с необходимый функционалом для управления "COEX DuoCam" можно скачать в нашем репозитории. Скачивать следует тот релиз, в названии которого присутствует слово `duocam` с максимальным порядковым номером.

Как использовать

При открытии виджета QGC пошлет MAVLink-сообщения с запросом текущих значений свойств. Необходимо дождаться, пока они все до конца загрузятся. Если загрузка не завершается в течение минуты, необходимо нажать кнопку "Refresh values".

При изменении значений свойств QGC осуществляет коммуникацию с DuoCam посредством протокола MAVLink, поэтому нет смысла очень часто нажимать контролы - наоборот, после нажатия следует дождаться изменения картинки и только потом переходить к следующему свойству.

Подробнее об архитектуре DuoCam и программе `duocam-mavlink`.

Конфигурационный файл

Для более глубокой настройки DuoCam можно использовать конфигурационный файл.

Как добраться

Для редактирования конфигурационного файла необходимо извлечь microSD-карту из RPi4, вставить в кард-ридер своего компьютера (от операционной системы требуется возможность читать файловую систему ext4), открыть файл по адресу `<microSD>/etc/duocam/camera.yaml`.

Также можно зайти по SSH на работающую RPi4 и отредактировать конфигурационный файл внутри системы по тому же пути.

Образец конфигурационного файла

```
show_visual: yes
show_thermal: yes
flip_thermal: no
apply_sobel:
  enable: yes
  sobel_scale: 3
  sobel_delta: 0
apply_canny: no
draw_temp: yes

output_video:
  device: /dev/video1
  width: 1280
  height: 720
  bytes_per_pixel: 3
  framerate: 20

path_to_save_photos: /media/usb0/
path_to_save_videos: /media/usb0/

encode_video:
  width: 720
  height: 480
  framerate: 10

thermal_camera_models:
  seek_thermal_pro:
    enable: no
    sensor_resolution:
      width: 320
      height: 240
    overlay_thermal_on_visual:
      # These values are for GitUp3
      x_start: 282
      y_start: 102
      thermal_region_width: 600
      thermal_region_height: 400
      transparency: 0.3
      # These values are for Raspicam
      #  x_start: 425
      #  y_start: 155
      #  thermal_region_width: 900
      #  thermal_region_height: 600
    final_frame_cropping:
      enable: yes
      x_start: 0
      y_start: 0
      width: 1160
      height: 610

  seek_thermal:
    enable: yes
    sensor_resolution:
      width: 206
      height: 156
    overlay_thermal_on_visual:
      x_start: 282
      y_start: 102
      thermal_region_width: 600
      thermal_region_height: 400
      transparency: 0.3
    final_frame_cropping:
      enable: yes
      x_start: 0
      y_start: 0
```

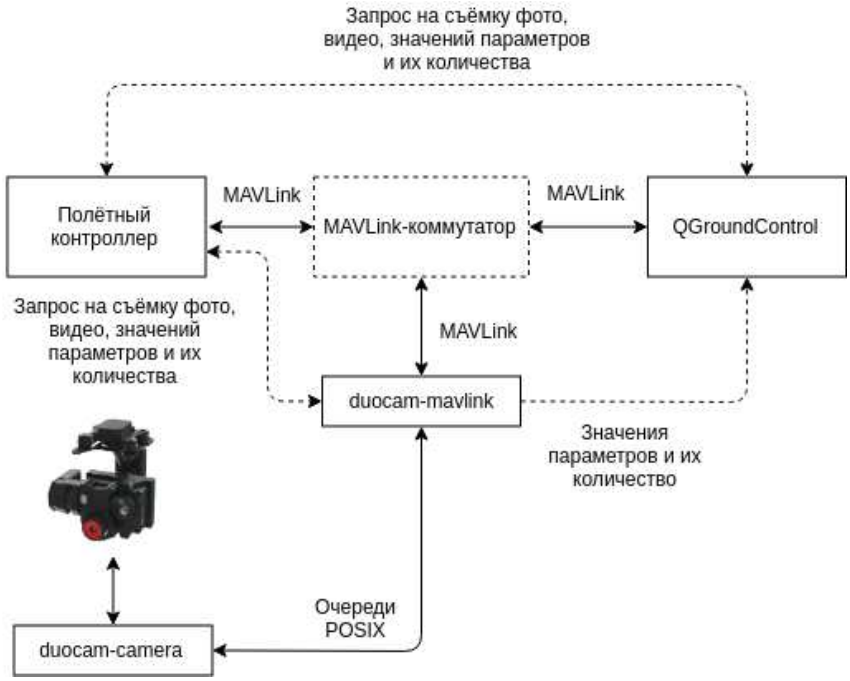
```
      width: 1160
      height: 610

# Colormaps are taken from: https://docs.opencv.org/2.4/modules/contrib/doc/fa
# COLORMAP_AUTUMN = 0,
# COLORMAP_BONE = 1,
# COLORMAP_JET = 2,
# COLORMAP_WINTER = 3,
# COLORMAP_RAINBOW = 4,
# COLORMAP_OCEAN = 5,
# COLORMAP_SUMMER = 6,
# COLORMAP_SPRING = 7,
# COLORMAP_COOL = 8,
# COLORMAP_HSV = 9,
# COLORMAP_PINK = 10,
# COLORMAP_HOT = 11
apply_colormap:
  enable: yes
  colormap: 4
```

Виртуальная MAVLink-камера COEX DuoCam

Полётные контроллеры поддерживают разные способы взаимодействия с внешними камерами, включая протокол MAVLink. Обычно коммуникация с помощью данного протокола требует использования UART-порта на полётном контроллере, но есть возможность работать с камерой в основном потоке MAVLink-телеметрии с БПЛА.

Утилита `duocam-mavlink` отвечает за работу виртуальной MAVLink-камеры и встраивается в общую телеметрию между полётным контроллером и QGroundControl.



На блок-схеме прямыми линиями обозначены взаимодействия между блоками, пунктирными линиями уточняется характер взаимодействия.

> Протокол взаимодействия находится в процессе модификации. В новых версиях планируется избавиться от прямой отправки значений параметров и их количества от `duocam-mavlink` к QGroundControl.

`duocam-camera` и `duocam-mavlink` обмениваются данными с помощью очередей POSIX. Имена очередей и формат сообщений доступен в репозитории duocam-common.

Для объединения блоков, взаимодействующих через MAVLink, можно использовать любой MAVLink-коммутатор/маршрутизатор, который либо позволяет отключить таблицу коммутации, либо заполняет её по схеме *MAVLink ID:Component ID* (например, `cmavnode`, `mavlink-fast-switch`, `mavlink-switch`).

> При использовании `mavlink-fast-switch` требуется использовать `mavlink-serial-bridge`, либо любой другой мост для передачи MAVLink из последовательного порта в UDP, так как `mavlink-fast-switch` работает только с UDP.

Конфигурационный файл

Для редактирования конфигурационного файла необходимо извлечь microSD-карту из RPi4, вставить в кард-ридер своего компьютера (от операционной системы требуется возможность читать файловую систему ext4), открыть файл по адресу `<microSD>/etc/duocam/mavlink.yaml`.

Также можно зайти по SSH на работающую RPi4 и отредактировать конфигурационный файл внутри системы по тому же пути.

Для корректной работы `duocam-mavlink` требуется *MAVLink ID* полётного контроллера (параметр `vehicle_id`) и `MAVLink ID` QGroundControl (параметр `qgc_vehicle_id`). `duocam-mavlink` ожидает телеметрию на UDP (параметры `ip` и `port`) и отправляет сообщения HEARTBEAT с частотой, согласно параметру `heartbeat_frequency`, представляясь, как компонент полётного контроллера `MAV_COMP_ID_CAMERA`.

Максимальная задержка от `duocam-camera` задаётся параметром `command_timeout`.

Если система DuoCam используется для полётов внутри помещений, то должен быть включен параметр `no_gps`.

Пример конфигурационного файла

```yaml
# MAVLink vehicle ID that owns the camera
vehicle_id: 1
# Enable this flag for indoor use
no_gps: False
# QGC vehicle ID
qgc_vehicle_id: 255
mavlink:
  # IP address of the interface to listen port on (0.0.0.0 for all interfaces)
  ip: 127.0.0.1
  # UDP port
  port: 14540
  # Heartbeat frequency (Hz)
  heartbeat_frequency: 1.0
# Camera driver command timeout (s)
command_timeout: 3.0
```

Пример конфигурационного файла для `mavlink-fast-switch`

```
# MAVLink endpoints
endpoints:
  # UAV endpoint
  - name: "uav"
    local:
      port: 14588
  # DuoCam MAVLink endpoint
  - name: "duocam-mavlink"
    remote:
      ip: "127.0.0.1"
      port: 14540
  # GCS endpoint
  - name: "gcs"
    remote:
      ip: "127.0.0.1"
      port: 14550
# Enable MAVLink ID table
# HINT: Can't use this feature with duocam
id-table: False
```

Пример конфигурационного файла для `mavlink-serial-bridge`

```
# Serial device settings
serial:
  # Device file
  device: "/dev/ttyS0"
  # Baudrate
  baudrate: 57600
  # Flow control (hardware, none)
  flow: none
  # Software serial TX buffer (bytes) (2048 by default)
  tx-buffer: 2048
# UDP port settings

udp:
  # Remote host settings (optional, listening mode if not presented)
  remote:
    ip: 127.0.0.1
    port: 14588
    # Lock remote host on the initial value (optional, False by default)
    lock: True
    # Broadcast mode (optional, False by default)
    broadcast: False
  # Local settings (optional, all interfaces and a random port by default)
  local:
    # Local IP address (0.0.0.0 to listen on all interfaces) (optional, all int
    ip: 127.0.0.1
    # Local UDP port (0 to select a random free port) (optional, 0 by default)
    port: 0
```

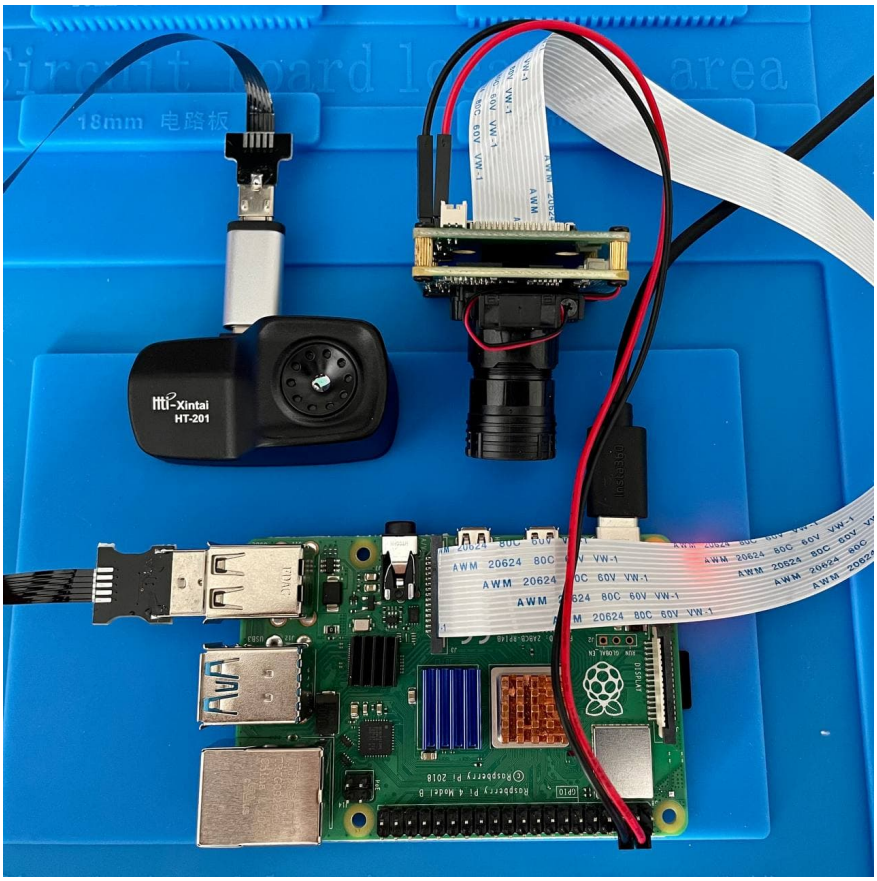
Настройка и трансляция изображения с DuoCam

DuoCam включает в себя две камеры: визуальная камера VEYE-MIPI-327E (MIPI Camera) и тепловизор HT-201 (HTI Thermal Camera HT-201). В этой статье мы рассмотрим, как установить их драйвера, настроить RTSP-трансляцию и вывести изображение с камер в ROS-топики.

Для начала подготовьте Raspberry Pi 4 со свежим образом Клевера. Подключите Raspberry к Wi-Fi роутеру с интернетом по инструкции в статье "Настройка Wi-Fi".

> Есть альтернативный, устаревший, но более простой способ подключения к Wi-Fi-роутеру. Для этого в файл `/etc/network/interfaces` нужно добавить строчки:
>
> ```
> auto wlan0
> allow-hotplug wlan0
> iface wlan0 inet dhcp
> wpa-ssid "Your Wi-Fi's Name"
> wpa-psk "Your Wi-Fi's Password"
> ```

Подключите к Raspberry Pi камеры VEYE-MIPI-327E и тепловизора HT-201:



Установка драйвера камеры VEYE-MIPI-327E

> Основная статья:
> http://wiki.veye.cc/index.php/V4L2_mode_for_Raspberry_Pi

Для начала обновите информацию о пакетах командой:

```
sudo apt update
```

Далее выполните команды для установки драйвера камеры:

```
git clone https://github.com/veyeimaging/raspberrypi_v4l2.git
cd raspberrypi_v4l2/release/
chmod +x *
sudo ./install_driver.sh veye327
```

> Важно, чтобы версия ядра Linux поддерживалась разработчиками драйвера камеры. Узнать свою версию ядра можно командой `uname -r`. Список поддерживаемых версий ядра Linux можно посмотреть в репозитории драйвера для камеры VEYE-MIPI-327E. Откатиться на более старую версию ядра можно командой `sudo rpi-update commit-id` где вместо `commit-id` нужно подставить ID коммита с нужной версией ядра, список всех коммитов можно посмотреть тут. Например, команда `sudo rpi-update 1fad1d2beceaaf96e84e3f5caa79f52b87332e89` откатывает ядро Linux к версии 5.10.52.

Перезагрузите Raspberry Pi командой `sudo reboot`. После перезагрузки проверьте, что драйвер установился корректно с помощью команд:

```
dmesg | grep veye
v4l2-ctl --list-devices
v4l2-ctl -d /dev/video0 -V --info --list-formats-ext
```

Установка драйвера тепловизора Seek Thermal CompactPRO

Драйвер для тепловизора HT-201 такой же, как и для тепловизора Seek Thermal CompactPRO.

Сначала установите необходимые пакеты командой:

```
sudo apt install cmake libopencv-dev libusb-1.0-0-dev v4l2loopback-utils -y
```

Далее выполните следующие команды для установки драйвера тепловизора:

```
git clone https://github.com/OpenThermal/libseek-thermal.git
cd libseek-thermal
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

Далее создайте файл `/etc/udev/rules.d/seekpro.rules` и добавьте в него одну строчку:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="289d", ATTRS{idProduct}=="0011", MODE="066
```

Далее в файл `/etc/modules` добавьте строчку:

```
v4l2loopback
```

Далее создайте файл `/etc/modprobe.d/v4l2loopback.conf` и добавляем в него одну строчку:

```
options v4l2loopback devices=1 exclusive_caps=1 video_nr=5 card_label="Thermal
```

Далее выполните апдейт системы и перезагрузитесь:

```
sudo update-initramfs -c -k $(uname -r)
sudo reboot
```

После перезагрузки проверьте, что в системе появился интерфейс `/dev/video5`:

```
v4l2-ctl --list-devices
```

Далее нам нужно настроить автозапуск видео с тепловизора при включении Raspberry Pi. Для этого в домашней директории создайте файл `duocam_autostart.sh` и добавьте в него команду для запуска видео с тепловизора Seek Thermal CompactPRO:

```
seek_viewer --camtype=seekpro --colormap=1 --rotate=0 --mode=v4l2 --output=/dev
```

> Чтобы изображение с тепловизора имело красно-синюю раскраску используйте значение параметра `--colormap=2`.

Чтобы корректно запускать файл, необходимо присвоить ему соответствующие флаги доступа.

```
chmod a+x duocam_autostart.sh
```

Для того, чтобы видео с тепловизора запускалась каждый раз при включении системы, необходимо создать стартап-скрипт с помощью менеджера systemd. Создайте файл `/etc/systemd/system/duocam.service` и запишите в него соответствующие строки:

```
[Unit]
Description=DuoCam

[Service]
ExecStart=/bin/bash /home/pi/duocam_autostart.sh

[Install]
WantedBy=multi-user.target
```

Осталось только инициализировать ваш скрипт в системе и он будет запускаться при каждом ее включении.

```
sudo systemctl enable duocam.service
```

Перезагрузите Raspberry Pi командой `sudo reboot`. Проверьте, что видео с тепловизора доступно в интерфейсе `/dev/video5`:

```
v4l2-ctl -d /dev/video5 -V --info --list-formats-ext
```

Трансляция видео по RTSP

Чтобы запустить RTSP-трансляцию сначала нужно установить утилиту `gst-rtsp-launch`:

```
sudo apt install gst-rtsp-launch -y
```

Трансляция с тепловизора, разрешение 320x240:

```
gst-rtsp-launch "( v4l2src device=/dev/video5 ! video/x-raw,width=320,height=24
```

Трансляция с камеры VEYE-MIPI-327E, разрешение 1920x1080:

```
gst-rtsp-launch "( v4l2src device=/dev/video0 ! video/x-raw,format=UYVY,width=1
```

Трансляция с камеры VEYE-MIPI-327E, разрешение 1280x720:

```
gst-rtsp-launch "( v4l2src device=/dev/video0 ! video/x-raw,format=UYVY,width=1
```

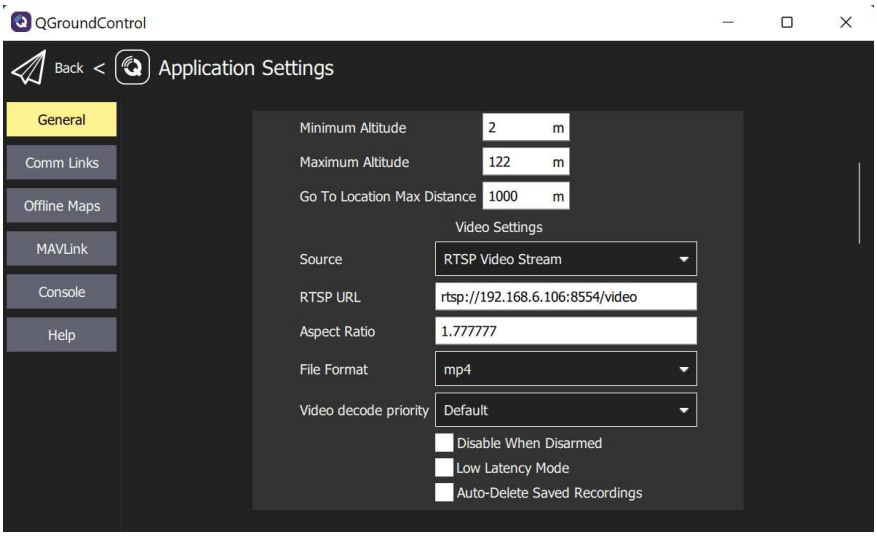
Трансляция с обеих камер, "картинка в картинке", разрешение 1280x720:

```
gst-rtsp-launch "( v4l2src device=/dev/video0 ! video/x-raw,format=UYVY,width=
```

Трансляция с обеих камер, "картинка в картинке", с обрезкой изображения тепловизора по краям, разрешение 1280x720:

```
gst-rtsp-launch "( v4l2src device=/dev/video0 ! video/x-raw,format=UYVY,width=
```

Смотреть трансляцию можно, например, в QGroundControl указав в настройках:



Для того, чтобы трансляция запускалась автоматически при включении Raspberry Pi добавьте нужную команду запуска трансляции в файл `duocam_autostart.sh`.

Создание ROS-топиков с изображениями с камер

Чтобы доступ к изображению с камер был доступен через ROS-топики, нужно создать для обоих камер launch-файлы для запуска в пакете `clover`.

Для камеры VEYE-MIPI-327E создайте файл `~/catkin_ws/src/clover/clover/launch/front_camera.launch` и добавьте в него следующий текст:

```
<launch>
    <!-- camera nodelet manager -->
    <node pkg="nodelet" type="nodelet" name="front_camera_nodelet_manager" args
        <param name="num_worker_threads" value="2"/>
    </node>

    <!-- camera node -->
    <node pkg="nodelet" type="nodelet" name="front_camera" args="load cv_camera
        <!-- v4l2 device -->
        <param name="device_id" value="0"/>

        <!-- camera FPS -->
        <param name="cv_cap_prop_fps" value="30"/>

        <!-- camera resolution -->
        <param name="image_width" value="1920"/>
        <param name="image_height" value="1080"/>
    </node>
</launch>
```

Для тепловизора создайте файл `~/catkin_ws/src/clover/clover/launch/thermal_camera.launch` и добавьте в него следующий текст:

```
<launch>
    <!-- v4l2 device -->
    <arg name="thermal_device" default="/dev/video5"/>

    <!-- camera nodelet manager -->
    <node pkg="nodelet" type="nodelet" name="thermal_camera_nodelet_manager" ar
        <param name="num_worker_threads" value="2"/>
    </node>

    <!-- camera node -->
    <node pkg="nodelet" type="nodelet" name="thermal_camera" args="load cv_came
        <param name="device_path" value="$(arg thermal_device)"/>

        <!-- camera FPS -->
        <param name="cv_cap_prop_fps" value="30"/>

        <!-- camera resolution -->
        <param name="image_width" value="320"/>
        <param name="image_height" value="240"/>
    </node>
</launch>
```

Далее отредактируйте файл `~/catkin_ws/src/clover/clover/launch/clover.launch`. Установите параметры `main_camera` и `optical_flow` в значение `false`:

```
    <arg name="main_camera" default="false"/>
    <arg name="optical_flow" default="false"/>
```

Добавьте два новых параметра для новых камер:

```
    <arg name="front_camera" default="true"/>
    <arg name="thermal_camera" default="true"/>
```

После секции `<!-- main camera -->` добавьте включение новых камер:

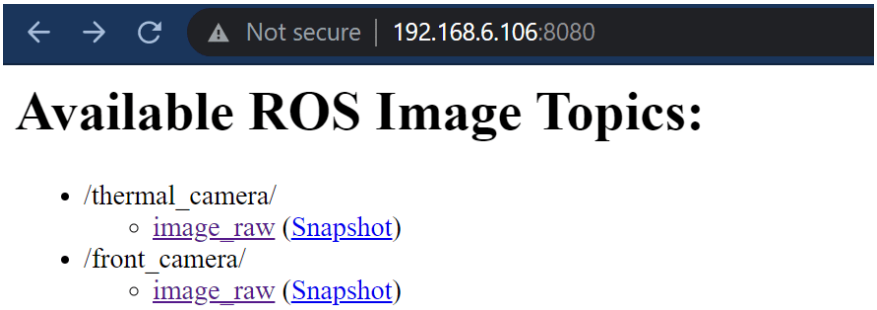
```
    <!-- front camera -->
    <include file="$(find clover)/launch/front_camera.launch" if="$(arg front_c

    <!-- thermal camera -->
    <include file="$(find clover)/launch/thermal_camera.launch" if="$(arg therm
```

Теперь осталось только перезагрузить сервис `clover` командой:

```
sudo systemctl restart clover
```

И новые топики с камерами появятся в списке:



Обработка изображения с камер с помощью GStreamer

Изображение с камер можно обрабатывать как методами OpenCV, так и GStreamer. С помощью GStreamer можно, например, обрезать или изменить размер кадра. В этом разделе расскажем, как изменить разрешение передней камеры с 1920x1080 на 1280x720, а также вырезать из изображения с тепловизора (обрезать края кадра) в центральной части квадрат размером 120x120 и растянуть его до 240x240.

Для начала нам нужно создать еще 2 виртуальных видеоустройства, куда мы направим вывод с GStreamer. В файле `/etc/modprobe.d/v4l2loopback.conf` измените содержимое на:

```
options v4l2loopback devices=3 exclusive_caps=1 video_nr=5,6,7 card_label="Ther
```

Теперь нам нужно добавить непосредственно команды GStreamer, которые делают то, что нам нужно. Для этого в файле `duocam_autostart.sh` измените содержимое на:

```
seek_viewer --camtype=seekpro --colormap=1 --rotate=0 --mode=v4l2 --output=/dev

sleep 10

gst-launch-1.0  v4l2src device=/dev/video5 ! video/x-raw,width=320,height=240

gst-launch-1.0  v4l2src device=/dev/video0 ! video/x-raw,format=UYVY,width=192(
```

В заключение измените содержимое файла `~/catkin_ws/src/clover/clover/launch/front_camera.launch` на следующий текст:

```
<launch>
    <!-- v4l2 device -->
    <arg name="front_device" default="/dev/video7"/>

    <!-- camera nodelet manager -->
    <node pkg="nodelet" type="nodelet" name="front_camera_nodelet_manager" args
        <param name="num_worker_threads" value="2"/>
    </node>

    <!-- camera node -->
    <node pkg="nodelet" type="nodelet" name="front_camera" args="load cv_camera
        <!-- v4l2 device -->
        <param name="device_path" value="$(arg front_device)"/>
    </node>
</launch>
```

А также измените содержимое файла `~/catkin_ws/src/clover/clover/launch/thermal_camera.launch` на следующий текст:

```
<launch>
    <!-- v4l2 device -->
    <arg name="thermal_device" default="/dev/video6"/>

    <!-- camera nodelet manager -->
    <node pkg="nodelet" type="nodelet" name="thermal_camera_nodelet_manager" ar
        <param name="num_worker_threads" value="2"/>
    </node>

    <!-- camera node -->
    <node pkg="nodelet" type="nodelet" name="thermal_camera" args="load cv_came
        <param name="device_path" value="$(arg thermal_device)"/>
    </node>
</launch>
```

После перезагрузки Raspberry Pi в топиках с камерами будут обработанные изображения.

Мероприятия

"Клевер" применяется в в большом количестве образовательных мероприятий и соревнований: WorldSkills, Олимпиада НТИ, Copter Hack, Всероссийская Робототехническая Олимпиада и других.

Этот раздел содержит статьи, написанные специально к тому или иному мероприятию.

CopterHack 2023

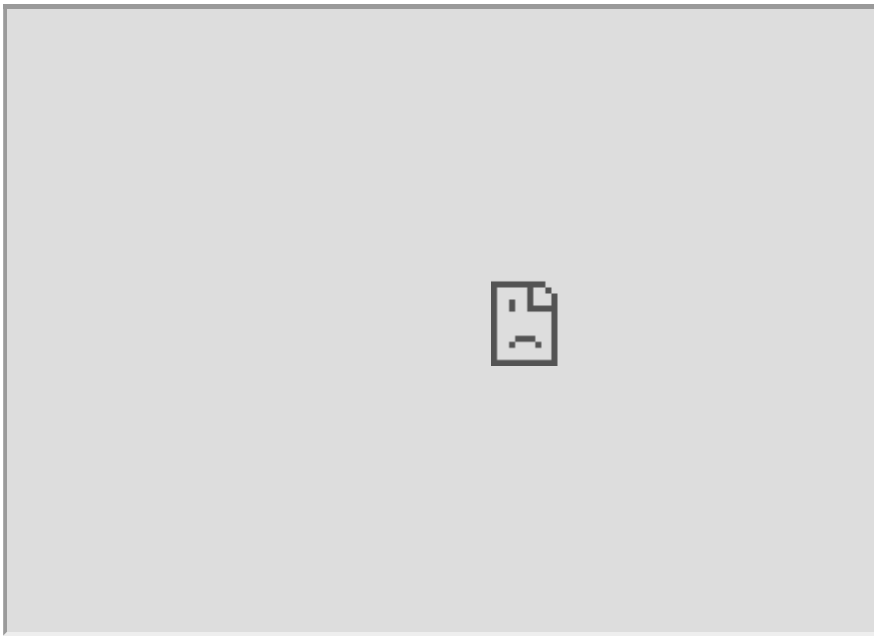
CopterHack 2023 — это международный конкурс по разработке проектов по летающей робототехнике с открытым исходным кодом. Основным языком конкурса является английский.

Ознакомиться со статьями команд-финалистов предыдущих лет можно в статьях о CopterHack 2021, CopterHack 2022.

На конкурс принимаются проекты с открытым исходным кодом и совместимые с платформой квадрокоптера "Клевер". На протяжении конкурса команды работают на собственными идеями и разработками, приближая их к состоянию готового продукта. В этом участникам помогают эксперты отрасли через лекции и регулярную обратную связь.

Финал конкурса CopterHack 2022 прошел 27 мая 2023. Победителями стала команда 🇷🇺 **Clover Cloud Platform**.

Полный стрим финала



Проекты участников конкурса

Место	Команда	Проект	Балл
1	🇷🇺 Clover Cloud Team	Clover Cloud Platform	21.7
2	🇧🇾 FTL	Advanced Clover 2	21
3	🇨🇦 Clover with Motion Capture System	Clover with Motion Capture System	20.5
4	🇧🇷 Atena	Swarm in Blocks 2	20.3
5	C305	Система радио-навигации	17.5
6	🇮🇳 DJS PHOENIX	Autonomous Racing Drone	14.6
7	Лицей №128	Платформа для зарядки квадрокоптера	13.7
×	🇰🇬 Zavarka	Система обмена грузами с помощью конвейера	
×	FSOTM	Дрон-перехватчик	
×	🇰🇬 Бездомные	Дрон-бездомный	
×	Digital otters	Digital otters	
×	Light Flight	Сопровождение БПЛА при посадке	
×	🇰🇬 LiveSavers	LiveSavers	
×	XenCOM	Bound by fate	
×	Ava_Clover	DoubleClover	
×	TPU_1	Совместная транспортировка груза	
×	🇷🇺 TPU_2	Алгоритм полета сквозь лесную местность	

Смотрите все оценки по критериям в полной таблице.

Этапы CopterHack 2023

Отборочный и проектный этапы конкурса проходят в онлайн-формате, формат проведения финала – гибридный (оффлайн + онлайн). Конкурс подразумевает ежемесячные апдейты от команд с получением регулярной обратной связи от жюри. Для участия в заключительном этапе необходимо подготовить финальное видео и презентацию о результатах проекта.

1. Отборочный этап. Подача заявок (до 31 октября 2022).
2. Проектный этап. Менторство проектов (1 ноября 2022 — 28 февраля 2023).
3. Подготовка финального видео (1 — 31 марта 2023).
4. Заключительный этап. Финальная защита проектов на английском языке (23 апреля 2023).

Условия и критерии оценки

Условия, предъявляемые к проектам:

1. Открытый исходный код/модели/схемы/чертежи.
2. Совместимость с платформой "Клевер".

Критерии оценивания жюри в финале:

1. Готовность и статья (макс. 10 баллов): степень готовности проекта; доступное и понятное описание проекта в статье; прикреплены код с комментариями, схемы, чертежи. По статье должно быть возможно повторить проект, получить результат.
2. Объем проделанной работы (макс. 6 баллов): объем проделанной командой работы в рамках CopterHack, ее сложность и технический уровень.
3. Полезность для Клевера (макс. 6 баллов): актуальность применения на практике в платформе Клевер и PX4, потенциальный уровень спроса на разработку со стороны других пользователей Клевера.
4. Презентация на финале (макс. 3 балла): качество и зрелищность финальной презентации; полнота освещения проекта; демонстрация; ответы на вопросы жюри.

Призовой фонд

Призы от компании COEX по результатам оценивания жюри на финале:

- I место: $3000.
- II место: $2000.
- III место: $1000.
- IV место: $500.
- V место: $500.

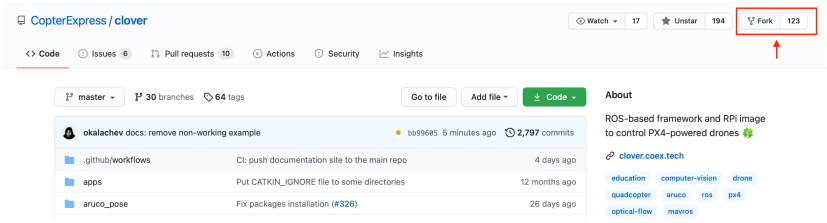
Партнеры конкурса могут поощрить команды по дополнительным критериям, выявленным в результате оценки проектов в ходе финала.

Как подать заявку?

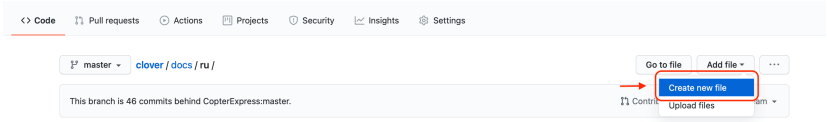
> Для подачи заявки необходимо иметь аккаунт на GitHub.

Подготовьте вашу заявку и пришлите ее в виде Draft Pull Request в репозиторий Клевера.

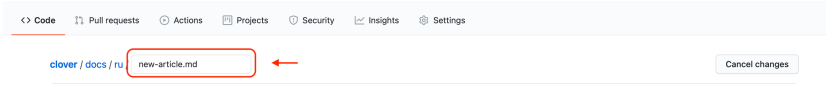
1. Сделайте форк репозитория Клевера:



2. На странице вашего форка зайдите в раздел `docs/ru` и создайте новый файл в формате Markdown:



3. Введите название вашей статьи. Например, `new-article.md`



4. Оформите вашу заявку в соответствии с рекомендуемым шаблоном:

```
# Название проекта

[CopterHack-2023](copterhack2023.md), команда **Название команды**.

## Информация о команде

Состав команды:

(Опишите состав команды: имя и фамилия, контакты (имя пользователя в Teleg

* Александр Соколов, @aleksandrsokolov111, инженер.
* Елена Смирнова, @elenasmirnova111, программист.

## Описание проекта

### Идея проекта

Опишите кратко идею и стадию проекта.

### Планируемые результаты

Опишите как вы видите результат проекта.

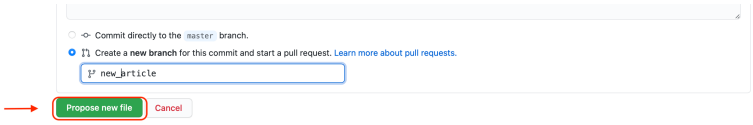
### Использование платформы "Клевер"

Опишите как в вашем проекте будет использоваться платформа "Клевер".

### Дополнительная информация по желанию участников

Например, информация об опыте работы команды над проектами, прикрепить ссы
```

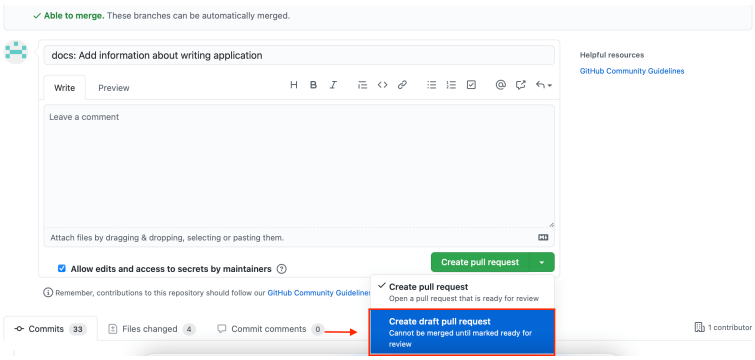
5. Перейдите вниз страницы и создайте новую ветку с названием вашей статьи:



> Не добавляйте ваши изменения непосредственно в ветку `master`, создайте новую ветку.

6. При необходимости поместите дополнительные визуальные материалы в папку `docs/assets` и оформите на них ссылки в вашей статье.

7. Сделайте Draft Pull Request вашей ветки в master Клевера:



8. В комментариях Pull Request вам будет дана обратная связь по заявке.

9. Обратите внимание на блок *Checks*, в графе Documentation должна стоять галочка. Если там стоит крестик, перейдите по ссылке *Details*, чтобы увидеть список проблем с оформлением статьи. При необходимости изменения добавляемых файлов, меняйте их в вашей ветке – изменения будут появляться в Pull Request автоматически. **Не создавайте новый Pull Request для одной и той же заявки**.

10. На протяжении конкурса вы будете работать над этим документом, приближая его к состоянию статьи. В документе будет видна история разработки и ежемесячные апдейты. К финалу конкурса вы сможете опубликовать вашу статью, это и будет результат вашей работы в CopterHack.

Участники конкурса будут добавлены в Telegram-группу, куда можно отправлять первый апдейт и получить обратную связь от жюри. Для команд-участников предусмотрена скидка 40% на конструктор программируемого квадрокоптера "Клевер".

> Ограничения по возрасту, образованию и количеству человек в команде отсутствуют.

Конкурс статей участников проектов CopterHack 2023

Наши участники уже 2 года работают над передовыми проектами в области летающей робототехники. В этом году мы хотим ввести новый конкурс, стимулирующий участников презентовать результаты исследований, выполняющихся в рамках конкурса на престижных международных конференциях, а также публиковать их в российских и международных журналах по тематике конкурса.

На конкурс принимаются оригинальные статьи в следующих номинациях:

- $2000 за статью в журнале первого квартиля (Q1), индексируемом в Scopus, Web of Science.
- $1000 за статью журнале, индексируемом в Scopus, Web of Science.
- $500 за статью, опубликованную в сборнике материалов конференции (Conference Proceedings), индексируемые в Scopus, Web of Science.

> Как узнать квартиль журнала в Scopus и WOS.

Правила:

1. Статья должна отражать результаты проекта, разработанного в рамках CopterHack 2023.
2. Статья должна быть принята к печати к моменту подачи заявки на участие в конкурсе статей.
3. В acknowledgment следует указать, что работа выполнена в рамках конкурса.

Прием заявок: до 10 декабря 2023 года. Прием заявок осуществляется через Google Форму.

Объявление результатов: 24 декабря 2023 года.

По всем вопросам: группа CopterHack в Telegram.

> Если вы хотите стать партнером конкурса или членом жюри, обращайтесь к Олегу Понфиленку в Telegram.

CopterHack 2022

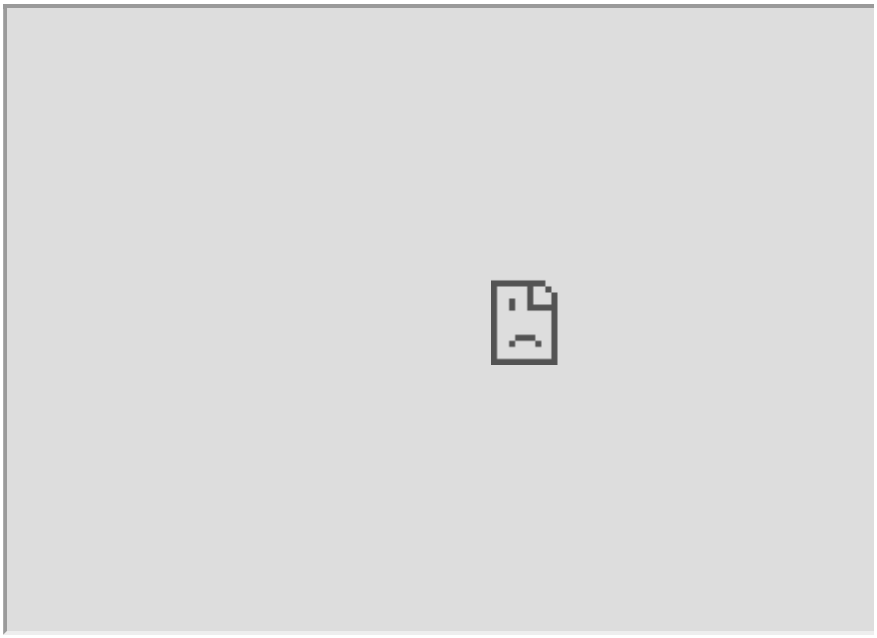
CopterHack 2022 — это международный конкурс по разработке проектов по летающей робототехнике с открытым исходным кодом. CopterHack 2022 имеет основное направление со свободным выбором темы проекта, а также отдельную номинацию "кейсы компании". Основным языком конкурса является английский.

Ознакомиться со статьями команд-финалистов предыдущего года можно в статье о CopterHack 2021.

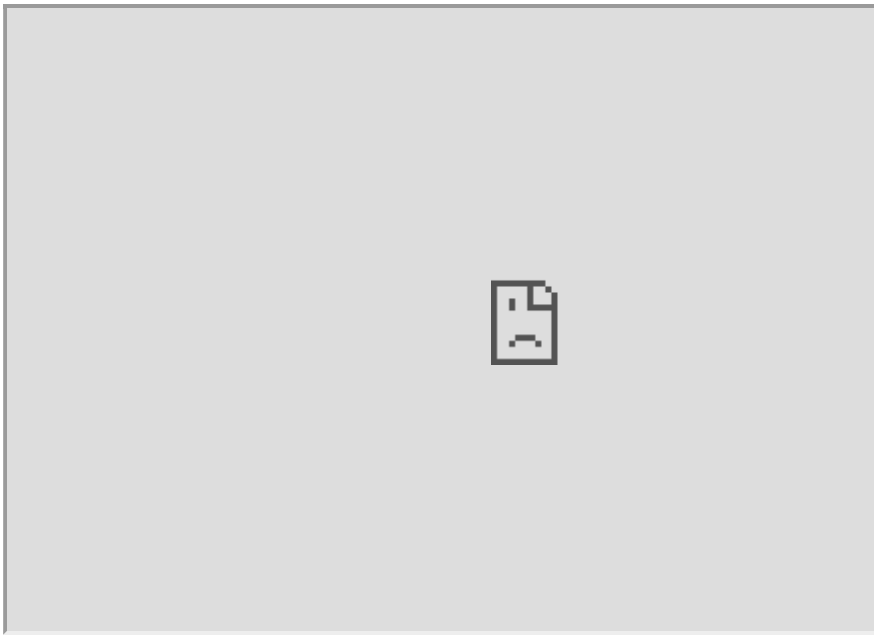
На конкурс принимаются проекты с открытым исходным кодом и совместимые с платформой квадрокоптера "Клевер". На протяжении конкурса команды работают на собственными идеями и разработками, приближая их к состоянию готового продукта. В этом участникам помогают эксперты отрасли через лекции и регулярную обратную связь.

Финал конкурса CopterHack 2022 прошел 23 апреля 2022. Победителями стала команда 🇧🇷 **Atena - Grupo SEMEAR**.

Полный стрим финала



Видео



Проекты участников конкурса

Место	Команда	Проект	Балл
1	🇧🇷 Atena - Grupo SEMEAR	Swarm in Blocks	21.6
2	🇧🇾 FTL	Advanced Clover 2	19.9
3	🇷🇺 Clover Rescue Team	Rescue Clover	17.7
4	C305	Система мониторинга воздуха	17.3
5	Space clowns	Copter For Space	16.2
6	CopterCat	CopterCat	16.1
7	Stereo	Neural obstacle avoidance	15.85
8	🇮🇳 DJS Phoenix	Autonomous valet parking drone assistance	11.7
×	R.S.	Drone Hawk	
×	🇲🇾 Moopt	IoT Water Monitoring & Optimization	
×	Дрой Ронов	Clover Swarm	
×	SPECTRE	SPECTRE	
×	🇰🇬 Alatoo University Team	Облачная платформа для симулятора Клевера	
×	Clevertron	Clevertron	
×	🇵🇱 Edgenoon	Neural and vision-based landing method	
×	🇩🇪 Inondro	Inondro Pix	
×	SolidEye	Разработка лидара без движущихся частей	
×	🇰🇬 AI_U_CLOVER	AIU_CLOVER	
×	🇻🇳 Dragon&Tanker	Dragon&Tanker	
×	V-NAV	Visual Navigation	
×	Джедаи 1581	Ретранслятор на базе Клевера	
×	Lucky flight	Swarm of Improved Clover	
×	🇺🇸 EnviroFleet	EnviroFleet	
×	Бизнес-гуси	Drone Rover Climbing System	
×	🇷🇺 fuall	Доставка дронами	

Место	Команда	Проект	Балл
×	🇷🇺 Scout_Drone	Создание поисково-спасательного беспилотного летательного аппарата	

× – команды, не дошедшие до финала.

Смотрите все оценки по критериям в полной таблице.

Направление "кейс компании"

Команды приглашаются принять участие в работе над следующими кейсами компании:

1. Разработка платы полетного контроллера Pixhawk FMUv6U размером 55*40 мм и возможностью установки Raspberry Pi CM4.
2. Облачная платформа для симулятора Клевера по аналогии или на основе ROS Development Studio.

Список кейсов может быть расширен.

Этапы CopterHack 2022

Отборочный и проектный этапы конкурса проходят в онлайн-формате, формат проведения финала – гибридный (оффлайн + онлайн). Конкурс подразумевает ежемесячные апдейты от команд с получением регулярной обратной связи от жюри. Для участия в заключительном этапе необходимо подготовить финальное видео и презентацию о результатах проекта.

1. Отборочный этап. Подача заявок (10 июня — 31 октября 2021).
2. Проектный этап. Менторство проектов (10 июня 2021 — 28 февраля 2022).
3. Подготовка финального видео (1 — 31 марта 2022).
4. Заключительный этап. Финальная защита проектов на английском языке (9 — 10 апреля 2022).

Условия и критерии оценки

Условия, предъявляемые к проектам:

1. Открытый исходный код/модели/схемы/чертежи.
2. Совместимость с платформой "Клевер".

Критерии оценивания жюри в финале:

1. Готовность и статья (макс. 10 баллов): степень готовности проекта; доступное и понятное описание проекта в статье; прикреплены код с комментариями, схемы, чертежи. По статье должно быть возможно повторить проект, получить результат.

2. Объем проделанной работы (макс. 6 баллов): объем проделанной командой работы в рамках CopterHack 2022, ее сложность и технический уровень.
3. Полезность для Клевера (макс. 6 баллов): актуальность применения на практике в платформе Клевер и PX4, потенциальный уровень спроса на разработку со стороны других пользователей Клевера.
4. Презентация на финале (макс. 3 балла): качество и зрелищность финальной презентации; полнота освещения проекта; демонстрация; ответы на вопросы жюри.

Призовой фонд

Основное направление конкурса предполагает следующие призы от компании COEX по результатам оценивания жюри на финале:

- I место: $3000.
- II место: $2000.
- III место: $1000.
- IV место: $500.
- V место: $500.

Партнеры конкурса могут поощрить команды по дополнительным критериям, выявленным в результате оценки проектов в ходе финала.

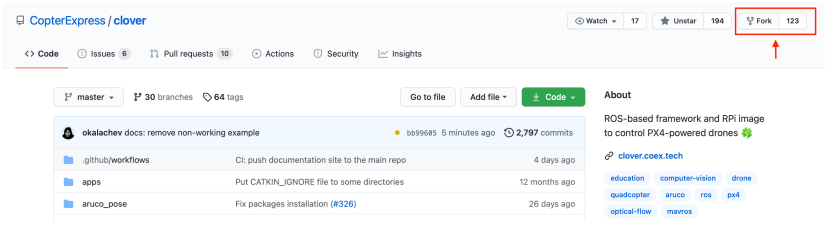
Номинация "кейс компании" предоставляет приз от компании COEX для дальнейшего развития проекта в размере $2500 для команды с лучшим результатом по каждому из кейсов.

Как подать заявку?

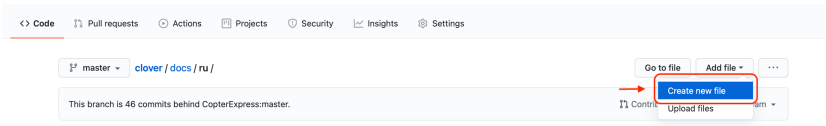
> Для подачи заявки необходимо иметь аккаунт на GitHub.

Подготовьте вашу заявку и пришлите ее в виде Draft Pull Request в репозиторий Клевера.

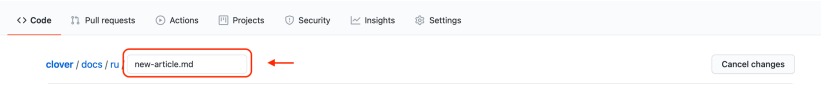
1. Сделайте форк репозитория Клевера:



2. На странице вашего форка зайдите в раздел `docs/ru` и создайте новый файл в формате Markdown:



3. Введите название вашей статьи. Например, `new-article.md`



4. Оформите вашу заявку в соответствии с рекомендуемым шаблоном:

```
# Название проекта

[CopterHack-2022](copterhack2022.md), команда **Название команды**.

## Информация о команде

Состав команды:

(Опишите состав команды: имя и фамилия, контакты (e-mail/имя пользователя

* Александр Соколов, @aleksandrsokolov111, инженер.
* Елена Смирнова, @elenasmirnova111, программист.

## Описание проекта

### Идея проекта

Опишите кратко идею и стадию проекта.

### Планируемые результаты

Опишите как вы видите результат проекта.

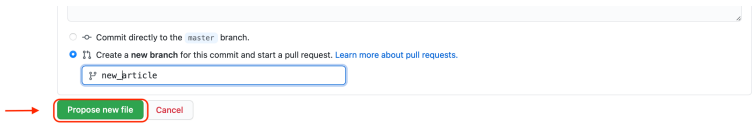
### Использование платформы "Клевер"

Опишите как в вашем проекте будет использоваться платформа "Клевер".

### Дополнительная информация по желанию участников

Например, информация об опыте работы команды над проектами, прикрепить ссы
```

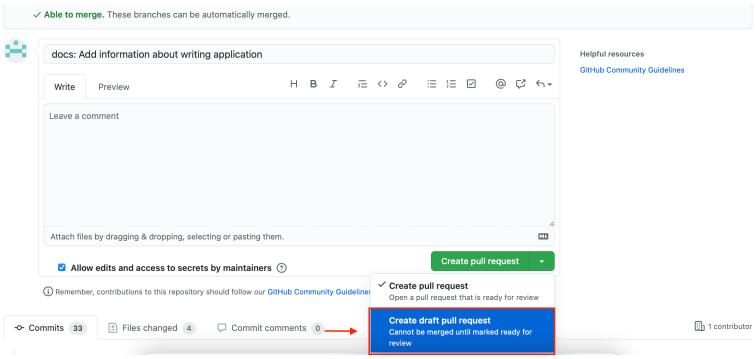
5. Перейдите вниз страницы и создайте новую ветку с названием вашей статьи:



> Не добавляйте ваши изменения непосредственно в ветку `master`, создайте новую ветку.

6. При необходимости поместите дополнительные визуальные материалы в папку `docs/assets` и оформите на них ссылки в вашей статье.

7. Сделайте Draft Pull Request вашей ветки в master Клевера:



8. В комментариях Pull Request вам будет дана обратная связь по заявке. На страничке конкурса в разделе "Проекты участников конкурса" будет опубликована ссылка на вашу заявку в вашем форке.

9. Обратите внимание на блок *Checks*, в графе Documentation должна стоять галочка. Если там стоит крестик, перейдите по ссылке *Details*, чтобы увидеть список проблем с оформлением статьи. При необходимости изменения добавляемых файлов, меняйте их в вашей ветке – изменения будут появляться в Pull Request автоматически. **Не создавайте новый Pull Request для одной и той же заявки**.

10. На протяжении конкурса вы будете работать над этим документом, приближая его к состоянию статьи. В документе будет видна история разработки и ежемесячные апдейты. К финалу конкурса вы сможете опубликовать вашу статью, это и будет результат вашей работы в CopterHack.

Как только ссылка на заявку будет добавлена на эту страничку в раздел "Проекты участников конкурса", ваша команда стала официальным участником CopterHack 2022!

Участники конкурса будут добавлены в Telegram-группу, куда можно отправлять первый апдейт и получить обратную связь от Жюри. Для команд-участников предусмотрена скидка 50% на конструктор программируемого квадрокоптера "Клевер".

> Ограничения по возрасту, образованию и количеству человек в команде отсутствуют.

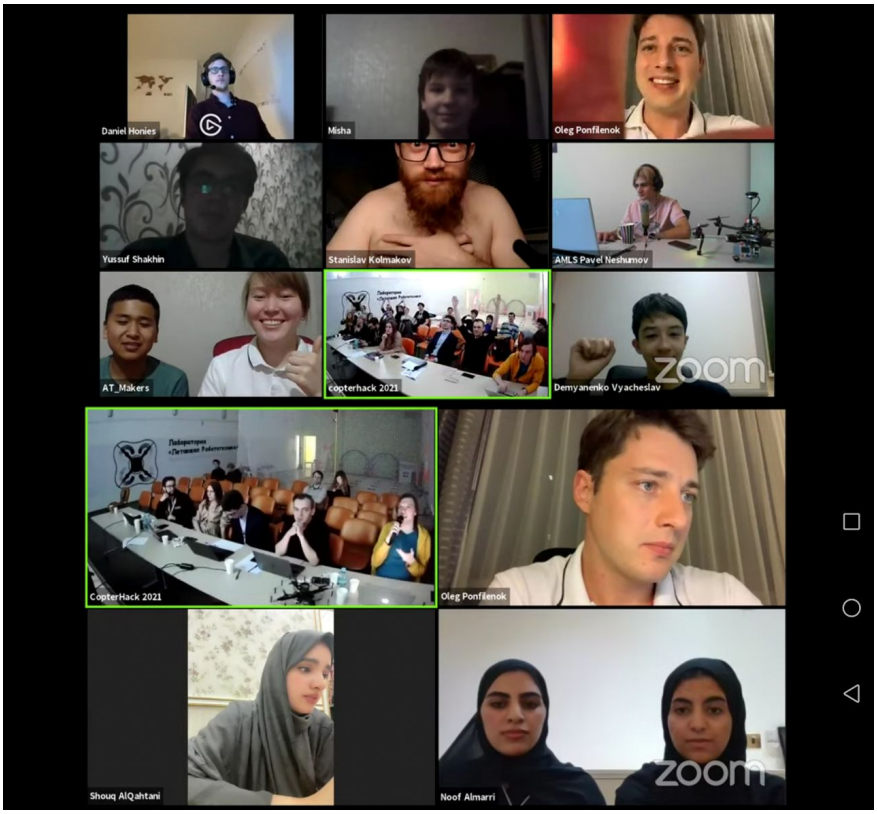
По всем вопросам: CopterHack 2022.

CopterHack 2021

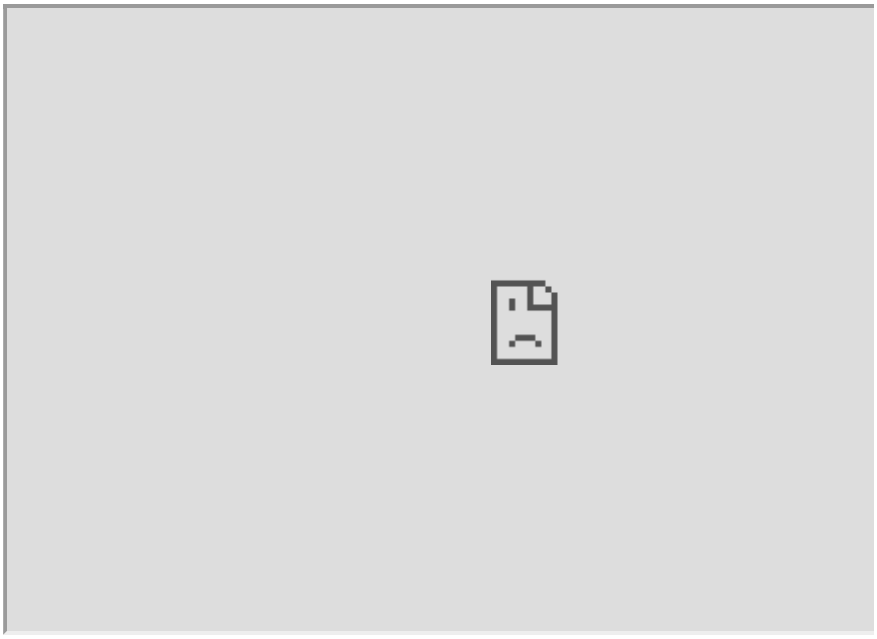
CopterHack 2021 – это командный конкурс по разработке проектов с открытым исходным кодом для платформы квадрокоптера "Клевер". В конкурсе приняло участие 54 команды из 12 стран.

Все информацию о мероприятии смотрите на официальном сайте: https://ru.coex.tech/copterhack.

Полная запись трансляции финала: https://www.youtube.com/watch?v=Z06vxuAHmuE.



Видео



Отчетные статьи команд-участников

Место	Команда	Проект	Балл
1	🇧🇾 FTL	AdvancedClover	18.8
2	🇷🇺 EasyToFly	EasyToFly	18.5
3	🇩🇪 ADDI	3D-printed generative design frame	17.8
4	🇰🇬 AT Makers	Граффити-коптер D-drone	16.7
5	🇶🇦 DroMap	The Indoor Mapping Drone	16.5
6	MINIONS	Дрон для высаживания семян	15.5
7	🇷🇺 Хардатон	Хардатон Квиддич	15.48
8	Atomic Ferrets	Система засечки для дронов	15
9	Drones to fight Corona	Drones to fight Corona	14.6
10	AMLS	Система автоматической посадки	12.8
11	PaD30DЖ	Октокоптер со специфичным расположением пропеллеров	11.6
12	Зауральский Викинг	Программируемый летающий автомобиль	11.4
13	🇨🇦 Bennie and the Jetson TX2	Retail Drone	9.8
14	🇳🇱 Blue Jay Eindhoven	Designing a drone and a path planning algorithm	9.6
15	ProCleVeR	Разработка системы для управления БПЛА с помощью шлема виртуальной реальности	8.5
16	🇷🇺 Quadrotor	Дрон-Агроном	7.7

Смотрите оценки по критериям в полной таблице.

Copter Hack 2019

Хакатон Copter Hack 2019 проходит 11–13 октября в Технополисе "Москва".

Официальный сайт: https://ru.coex.tech/copterhack.

Чат хакатона: https://t.me/CopterHack.

Timepad: https://copterexpress.timepad.ru/event/1017592/.

Информация для участников

Особенности настройки полетного контроллера COEX PIX

При использовании полетного контроллера *COEX Pix* перед калибровкой датчиков вам стоит обратить внимание, что в графе *Autopilot orientation* вы должны выбрать параметр `ROTATION_ROLL_180_YAW_90`. Данную настройку требуется проводить при калибровке каждого из датчиков.



Этот параметр устанавливается для того, чтобы на программном уровне настроить ориентацию вашего *IMU* датчика находящегося на полетном контроллере.

Рекомендуемая версия образа

Для Raspberry Pi версий до 3B+: v0.18

Для Raspberry Pi версии 4: v0.19-alpha.1

Ориентация камеры

На многих дронах камера ориентирована шлейфом вперёд. Это следует отразить в файле `main_camera.launch` в пакете `clever`.

Подробнее см. статью Ориентация камеры.

Полет с использованием Optical Flow

Для включения optical flow установите параметры `optical_flow` и `rangefinder_vl53l1x` в файле `clever.launch` в `true`.

Также необходимо в QGroundControl в параметре `LPE_FUSION` включить галочку `pub agl as lpos down`.

Необходимо также убедиться, что лазерный дальномер корректно установлен и работает (см. конфигурирование дальномера).

Подробнее: Использование Optical Flow.

Использование карты маркеров

Для настройки большой карты маркеров используйте карту с названием `cmit.txt`. Далее используйте инструкцию.

Аккумуляторы

При полетах обязательно использование датчика напряжения ("пищалки"). В случае выхода из строя аккумулятора новый не предоставляется!

Съемка видео

Снимайте **ВСЕ** ваши полеты на видео! В случае поломки дрона на защите сможете показать видео, что будет учтено при оценке.

Проблема с yaw

При полете по маркерам (VPE) в прошивке v1.8.2-clever.7 возможно есть ошибка, которая проявляется в том, что дрон не держит yaw по маркерам. Если у вас есть такая проблема, попробуйте залить более старую прошивку v1.8.2-clever.6, доступную по ссылке https://github.com/CopterExpress/Firmware/releases/tag/v1.8.2-clever.6. Для COEX Pix необходимо скачивать файл `px4fmu-v4_default.px4`.

Проблема с navigate

В образе 0.18 обнаружился баг из-за которого дрон может летать по точкам слишком быстро. Если у вас это происходит, поставьте в файле `~/catkin_ws/src/clever/clever/launch/clever.launch` параметр `nav_from_sp` в значение `false` таким образом:

```
<!-- simplified offboard control -->
<node name="simple_offboard" pkg="clever" type="simple_offboard" output="screen
    <param name="reference_frames/body" value="map"/>
    <param name="reference_frames/base_link" value="map"/>
    <param name="reference_frames/navigate_target" value="map"/>
    <param name="reference_frames/navigate_target" value="map"/>
    <param name="nav_from_sp" value="false"/>
</node>
```

Лекции

Лекция 1: введение – https://www.youtube.com/watch?v=cjtmZNuq7z0.

Лекция 2: настройка полетного контроллера – https://www.youtube.com/watch?v=PJNDYFPZQms.

Лекция 3: архитектура полетного контроллера PX4 – https://www.youtube.com/watch?v=_jl7FImq3jk.

Лекция 4: автономные полеты: https://www.youtube.com/watch?v=ThXiNG1IzvI.

См. также другие видео на канале COEX на YouTube: https://www.youtube.com/channel/UCeCu93sLBkcgbIkIC7Jaauw/featured.

Результаты

Команды-победители:

1. Бульболет – доставка картошки с помощью умной лебедки.
2. Copter don't hurt me – управление дроном с помощью нейроинтерфейса.
3. import torch – active track на нейронках.
4. Автобот – freeze light через бота ВКонтакте.
5. Stardust Crusaders – AR среда симуляции для дронов.

Олимпиада НТИ 2019

Работа с MQTT

MQTT – протокол для обмена сообщениями между различными устройствами. Этот протокол используется для отправки команд дрону на Олимпиаде НТИ 2019. Для отправки сообщения оно публикуется в определенный топик; все подписчики этого топика получают это сообщение.

Подписка на топики

В образе Клевера для Олимпиады НТИ 2019 предустановлена библиотека `paho-mqtt` для Python. Пример работы с этой библиотекой описан ниже:

```
import paho.mqtt.client as mqtt  # Импортирование библиотеки mqtt

# Callback, вызываемый при получении от сервера подтверждения о подключении
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

    # Если подписываться на топик в on_connect, то при обрыве соединения
    # и повторном подключении произойдёт автоматическое переподписание
    client.subscribe("/xxx")

# Callback, вызываемый при появлении сообщения в одном из топиков, на который
# подписан клиент
def on_message(client, userdata, msg):
    # В объекте msg хранится топик, в который пришло сообщение (в поле topic)
    # и само сообщение (в поле payload)
    print(msg.topic, str(msg.payload))

# Инициализация клиента MQTT
client = mqtt.Client()
# Здесь указываются callback'и, вызываемые при подключении и получении сообщени
client.on_connect = on_connect
client.on_message = on_message

# Подключение к MQTT-брокеру. Первый параметр - имя или адрес брокера, второй
# (по умолчанию 1883), третий - максимальное время между сообщениями в секунда
# (по умолчанию 60).
client.connect('192.168.1.199', 1883, 60)

# Метод loop_start создаёт поток, в котором будет производиться опрос сервера
# вызов callback'ов.
client.loop_start()
# Далее продолжается ваша программа
```

Более подробная документация доступна на странице библиотеки в PyPI.

Отправка сообщений

Для отправки сообщений можно использовать метод `publish` клиента:

```
import paho.mqtt.client as mqtt
# Создание подключения - аналогично предыдущему примеру кода
# ...
client.publish(topic='/xxx', payload='connected')
```

Данный код опубликует сообщение `connected` в топик `/xxx`.

Проверка

Для проверки вы можете опубликовать любое сообщение в топик с помощью команды `hbmqtt_pub`:

```
hbmqtt_pub --url mqtt://192.168.1.199:1883 -t /xxx -m 'сообщение'
```

Где `192.168.1.199` – IP-адрес MQTT-брокера, `сообщение` – сообщение для публикации, `/xxx` – необходимый топик для публикации.

Чтобы проверить публикацию сообщений от клиента, воспользуйтесь командой `hbmqtt_sub`:

```
hbmqtt_sub --url mqtt://192.168.1.199:1883 -t /xxx
```

Отправленные в топик `/xxx` сообщения будут показаны в терминале.

Работа с Клевером

Для выполнения команд на Клевере:

- подключитесь в Wi-Fi сети NTI;
- подключитесь к вашему Клеверу по SSH по его IP-адресу (подробнее см. подключение по SSH);

> После подключения к своему дрону по SSH, смените пароль SSH-доступа, чтобы другие участники не смогли несанкционированно подключаться к нему. Для этого используйте команду `passwd`.

Для редактирования файлов на Клевере вы можете использовать консольные редакторы `nano` или `vim`. Также вы можете загружать файлы используя PyCharm или WinSCP.

Для автономного полета используйте API модуля simple_offboard.

> При использовании русских букв в скрипте на Python 2, добавьте в начало программы следующую строку:
>
> ```
> # coding: utf8
> ```

Пример программы, выполняющей взлет, полет в точку в системе координат площадки и посадку на Python:

```
# coding: utf8

import rospy
from clever import srv
from std_srvs.srv import Trigger

rospy.init_node('flight')

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
land = rospy.ServiceProxy('land', Trigger)

# Взлет на 1 метр со скоростью 1 метр в секунду
navigate(x=0, y=0, z=1, speed=1, frame_id='body', auto_arm=True)

# Ждем 5 секунд
rospy.sleep(5)

# Полет на координаты x=3, y=2, z=1 площадки с углом по рысканью 3.14 радиан со
navigate(x=3, y=2, z=1, yaw=3.14, speed=0.5, frame_id='aruco_map')

# Ждем 5 секунд
rospy.sleep(5)

# Посадка
land()
```

Пример взлета на высоту 1 метр из командной строки:

```
rosservice call /navigate "{x: 0.0, y: 0.0, z: 2, yaw: 0.0, yaw_rate: 0.0, spee
```

Для более подробной информации и описания других команд смотрите API simple_offboard и примеры кода.

Работа с светодиодной лентой

В используемой версии Клевера LED-лента подключена напрямую к Raspberry Pi. При включении всех светодиодов ленты на полную мощность возможно повреждение цепей питания микрокомпьютера.

Сигнальный провод ленты подключен к GPIO-пину 18.

Про работу с LED-лентой можно прочитать в соответствующей статье.

Работа с LED-лентой через ROS

В образ Клевера для Олимпиады НТИ включена нода ROS, работающая со светодиодной подсветкой. С её помощью можно управлять светодиодами, не запуская свою программу из-под `sudo`. По умолчанию эта нода выключена, но её можно включить, если в файле `/home/pi/catkin_ws/src/ros_ws281x/launch/clever4.launch` изменить строку

```
    <arg name="enable" default="false"/>
```

на

```
    <arg name="enable" default="true"/>
```

и перезапустить службу `rosled`:

```
sudo systemctl restart rosled
```

Пример работы со светодиодной лентой:

```
import rospy
# Загружаем из ноды LED-ленты описание сервиса SetLeds...
from ros_ws281x.srv import SetLeds
# ...и сообщений LEDState и LEDStateArray. Сообщение LEDState
# содержит номер светодиода и его цвет, LEDStateArray - массив
# сообщений LEDState
from ros_ws281x.msg import LEDState, LEDStateArray
# Для задания цвета используется стандартное сообщение ColorRGBA
from std_msgs.msg import ColorRGBA

# Количество светодиодов в ленте
NUM_LEDS = 60

# Прокси к сервису установки состояния светодиодов
set_leds = rospy.ServiceProxy("/led/set_leds", SetLeds, persistent=True)

# Вспомогательная функция заполнения всей ленты указанным цветом.
# red, green, blue - интенсивность красного, зелёного, синего цвета
# (задаётся числом от 0 до 255)
def fill_strip(red, green, blue):
    # Создаём сообщение для setLeds
    led_msg = LEDStateArray()
    led_msg.leds = []
    # Для каждого светодиода указываем его новое состояние
    for i in range(NUM_LEDS):
        led = LEDState(i, ColorRGBA(red, green, blue, 0))
        # Записываем состояние светодиода в сообщение
        led_msg.leds.append(led)
    # Вызываем сервис. После этого вся лента должна стать указанного цвета
    set_leds(led_msg)

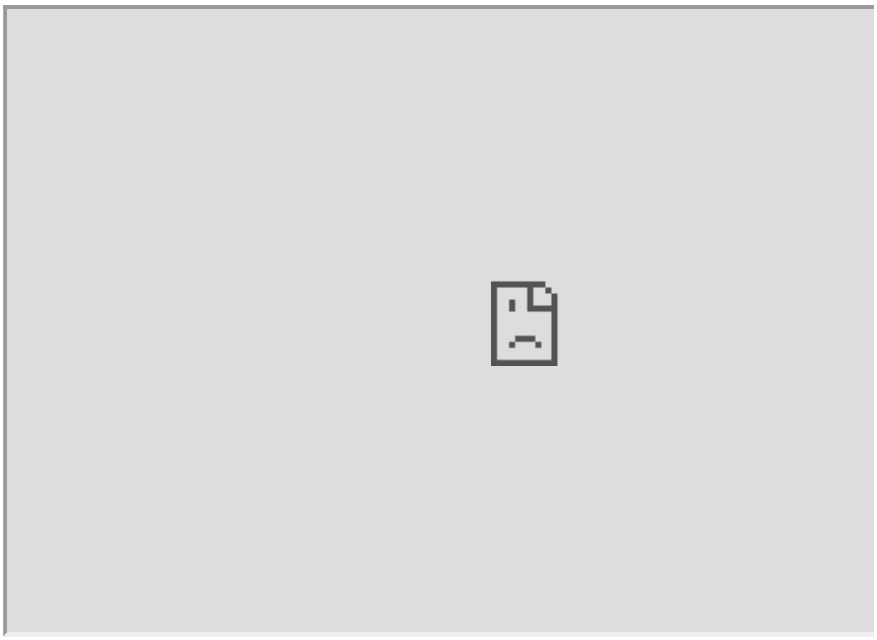

# Заполняем ленту разными цветами
fill_strip(255, 0, 0)
rospy.sleep(2.0)
fill_strip(0, 255, 0)
rospy.sleep(2.0)
fill_strip(0, 0, 255)
rospy.sleep(2.0)
# Выключение ленты
fill_strip(0, 0, 0)
```

Робокросс-2019

В июле 2019 команда Коптер Экспресс в 4-й раз подряд одержала победу на ежегодных испытаниях беспилотных транспортных средств "Робокросс". Испытания проводятся на полигоне ГАЗ под Нижним Новгородом.

Основной задачей испытаний в категории БПЛА было локализовать и уничтожить цель — красный воздушный шар — в автономном режиме.

Видео



Реализация

Команда использовала квадрокоптер на базе рамы F450 и платформу Клевер. Полный итоговый исходный код доступен для изучения на GitHub.

Разработанный пакет `robocross2019` разбит на модули: ROS-нодлет `red_dead_detection` распознает красный шар, `balloon.py` реализует высокоуровневую логику полета коптера.

red_dead_detection

Нодлет `red_dead_detection` обеспечивает обнаружение красного шара на изображении с камеры квадрокоптера, смотрящей вперед (топики `/front_camera/image_raw` и `/front_camera/camera_info`). Используется простейший метод фильтрации изображения по цвету. Затем вычисляется геометрический центр полученных участков и производится компенсация искажений камеры (`cv::undistortPoints`).

Используя известное фокусное расстояние камеры (из топика `camera_info`) вычисляется вектор, направленный в сторону цели. Полученный вектор публикуется в топик `/red_dead_detection/direction`, причем его система координат (`frame_id` связана с расположением передней камеры `front_camera_optical`).

balloon.py

Для полета к шару используется вектор направления `red_dead_detection/direction`, который используется как setpoint по скорости дрона. Угол по рысканью также устанавливается по направлению к шару. Цель считается уничтоженной, когда на протяжении заданного количества кадров общая площадь участка с красными пикселями меньше определенного порога.

Copter Hack 2018

Хакатон Copter Hack 2018 прошел 19–21 октября в Технополисе "Москва".

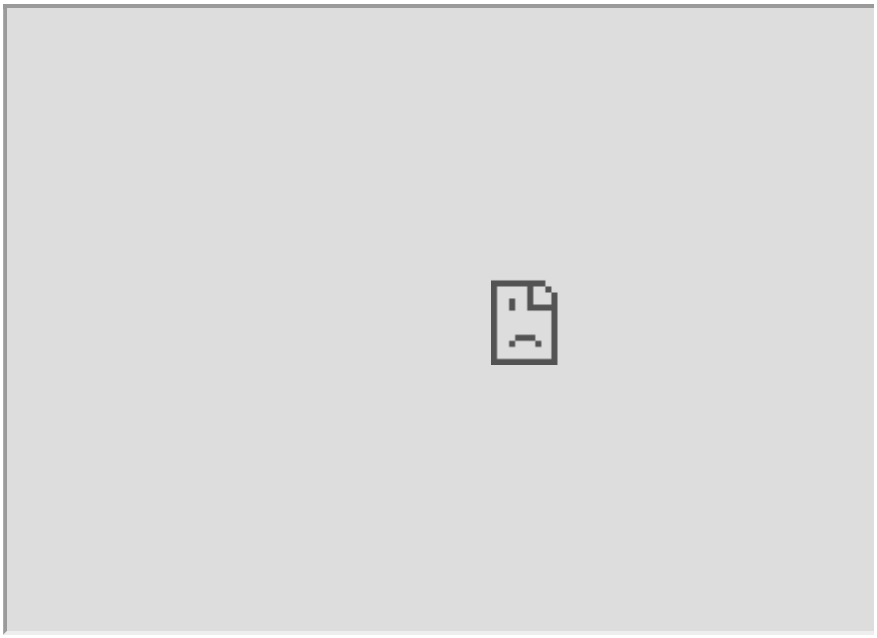


Чат хакатона: https://t.me/CopterHack.

Стрим хакатона: https://www.youtube.com/watch?v=nIo5HSqIt6I.

Фотографии с хакатона: https://drive.google.com/open?id=1ozdXol4rhKwhHbsrnfxrp3CqazBRm-3W.

Видео



Лекции

Лекция 1: сборка – https://www.youtube.com/watch?v=gEs-w7BRPM8.

Лекция 2: настройка – https://www.youtube.com/watch?v=sPqSCCmgdG0.

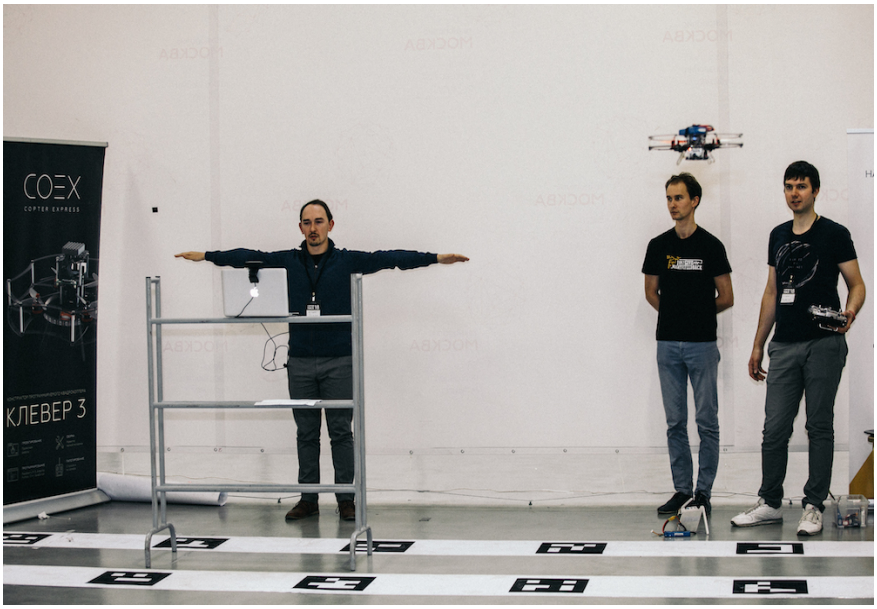
Лекция 3: прошивка PX4 – https://www.youtube.com/watch?v=WFnZAIypgMQ.

Лекция 4: автономные полеты – https://www.youtube.com/watch?v=gD6a7aSEf9M.

Результаты

Команды-победители:

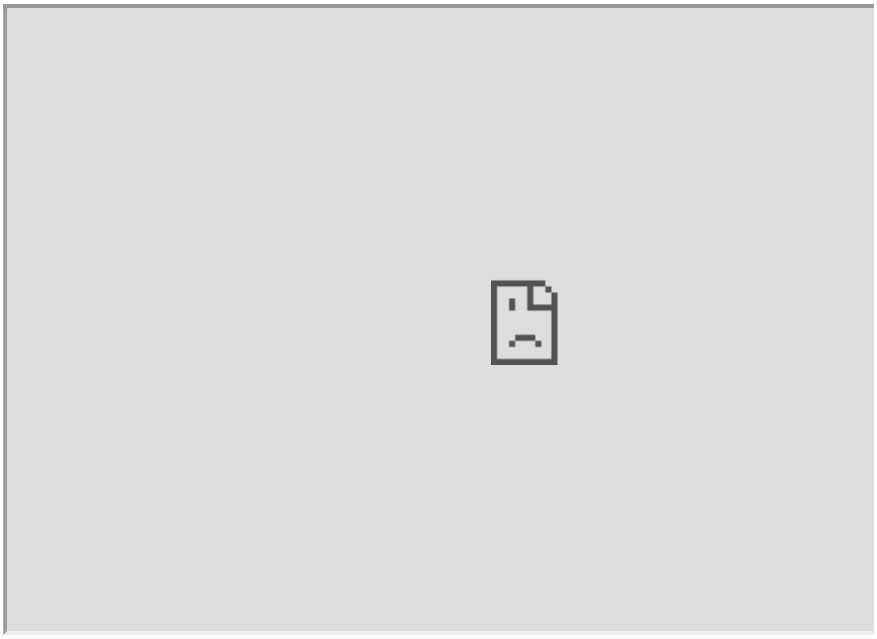
1. Starshine (Москва) — управление дроном с помощью "умной" перчатки.
2. Alcopter (Москва) — управление дроном с помощью жестов и смены поз.
3. Весёлый коптер (Самара) — бот Vkontakte для управления коптером, совместный полет "Жужи" и "Клевера 3".
4. International Post (Новосибирск) — автоматический разброс листовок с дрона.
5. ЛАМАР (Екатеринбург) — станция автоматической смены аккумулятора квадрокоптера.


COEX
COPTER EXPRESS
КЛЕВЕР 3

Copter Hack 2017

28–30 июля 2017 "Коптер Экспресс" провел хакатон "Copter Hack 2017", на котором необходимо было запрограммировать "Клевер" на автономный танец-полет под случайную музыку.

Победителем стала команда "Ящеры".



Запись видеолекций – https://copterexpress.timepad.ru/event/510375/.

Модули

- Навигация в поле маркеров и упрощенное управление коптером: https://github.com/CopterExpress/marker_navigator (установлено на флешке)
- ROS-модуль для взаимодействия с сервером воспроизведения музыки https://github.com/CopterExpress/copter_hack_music (установлено на флешке)
- Исходник сервера воспроизведения музыки https://github.com/CopterExpress/copter_hack_music_server

Просмотр видео с камеры

Выполнить на Raspberry:

```
rosrun web_video_server web_video_server
```

Открыть в браузере страницу `http://<ip raspberry>:8080`.

Внимание: раздача видеострима сильно снижает производительность распознавания маркеров для полета.

SSID Wi-Fi

Чтобы изменить SSID раздаваемого Wi-Fi необходимо любым способом изменить параметр ssid в файле `/etc/hostapd/hostapd.conf`.

Список распознанных маркеров

```
rostopic echo /marker_data
```

Полезные статьи

- Настройка коптера
- Полетные режимы
- Пакет MAVRos
- Неплохая вводная статья: https://habrahabr.ru/post/227425/
- Сигналы, применяющиеся в дронах: https://geektimes.ru/post/258186/
- Хорошая статья про ПИДы: https://habrahabr.ru/company/technoworks/blog/216437/
- Запись видеолекций: https://copterexpress.timepad.ru/event/510375/
- Aubio, библиотека для анализа звука (музыки)
- Пакеты для работы с музыкой для Python: https://wiki.python.org/moin/PythonInMusic

Конкурс на лучшее образовательное видео по сборке и настройке

Требования:

- видео содержит весь процесс сборки и настройки конструктора Клевер 4.2: от открытия коробки с компонентами до летающего Коптера в режиме Position по ArUco-маркерам;
- видео загружено на YouTube и находится в открытом доступе;
- видео содержит озвучивание и субтитры на русском языке;
- видео длится от 6 до 60 минут.

Даты проведения конкурса: 12 февраля – 13 декабря 2021.

Призы

- 🥇 1 место: $500.
- 🥈 2 место: $300.
- 🥉 3 место: $200.

Результаты

Место	Участник	Ссылка на видео
1	🇷🇺 Филипп Баталин	https://www.youtube.com/watch?v=f0rpdulOSEk
2	🇮🇹 Sara Pettinari	https://www.youtube.com/watch?v=PxxfyVH6RRA
3	🇲🇾 Kai Feng Chew	https://www.youtube.com/watch?v=skgSwFle6Ms
3	🇰🇿 Никита Лобанов	https://www.youtube.com/watch?v=93b1epEM3SQ

Образовательные конкурсы

1. Конкурс на лучшую образовательную лекцию

Компания «Коптер Экспресс» объявляет конкурс на лучшую образовательную лекцию с использованием конструктора программируемого квадрокоптера «COEX Клевер 4».

Основной целью конкурса является популяризация летающей робототехники и развитие сообщества данного направления.

Требования к лекции

- Тематика лекции - открытая, на выбор участника (пример: сборка, настройка, программирование, лекция интегрированная в школьную программу и т.п.).
- Основной инструмент лекции - «Конструктор программируемого квадрокоптера «COEX Клевер 4» и/или «Симуляционная среда программируемого квадрокоптера Клевер».

  > *Версия «COEX Клевер» не ранее 4 версии. «Симуляционная среда программируемого квадрокоптера Клевер» - не ранее версии 1.0.

- Видео загружено на YouTube или иную общедоступную платформу, и находится в открытом доступе для любых пользователей;
- Язык лекции - на выбор участника. Если язык лекции не русский/ английский - наличие субтитров к видео на английском языке.
- Продолжительность лекции - от 15 мин. до 3 часов.
- Лекция может являться частью курса или серии лекций, но должна раскрывать заданную участником тематику и быть завершенной.

Требования к участникам

- Участник должен быть автором лекции.
- Техническую поддержку при записи лекции могут оказывать сторонние лица.
- Статус участника - без ограничений (школьник, студент, представитель общеобразовательного учреждения, представитель ДПО, представитель ВПО, представитель СПО, представитель отрасли, любитель).

Подача заявки

Прием заявок осуществляется через Google Форму.

Дедлайн подачи заявок: 30 ноября 2022 года.

Призы

По итогам представленной заявки жюри определяет победителей конкурса. При определении победителей учитываются качество видео, содержание и показатели вовлечения аудитории.

- 1 место: $500.
- 2 место: $400.
- 3 место: $300.
- 4 место: $200.
- 5 место: $100.

2. Конкурс на лучший школьный урок

Компания «Коптер Экспресс» объявляет конкурс на лучший школьный урок с использованием конструктора программируемого квадрокоптера «COEX Клевер 4».

Основной целью конкурса является популяризация летающей робототехники и развитие сообщества данного направления.

Требования к уроку

- Основной инструмент урока - «Конструктор программируемого квадрокоптера «COEX Клевер 4».
  > *Версия «COEX Клевер» не ранее 4 версии.
- Интеграция квадрокоптера в любую из общеобразовательных дисциплин (физика, математика, информатика, урок технологии и т.д.).
- Практическое использование основного инструмента на уроке.
- Класс - без ограничений (начальная, основная школа).
- Продолжительность урока - 30-45 минут.
- Формат урока - оффлайн.
- Видео урока снято в классе общеобразовательного учреждения.

Требования к участникам

- Участник должен быть автором урока.
- Участник должен являться преподавателем общеобразовательного учреждения.

Подача заявки

Прием заявок осуществляется через Google Форму.

Дедлайн подачи заявок: 30 ноября 2022 года.

Призы

По итогам представленной заявки жюри определяет победителей конкурса. При определении победителей учитываются качество видео и материала.

- 1 место: $500.
- 2 место: $400.
- 3 место: $300.
- 4 место: $200.
- 5 место: $100.

3. Конкурс на лучший онлайн-курс

Компания «Коптер Экспресс» объявляет конкурс на лучший онлайн-курс с использованием конструктора программируемого квадрокоптера «COEX Клевер 4».

Основной целью конкурса является популяризация летающей робототехники и развитие сообщества данного направления.

Оценка курса производится по заявленному на конкурс отдельному, общедоступному уроку.

Требования к курсу

- Направление курса - «Летающая робототехника».
- Основной инструмент курса - «Конструктор программируемого квадрокоптера «COEX Клевер 4» и/или «Симуляционная среда программируемого квадрокоптера Клевер».

  > *Версия «COEX Клевер» не ранее 4 версии. «Симуляционная среда программируемого квадрокоптера Клевер» - не ранее версии 1.0.

- Курс расположен на общедоступной платформе (Stepik, Coursera и т.п.).
- Доступ к курсу может быть как платный, так и бесплатный, на конкурс принимается один бесплатный и общедоступный урок заявленного курса.
- Заявленный на конкурс урок должен быть в открытом доступе.
- Язык курса - на выбор участника. Если язык урока не русский/английский - наличие субтитров к видео (в случае наличия видео) на английском языке.
- Продолжительность курса и заявленного урока - не ограничена.
- Участники вправе подать на конкурс курс разработанный ранее, и применявшийся в системе образования, с сохранением всех вышеперечисленных требований.

Требования к участникам

- Участник должен быть автором курса.
- Техническую поддержку при подготовке курса могут оказывать сторонние лица.

- Статус участника - без ограничений (школьник, студент, представитель Общеобразовательного учреждения, представитель ДПО, представитель ВПО, представитель СПО, представитель отрасли, любитель).

Подача заявки

Прием заявок осуществляется через Google Форму.

Дедлайн подачи заявок: 30 ноября 2022 года

Призы

По итогам представленной заявки жюри определяет победителей конкурса. Оценка производится по заявленному на конкурс уроку, при определении победителей учитываются качество материала (формат подачи материала, общий объем и содержание курса).

- 1 место: $1000.
- 2 место: $800.
- 3 место: $600.
- 4 место: $400.
- 5 место: $200.

Проекты на базе Клевера

Конструктор "Клевер" широко используется в для проектной деятельности. В этом разделе приведены проектные идеи, а также статьи пользователей, рассказывающие о реализованных проектах.

Примеры

Список возможных проектов для стажировок и практических работ в вузах и колледжах.

Проект/Идея	Примечания
Шоу дронов	Решение от COEX (разработка началась по программе стажировок). Проект продолжает разрабатываться.
Интеграция позиционирования по Vive Tracker	Наработки.
Интеграция системы позиционирования Pozyx	Наработки.
Интеграция системы позиционирования типа Motion Capture	
Разработка и интеграция дешевого Motion Capture	
Интеграция блочного программирования (Scratch, Blockly)	Решение от COEX.
Улучшение алгоритма визуального позиционирования (optical flow): трекинг по фичам, расчет угла по yaw	
Симулятор Клевера	Решение от COEX (на основе Gazebo). Проект продолжает разрабатываться.
Облачная платформа для симулятора Клевера	По аналогии или на основе ROS Development Studio.
Автоматический захват грузов Клевером	Впервые было реализовано на Олимпиаде НТИ 2019.
Распознавание и ловля летящих объектов (шариков)	
Веб-инструмент для калибровки камеры	
Инструмент для редактирования карт маркеров	Решение с программы стажировок
Точный полет по линии	Решение с программы стажировок
Двойные, тройные, четверные флипы	Исследование стратегий и конфигураций. Двойные флипы на Клевере. "A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips".
Граффити-дрон	Пример реализации на другой платформе.
Коптер-официант	
Автоматическая зарядная станция для Клевера	

Проект/Идея	Примечания
Полет за другим дроном / перехват другого дрона	
Рама Клевера 4 на 3D-принтере	
Измерения уровня загрязнения воздуха / газоанализатор на дроне	
Полет двух коптеров в жесткой связке	
Автоматический подсчёт автомобилей на трассах	
Программное определение падений в PX4	
Начало полета в броске	Throw Mode в ArduPilot.
Полет коптера на точку на изображении с камеры, направленной вертикально вниз	
Внедрение лидара (RPLIDAR) в Клевер	
Зарядная станция для коптера на солнечном концентраторе	
Стенд для настройки коэффициентов PID на Клевере	

Вышеперечисленные и другие проекты вы также можете реализовать в рамках конкурса проектов Copter Hack. Мы приглашаем команды для реализации проектов и в других форматах.

Система радио-навигации

CopterHack-2023, команда **C305**.

Информация о команде

Мы команда студентов, представляющая Центр Проектной Деятельности Дальневосточного федерального университета (C305).

Состав команды:

- Антонов Георгий, @SonSobaki, инженер.
- Филимонов Сергей, @Lukerrr, программист.
- Смадыч Никита, @Sm_nikita, инженер-программист.
- Максим Харченко, @milian_c305, программист.

Описание проекта

Идея проекта

Проект направлен на разработку системы позиционирования внутри помещений для квадрокоптеров с использованием широкополосных передатчиков DWM1000.

Разрабатываемая система позиционирования использует передатчики DWM1000, которые обеспечивают широкополосную связь на радиочастотах. Она предназначена для обеспечения точного и надежного позиционирования квадрокоптеров внутри зданий, где оптические системы могут оказаться ограниченными или неэффективными.

Одной из ключевых особенностей этой системы является ее способность обеспечивать высокую точность позиционирования и дальность действия. Широкополосные передатчики позволяют достичь высокой разрешающей способности и низкой задержки передачи данных, что особенно важно в случае быстрого и точного позиционирования квадрокоптеров. Благодаря использованию радиочастотных сигналов, система не подвержена помехам от окружающей среды, таких как освещение или преграды. Это обеспечивает стабильную работу системы в различных условиях и позволяет использовать ее в помещениях с ограниченной видимостью или сложным рельефом.

В целом, разрабатываемая система предлагает более точное, надежное и экономичное решение по сравнению с оптическими системами.

Ключевые особенности

- Точность системы позиционирования +-0.1м.
- Поддержка indoor и outdoor навигации.
- Лёгкая масштабируемость системы.

- Лёгкое развёртывание и лёгкая настройка системы.

Презентационный ролик



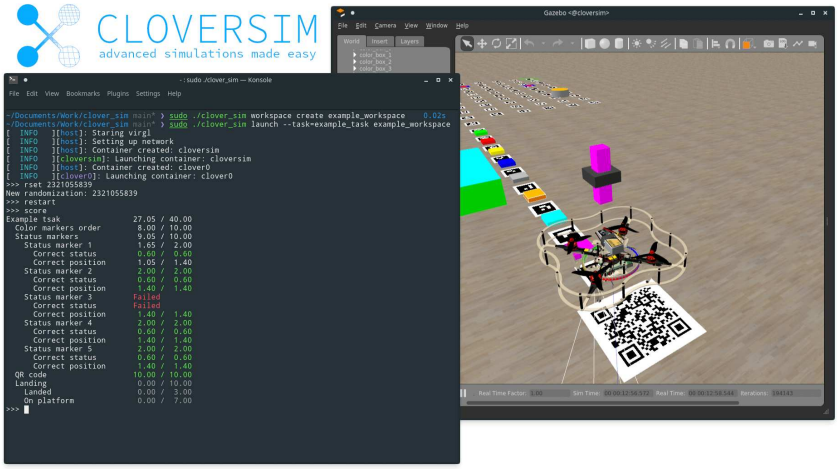
Документация к проекту

- Работа с UWB модулями
- Запуск ROS пакета позиционирования
- Настройка системы позиционирования

Ресурсы проекта

- Исходный код прошивок UWB модулей
- Исходный код ROS пакета позиционирования
- Модели корпуса UWB модулей

Advanced Clover Simulator



CopterHack-2022, команда **FTL**.

Ссылка на проект.

Информация о команде

```
#include "interestingCommandDescription.h"
```

Состав команды:

- Максим Романовский, @maximmasterr, капитан, программист.
- Михаил Кулик, @mkulik05, программист.

Страна: Беларусь.

Описание проекта

Разработка симулятора для Clover с возможностью создания workspace (окружения, позволяющие легко управлять и транспортировать файлы симулятора такие как: проекты, конфигурации, установленные пакеты).

- Простой, быстрый и производительный способ запуска симулятора.
- Инструментарий для создания заданий (как на IOR2020).
- Изоляция Gazebo и чекера заданий от Workspace пользователя и clover.
- Worker для проверки задач со степика.
- Возможность запуска несколько коптеров каждый из которых имеет свой контейнер.
- Полная симуляция raspberry с помощью qemu (включая GPIO, USB и камеру).

Использование платформы "Клевер"

Данный проект является набором инструментов для запуска симуляторов Клевера.

Первые пуски

Для начала мы решили разработать симулятор имеющий такой же функционал как и виртуальная машина

В качестве инструменты для запуска контейнеров мы используем `systemd-nspawn` (read more), так как он запускает systemd init и позволяет пользоваться `systemctl` , так как: не эмулируется ядро линукс, отсутствует ограничение на процессор и память, производительность такого метода выше чем у виртуальной машины.

С графикой посложнее, в виртуальной машине используется графика от VMware, однако в случае если мы не эмулируем ядро мы могли бы пробросить видеокарту. В отличии от виртуальной машины видеокарта останется доступна в хост машине, однако данный метод требует что бы внутри были установлены такие же драйвера как и в хосте(привет Nvidia). Поэтому мы используем `virgl` (read more), а точнее virgl_test_server. VirGL представляет собой очень производительный способ проброса OpenGL (и Vulkan). Из-за бага в mesa требуется чтобы стояла mesa 21, для этого мы добавляем ppa kisak-mesa. Работает это примерно так: virgl_test_server создаёт unix-сокет, он пробрасывается в контейнер, mesa в контейнере отправляет запросы на рендер в этот сокет, VirGL принимает их и транслирует в видеокарту, результат отображается на экране. Такой способ позволяет получить более 120fps в Gazebo.

Для реализации механизма workspace, мы используем overlayfs, она позволяет создать фс в которой файлы модифицируются и читаются в "верхней" файловой системе, а за тем по порядку в нижних. Таким образом у нас есть директория base которая хранит внутри симулятор, ubuntu, clover. A workspace являются папки поверх base которые хранят только изменения что позволяет сильно экономить место. Workspace можно быстро создавать и удалять, экспортировать и импортировать.

Для сборки base мы используем GitHub Actions, здесь находится репозиторий со скриптами, из интересного можно отметить то что требуется прописать в `/etc/resolv.conf` dns-серверы, прописать нормальный hostname в `/etc/hostname` (иначе debootstrap скопирует его с runner-a GitHub, и оно будет странным) и прописать в `/etc/hosts` строку для резолва этого имени на `127.0.0.1` иначе ROS не запустится.

Для того чтобы было всё красиво и удобно мы создали инструмент на Go, который занимается управлением workspace-ами и запуском симулятора.

При попытке запуска симулятора происходило полное поглощение оперативной памяти и swap, а это 8+32 гигабайта. В чём же дело? А в том, что rosout для того чтобы проверить не упрётся ли оно в лимит файловых

дескрипторов... аллоцирует их все :/ Гениальное решение. Решается добавление лимитов в `/etc/security/limits.d/30-nofilelimit.conf`. Более подробно здесь.

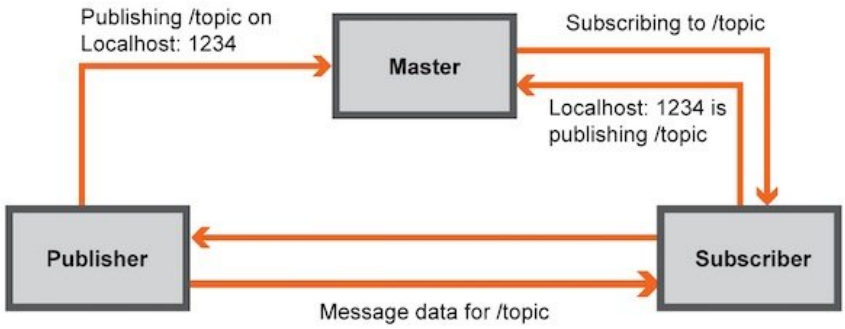
Сеть, раздельные контейнеры, внутренности systemd и ROS

Для того чтобы обеспечить лучшую переносимость workspace-ов и большую безопасность логично разделить симулятор на две части: Gazebo и clover. Часть Gazebo запускает внутри себя Gazebo и проверяющий скрипт, а часть clover запускает эмулятор PX4, сервисы clover и пользовательские программы. Для упрощения разработки и обеспечения большой безопасности в каждом контейнере должен запускаться свой roscore.

И чтобы это всё могло общаться между собой необходимо чтобы симулятор создавал свою виртуальную сеть. Имеется очень хорошая статья на эту тему, она рассказывает про тот как создаётся виртуальная сеть в докере. В отличие от докера systemd имеет возможность автоматической настройки network namespace если передать ему bridge что достаточно упрощает нашу задачу. Итак что происходит при запуске симулятора мы создаём bridge под названием `cloversim` и задаём ему IP `192.168.77.1`, такой IP будет у хост системы, команды которые это делают. Так же для того чтобы всё правильно работало необходимо выделять и присваивать IP адреса контейнерам. Выделение реализованно достаточно просто: если при запуске контейнера был указан рекомендуемый IP то начинаем поиск свободного с него, иначе с IP `192.168.77.10`, поиск происходит простой инкрементацией последнего числа, реализация. Для того чтобы настроить сеть в контейнере внутри него выполняется команда `ip addr add <ip>/24 dev host0 && ip link set host0 up && ip route add default via 192.168.77.1`, она присваивает IP, поднимает подключение и говорит что все пакеты следует слать через него. Маршрутизацией пакетов между контейнером и хостом занимается bridge, а вот для того чтобы контейнеры могли получить доступ к интернету нужно поднять небольшой NAT. В статье упомянутой раньше используется iptables, но вместо него теперь рекомендуется nftables. Работает это примерно так: создать таблицу роутинга, добавить цепочку правил к таблице, добавить правило которое применяет nat. Реализация.

О запуске внутри systemd-nspawn. Можно подумать что запуск внутри контейнера происходит как то так: `systemd-nspawn exec ls`, однако это в корни не так. Запускать команды нужно как-то так `systemd-run --machine=clover0 -t /bin/bash -c "ls"`. Что же такое systemd-run? Это инструмент systemd который создаёт временный systemd-сервис и запускает его. Кстати для того чтобы контролировать systemd-сервисы внутри контейнера нужно использовать `systemctl --machine=clover0 start SSHd.service`. И так для любых команд для работы с systemd. Как бы это не казалось странным всё дело в упрощении, для того чтобы не внедрять дополнительный функционал по запуску команд в systemd-nspawn, проще устанавливать соединение с systemd внутри контейнера и просить его

создать и запустить сервис. У такого подхода есть свои плюсы: мы можем запускать команды после того как определённые сервисы запустились, мы получаем мощь сервисных файлов systemd для запуска команд, не нужно заново реализовывать логирование. Что касается создания соединения с systemd в контейнере, на самом деле systemctl (и др.) умеют подключаться не только к контейнерам но и к удалённым SSH машинам, так же это необходимо для того чтобы systemctl (и др.) могли подключаться не к системному systemd а к юзерскому (такая возможность также существует). Так что как бы не казалось монолитной структура systemd изначальна, в действительности она очень красиво реализована.



Далее для того чтобы контейнеру с clover было доступна камера и LED-лента, которой управляет симулятор. За работу камеры отвечают топики `/main_camera/image_raw` и `/main_camera/camera_info`, для работы LED нужен топик `/led/state` и сервис `/led/set_leds`. Конечно можно просто написать сервер который будет получать ROS-сообщения и отправлять их клиенту который будет публиковать их в другом клиенте. Но такой подход не самый производительный, по этому мы реализовали скрипт на Python который получает информацию из одного ROS-мастера и публикует его в другой, и в результате ноды разных мастеров общаются между собой напрямую. ROS мастер выполняет функцию связывающего который сообщает на каких адресах находятся какие ноды и какие топики они публикуют/подписаны, так же он сообщает адрес по которому доступна нода для сервисов, более подробно. Соответственно, для того чтобы сервисы и топики стали видны необходимо передать информацию из одного мастера в другой. Для этого можно обращаться напрямую к ROS Master API, API ROS представляет собой набор функций вызываемых с помощью xmlrpc. Для того чтобы "пробросить топик" из мастера А в мастер Б, мы подписываемся на нужный топик, получаем информацию о том кто его публикует, отправляем эту информацию в мастер В, топик становится доступным в мастере В. С сервисами ситуация чуть проще, при вызове сервиса клиент каждый раз запрашивает мастер про адрес по которому нужно вызывать сервис, так как сервис может представлять только одна нода то мы просто с определённой периодичностью запрашиваем информацию о сервисе у мастера А и отправляем её в мастер В. Вот так это всё реализовано: тык.

Launch-файл не сильно поменялся по сравнению с прошлой версией. Во первых он был разнесён на две части: первая часть это - `Gazebo instance` и `Clover model`. Во вторую часть попали `PX4 instance` и `Clover services`. Так же в первую часть был добавлен запуск ROS-прокси. Во второй части была задана переменная окружения которая говорила PX4 что симулятор находиться в первом контейнере и gcs_bridge был переключён на UDP чтобы QGroundControl мог автоматически подключаться к контейнерам. Итоговые файлы: контейнер с симулятор, контейнер с clover-ом.

Видео с демонстрацией работы после описанного выше.

Задачи, Catkin и Gazebo

Нам было необходимо реализовать какой-то метод для управления симулятором. Первой идеей было добавление подкоманд в clover_sim, тогда перезапуск симулятора выглядел бы примерно так: `sudo ./clover_sim simulator restart`. У такого подхода есть две проблемы: во-первых каждый раз надо набирать `sudo ./clover_sim`, во вторых нужно как-то(например с помощью unix-сокетов) реализовать обмен данными между запущенной командой и `sudo ./clover_sim launch` который контролирует всё. Поэтому мы решили реализовать консоль внутри `sudo ./clover_sim launch` которая позволит управлять симулятором. Эта статья очень помогла сделать консоль удобной. Реализация внутри clover_sim. Выглядит это так:

```
~/Documents/Work/clover_sim docs ❯ sudo ./clover_sim launch example_workspace
[sudo] password for max:
[  INFO   ][host]: Staring virgl
[  INFO   ][host]: Setting up network
[  INFO   ][host]: Container created: cloversim
[  INFO   ][cloversim]: Launching container: cloversim
[  INFO   ][host]: Container created: clover0
[  INFO   ][clover0]: Launching container: clover0
>>> score
Fly rectangle task      10.00 / 50.00
  Point A                0.00 / 10.00
  Point B                0.00 / 10.00
  Point C                0.00 / 10.00
  Point D                0.00 / 10.00
  Land                  10.00 / 10.00
>>> rnew
New randomization: 88337090
>>> help

Available commands:
  help     - show this help
  exit     - shutdowns cloversim
  start    - start simulator (gazebo, clover, etc.)
  stop     - stop simulator (gazebo, clover, etc.)
  restart  - restart the simulator (gazebo, clover, etc.)
  rget     - get current randomzation
  rset     - set randomzation to value from second argument
  rnew     - generate new random randomzation
  score    - show current task scoring
  run      - run user program (runs /home/clover/run.sh inside clover0 container)
>>> rget
Randomization: 88337090
>>> restart
>>> exit
```

Далее для упрощения распространения мы решили перенести ROS-пакет с нашим симулятором из base_fs в отдельный слой overlay, мы собираем этот слой с помощью systemd-nspawn без boot флага(работает почти так же как простой chroot). Для того чтобы не собирать слой с нашим пакетом каждый раз мы сверяем package.xml в исходниках нашего пакета с package.xml в слое и если они не совпадают мы пересобираем слой. Также не стоит забывать про то что при сборке нужно подгружать переменные окружения.

Задания

Задания представляют собой ROS-пакеты которые способны создавать миры Gazebo и проверять пользовательские решения, сами задания загружаются из папки tasks при launch-е. Так же как и в слое с cloversim мы храним задание в отдельном overlay слое, и при запуске cloversim сверяет package.xml в задании с package.xml в слое и если они не совпадают мы пересобираем слой. Следующая проблема это как нам вызвать что нибудь в задаче (например генерацию мира) тут на помощь приходит "особенность" catkin которая позволяет установить питоновский пакет в название независимое от название ROS-пакета. И используя эту особенность каждое задание вне зависимости от названия доступен так: `from cloversim_task.world import WORLD`. Файл который ложит пакет под одним и тем же именем: тык. Кроме того нам необходимо запускать генерацию мира из roslaunch перед тем как запускать ноды типа Gazebo или clover. На помощь приходит фича roslaunch которая позволяет положить в ROS-параметр результат команды, документация про это. Вот так это выглядит в roslaunch: тык. Также для того чтобы не стартовать симулятор раньше времени у нас есть механизм синхронизации между контейнерами: roslaunch в контейнере с clover запускает TCP сервер и ждёт коннекта и выходит затем, в свою очередь контейнер с симулятором после генерации мира подключается к контейнеру с clover вследствие чего команды roslaunch в обоих контейнерах продолжает выполнение и запускает ноды.

Переходим к тому как задание работает с миром. Во-первых генерация, миры Gazebo описаны с помощью SDFormat основанном на XML, и для генерации миров мы просто написали несколько классов (куб, цилиндр, модель из файла) которые генерирует мир в SDFormat и сохраняют его. Для того чтобы можно было использовать в качестве текстуры не только текст но и изображения мы реализуем функцию которая генерирует ogre-скрипты которые подаются в Gazebo и рендерят картинку. Так же нам необходимо иметь возможность создавать модели с разными текстурами на разных гранях, для того чтобы это сделать например для куба нам нужно создавать 6 тонких кубов и размещать их там где должны находится грани. Другой способ сделать разные текстуры на разных поверхностях это использование COLLADA-файлов, но мы отказались от этой идеи так как кроме генератора SDFormat нам пришлось бы делать генератор COLLADA что сильно усложнило бы библиотеку. Далее для того чтобы миры не были каждый раз одинаковыми необходимо добавить возможность рандомизации, на помощь приходит псевдо-рандом который позволяет генерировать псевдо-случайные значения используя seed, один seed один и тот же результат рандомизации. Использование seed так же позволяет легко переносить различные рандомизации и тестировать их по порядку. Теперь переходим к тому как проверять задания Gazebo предоставляет ROS-сервисы которые позволяют получить текущее положение вещей. Так что для определения позиции коптера нам нужно просто вызвать get_model_state, а для проверки на arm коптера можно проверять скорость в `get_joint_state`.

Видео с демонстрацией

«Copter For Space» (C4S) – принимаем данные со спутника на Клевер

CopterHack-2022, **команда Space clowns**.

Информация о команде

Состав команды:

- Илья Холодилов, https://t.me/ilyazxz, TeamLead, программист-инженер.
- Вячеслав Демьяненко, https://t.me/SlavikYD, инженер-программист.
- Андрей Пивоваров, https://t.me/DedAndrew, программист.
- Ярослав Мухатдинов, https://t.me/Euenot2, инженер-конструктор.
- Тимур Малышкин, https://t.me/Timur_Malyshkin, аналитик космических снимков и геопространственных данных (декодирование и постобработка данных), TechWriter.

Видео о проекте



Описание проекта

Создание инженерного конструктора станции приёма данных L-диапазона с метеорологических спутников на базе «Clover-4.2».

Наша цель - сделать конструктор доступнее, чтобы вовлечь как можно больше школьников в прикладные исследования в областях летающей робототехники, программирования, а также метеорологии.

Идея проекта

Аперитив

Однажды пятеро джентльменов, любуясь туманными горами Кабардино-Балкарии, решили проверить дерзкую, не имеющую аналогов в мире идею. А что если вместо программируемой руки-манипулятора или тяжёлой тросово-опорной конструкции, для переноса облучателя над фокальной плоскостью комплекса приёма космической информации использовать дрон или же сеть дронов?

Прежде чем начать повествование об инновационной разработке, Вы, дорогой читатель, должны понимать основной принцип работы станций приёма космических данных. Предлагаем Вам ознакомиться с обучающими вебинарами (https://youtu.be/adclrJgpJWg), где на примере комплекса «Link To Space», разработанного инженерной компанией «LoReTT», описываются основные аспекты приёма, декодирования, обработки, постобработки, а также анализа космических снимков.

Шоу начинается

В далёком 1963-м году, в 15-ти километрах от маленького пуэрто-риканского городка Аресибо, была открыта одноимённая радиообсерватория. В 1974 году именно оттуда было отправлено легендарное «Послание Аресибо», адресованное внеземным цивилизациям. Помимо этого, данная радиообсерватория принимала активное участие в изучении дальнего космоса. В 2020-ом году, из-за усталости конструкции, лопнул один из тросов, удерживающих облучатель, что повлекло за собой разрушение не только облучателя, но и, частично, «зеркала» приёмного комплекса. На восстановление комплекса уйдут годы и миллионы долларов, поэтому Корнеллский университет и Научный фонд США – главные операторы обсерватории, - сочли нецелесообразным восстановление объекта. Кто знает, может быть, США могли бы совершить ещё не одно открытие в области изучения космоса.





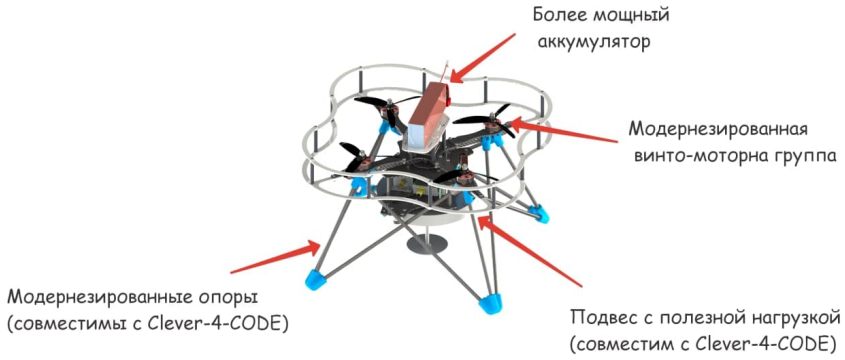
Но давайте немного отыграем ситуацию назад. А что если вместо программируемой руки-манипулятора или тяжёлой тросово-опорной конструкции, для переноса облучателя над фокальной плоскостью комплекса приёма космической информации использовать дрон или же сеть дронов? А если это возможно, то почему бы не создать инженерный конструктор для школьных технологических кружков?

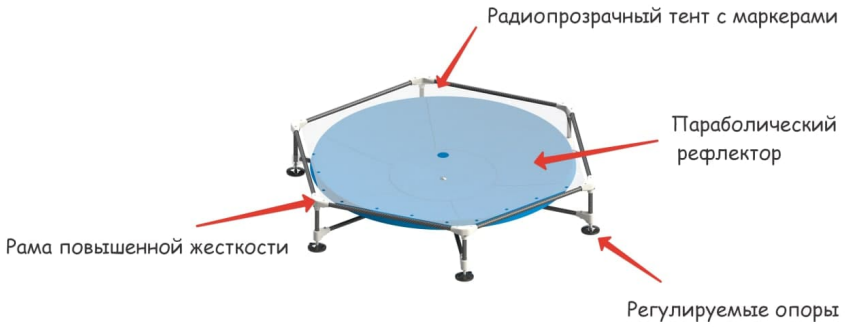
Использование платформы Клевер

В качестве устройства для переноса облучателя над фокальной плоскостью был выбран «Clover 4 Code». Выбор Клевера – не случаен. Данный дрон имеет широкий спектр выполняемых задач, возможности для модификации и самостоятельной доработки под конкретные цели. Поскольку предполагается перенос массивного модуля электроники с облучателем, мы поставили на Клевер более мощные моторы и аккумулятор увеличенной ёмкости, установили улучшенные винты. Скорректировав PID-ы, Клевер без проблем взлетал и выполнял полётное задание с подвешенным на него оборудованием.

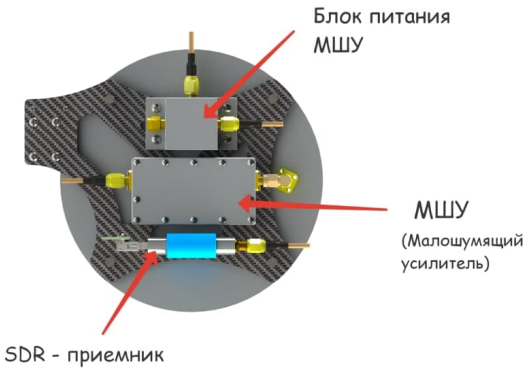
От идеи до практической реализации

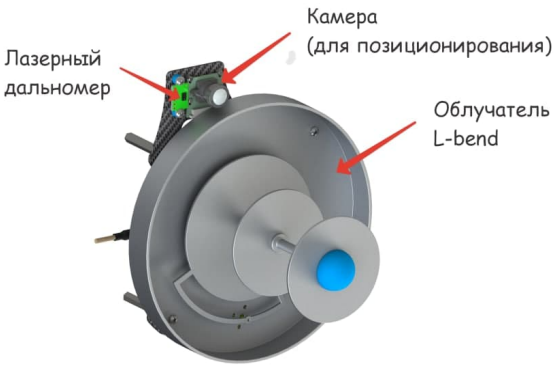
Первым делом наши инженеры создали модели блока электроники, облучателя, опор коптера. Нельзя не отметить факт перераспределения ударной нагрузки на ножки в 3-х направлениях, что повышает срок службы оборудования, а также «выживаемость» дрона в случае аварийной посадки.



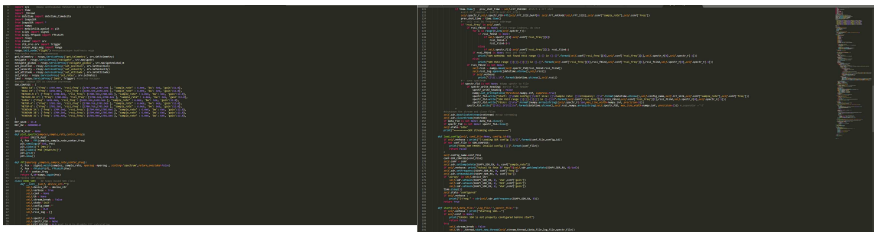


Блок электроники включает в себя всю необходимую аппаратуру для приёма и записи сигнала со спутника: SDR-приёмник, малошумящий усилитель (МШУ), блок питания МШУ, микрокомпьютер «Raspberry-Pi 4», облучатель.





Следующим шагом стало написание программной части. Было создано ПО для автономного полёта и записи принимаемого сигнала. Запись сигнала осуществлялась на Raspberry.



https://github.com/petayyyy/Lorett/ (ссылка на наш репозиторий).

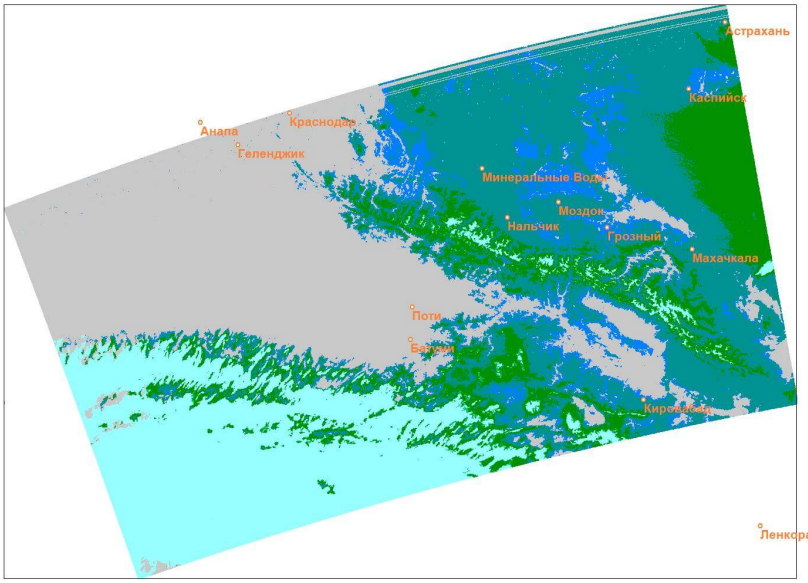
Итак, в чём же новизна нашей разработки? Вместо роботизированного манипулятора, облучатель над «тарелкой» переносит дрон. Клевер взлетает и начинает полёт по заранее рассчитанной траектории приёма спутниковых данных (траектория рассчитывается на основе орбитальных параметров конкретного космического аппарата). Координаты пролёта спутника конвертируются в координаты маркеров, которые закреплены над плоскостью тарелки на радиопрозрачном баннере. Ориентируясь по ArUco-маркерам, Клевер пролетает по заданной траектории, принимая сигнал со спутника на облучатель и записывая его через SDR в память

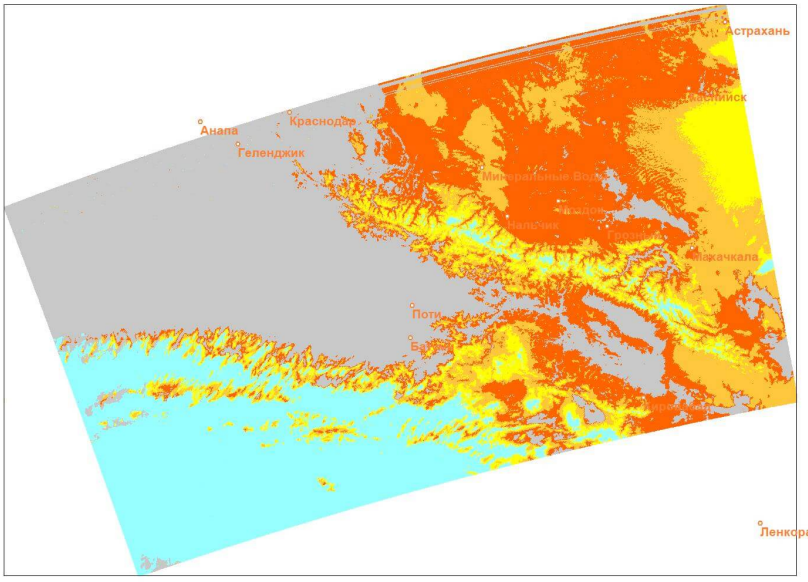
микрокомпьютера. По завершению пролёта, дрон совершает посадку. Для визуализации принципа действия дрона, предлагаем Вам взглянуть на фрагмент симуляции:

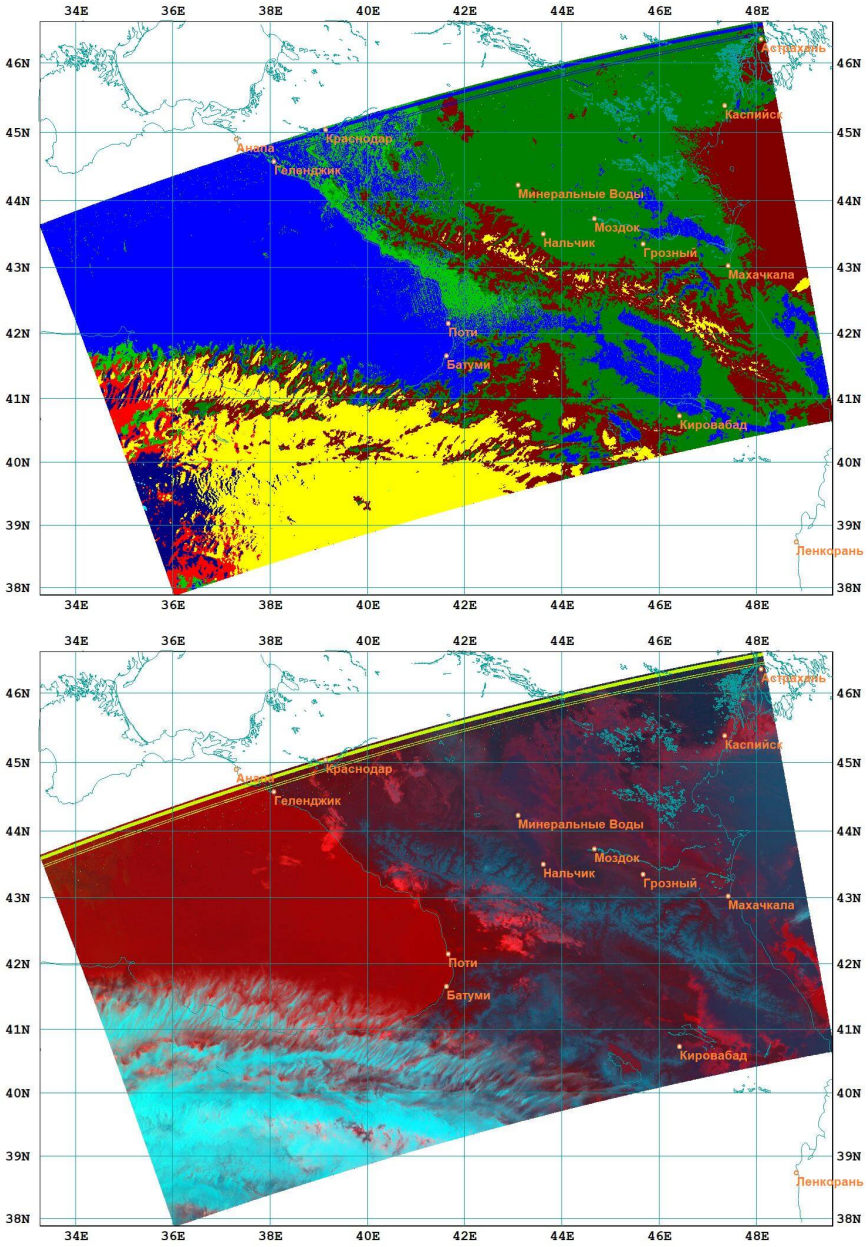
https://youtu.be/tWtlljBtSvw

После посадки дрона, мы переносим принятый сигнал на рабочий ноутбук, где приступаем к демодулированию и декодированию. Затем, можно начинать постобработку и анализ полученных данных. С помощью тематического ПО и анализа спутниковых снимков в разных цветовых каналах мы можем «вытащить» из них следующие данные: температура воды и подстилающей поверхности/верхней границы облаков, тип подстилающей поверхности, альбедо поверхности, тип облаков, водозапас облаков. Полученные данные помогут составить точный прогноз погоды для территории, с которой был принят сигнал.

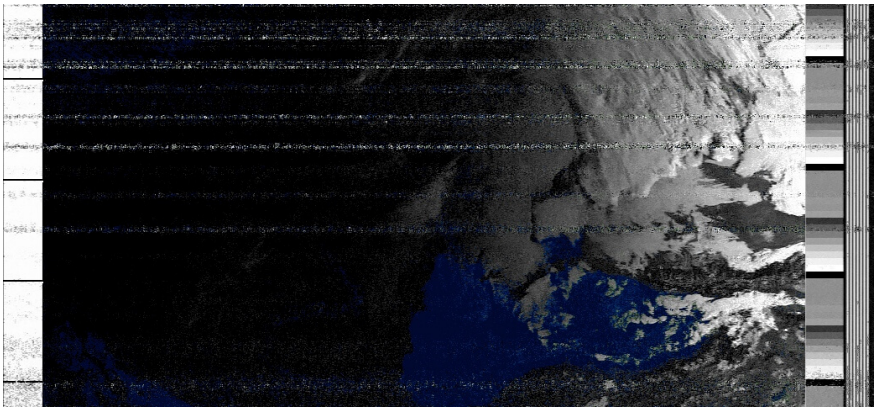
Примеры обработанных снимков:







Пример снимка без обработки:



Взлёт разрешён

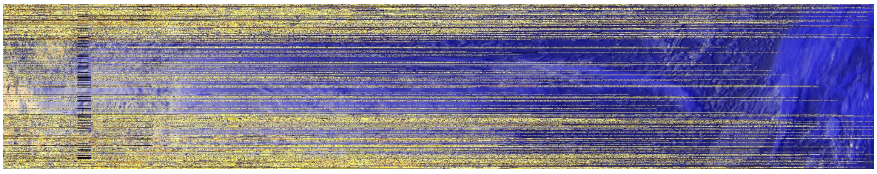
Положив рядом жгут, много перевязочного материала, пустырник, мы преступили к испытаниям. Полёты проходили в разных местах нашей Родины, а также в разные сезоны. Первая серия полётов была в Нальчике (Кабардино-Балкарская Республика) на образовательной программе ОЦ «Сириус» в марте 2021 года. Горный ветер не щадил никого, ни уши испытателей, ни Клевер. Было много сомнений насчёт практической применяемости нашей новинки, но в конечном счёте, мы приняли сигнал. Результат декодирования произвёл эффект разорвавшейся бомбы. На экране ноутбука появилась полоска, шириной в несколько пикселей, которую смог воспринять HRPT-Reader (ПО для постобработки). Данное изображение было возможно просмотреть в разных цветовых каналах. Это могло значить только одно – при более «плавном» перемещении дрона, мы имеем все шансы получить идеальное изображение, как со станции «Link2Space».

Принятая полоска:

Видео тестовых полетов: https://youtu.be/xP1Ne3j95zU

Вторые полётные испытания, после частичной доработки дрона (установка более мощных моторов, новых пропеллеров, доработка ПО), прошли в подмосковной Кубинке, на конкурсе «ИнтЭРА», проходившего в рамках международного военно-технического форума «Армия-2021». Казалось бы, всё должно пройти гладко, но одновременно с нами, на полигоне Алабино (в радиусе 5 км от нас) проходили «Международные армейские игры», где соревновались, в том числе военные-связисты. И результат их работы мы целиком и полностью ощутили на себе. Их средства подавления сигналов различного рода работали безотказно, что даже при приёме мощнейшего метеоспутника (китайский FengYun) мы получали 9 Гб «белого шума».

Белый шум:



Видео с испытаний: https://youtu.be/k1ORpj3o-ew

Доработка проекта в рамках "CopterHack 2022"

Модификация конструкции дрона

Были доработаны опоры "Клевера" таким образом, что блок электроники и облучатель стали лучше защищены в случае аварийной посадки дрона. Также был решен вопрос совместимости компонентов и подобрана их наилучшая конфигурация. Каждый компонент так и просится на свое место.

https://petayyyy.gitbook.io/copter-for-space/more/dev/3d-modeli-detalei - Ссылка на 3D-модели деталей

https://disk.yandex.ru/d/SrQ9xaMjKvO6vw - STL-файлы моделей

Специальное оборудование

Приёмная часть состоит из облучателя L-диапазона (ряд металлизированных дисков разных размеров, закрепленных при помощи шпильки на некотором расстоянии друг от друга), малошумящего усилителя соответствующих частот, тройника смешения(специальное устройство, позволяющее подавать питание на МШУ и принимать сигнал по одному проводу) и Программируемого радиоприемника (airspy sdr).

МШУ (Малошумящий усилитель) – устройство, входящее в состав базовой станции (БС) и используемое для повышения чувствительности приемника в восходящем направлении UL (Uplink).

Software-defined radio/SDR (рус.Программно определяемая радиосистема) — радиопередатчик и/или радиоприёмник, использующий технологию, позволяющую с помощью программного обеспечения устанавливать или изменять рабочие радиочастотные параметры, включая, в частности, диапазон частот, тип модуляции или выходную мощность, за исключением изменения рабочих параметров, используемых в ходе обычной предварительно определённой работы с предварительными установками радиоустройства, согласно той или иной спецификации или системы.

Блок питания МШУ - регуляция входного и выходного напряжений.

Облучатель − сосредоточенный элемент параболической антенны, находящийся в её фокусе (фазовом центре) или фокальной плоскости, формирующий диаграмму направленности и поляризацию антенны.

Программное обеспечение

Основываясь на открытом программном коде был создан автоматический демодулятор-декодер (далее Д-Д), который мы назвали "SatDump". Позднее, возможности Д-Д были оптимизированы для Raspberry Pi с некоторыми дополнительными возможностями. Например - автоматическая демодуляция и декодирования данных с метеорологических спутников с возможностью просмотра спутниковых снимков в браузере.

https://github.com/petayyyy/Lorett// - исходный код

https://gitlab.com/lpmrfentazis/HRPTAutoDecoder - автодекодер.

https://gitlab.com/lpmrfentazis/lorettorbital/-/blob/develop/lorettOrbital/orbital.py - библиотека “LoReTT Orbital” для расчёта расписания пролётов спутников и создания траектории перемещения облучателя дроном в фокальной плоскости “зеркала”

https://github.com/petayyyy/Lorett// - сборник ПО для функционирования комплекса-конструктора, описанного в "ГитБуке"

Документация

Концепция конструктора подразумевает из себя возможность освоения обучающимися определенных компетенций в областях инжиниринга, программирования, радиотехники, летающей робототехники и работы с данными дистанционного зондирования Земли. Для комфортного освоения мы подготовили все необходимые учебные материалы:

https://disk.yandex.ru/i/hJjB1w0ekQ0Lug - руководство по эксплуатация комплекса.

https://petayyyy.gitbook.io/copter-for-space/ - инструкция по сборке, настройке и эксплуатации комплекса в удобном формате “ГитБука”.

https://disk.yandex.ru/d/WOpfbO74N2cmPQ - инструкция по демодуляции и декодированию сигнала с метеорологических спутников, а также анализу полученных космических снимков.

Результаты

1. Подготовлен сборник подробной документации по работе с комплексом-конструктором.
2. Доработан конструктив дрона, улучшена “живучесть” и сбалансированность компонентов.
3. Проведены образовательные мероприятия на площадках в Гимназии МГУ и “ФизТех-Лицее им.Капицы”, в ходе которых обучающиеся произвели монтаж и настройку комплекса-конструктора, а также выполнили сеансы приема данных с метеоспутников. После образовательной программы в ФТЛ был создан “ГитБук” для удобства работы с учебными материалами.
4. Создано новое ПО для расчета расписания пролетов спутников и, в соответствии с параметрами ИСЗ, выбора оптимальной траектории движения дрона-манипулятора в фокальной плоскости “зеркала”. Разработан автоматический демодулятор-декодер “SatDump”.

Список дополнительных компонентов

Основные доп.компоненты:

- DIATONE MAMBA TOKA 2207.5/2450KV - моторы.
- HQProp DP6X4X3 PC (2 пары) - пропеллеры.
- ONBO 4200mAh 4S 35C Lipo Pack - аккумулятор.

Для более подробного ознакомления со списком компонентов и узнать, где приобрести данные комплектующие вы можете перейти в таблицу по ссылке ниже:

https://docs.google.com/spreadsheets/d/19Gwm3lu31WgYOmuQ6WVm2aCMhhrhTMLTLWVspxvzfWs/edit#gid=0

Для приобретения специального оборудования, свяжитесь с ИК "LoReTT", ввиду эксклюзивности данных компонентов.

Контакты: ООО "Лоретт", Россия, г. Москва, Инновационный центр "Сколково", Большой бульвар, 42, стр. 1, офис 334, 121205.

+7 (985) 727-7630.

contact@lorett.org.

Перспективы

Хочется отметить перспективы использования дрона или сети дронов для переноса облучателя на крупных приёмных комплексах. В будущем возможно использовать платформу Pelican. Данная конструкция значительно упростит обслуживание крупных антенн, а также снизит количество потенциальных узлов отказа.

Безусловно, работа школьников с конструктором развивает в них не только инженерно-технические компетенции, но и научно-исследовательские! Платформа “Clover” предоставляет широкий спектр возможностей для модификации, кто знает, может быть, найдётся Левша, который модифицирует исходный конструктор так, что нам и не снилось! Познание мира - в экспериментах. А команда “SpaceClowns”, совместно с компанией “LoReTT” поможет в формировании будущих “Кулибиных”.

CopterCat_cm4

CopterHack-2022, команда **CopterCat**.



Информация о команде

Состав команды:

- Лапин Матвей (https://t.me/l_motya), инженер/программист.
- Коновалов Евгений (https://t.me/egnknvlv), инженер/друг.
- Скандаков Егор (https://t.me/hjbaa), друг.
- Джалилов Эмиль, друг.

Описание проекта

Разработка современной платы под прошивку PX4 FMUv6U, разерами 55x40 мм и дополнительным модулем WiFi для реализации классных штук, например, распределённой сети.

Идея проекта

Полётник на stm32h7 с местом для RPi CM4 и встроенной ESP32 для создания распределённой сети.

Планируемые результаты

Плата полётного контроллера FMUv6U и API для взаимодействия с распределённой сетью через RPi.

Использование платформы "Клевер"

На этапе проекта: отладка и демонстрация возможностей. После: использования CopterCat в качестве основного.

Спецификация

FMU

- STM32H753IIK6 480Mhz Cortex-M7
- 2Mb + 256Kb FLASH
- 1Mb RAM
- ICM20602, ICM42605, BMI088, BMP388, BMM150
- Полностью совместима со стандартом FMU-v6u

Raspberry Pi

- Поддержка платы RPi CM4.
- Слот для SD-карты.
- Возможность прошивки встроенной eMMC.
- CAT24C256 EEPROM.
- Поддержка 2-х камер (CAM0-две линия, CAM1-четыре линии).
- Поддержка USB-OTG.

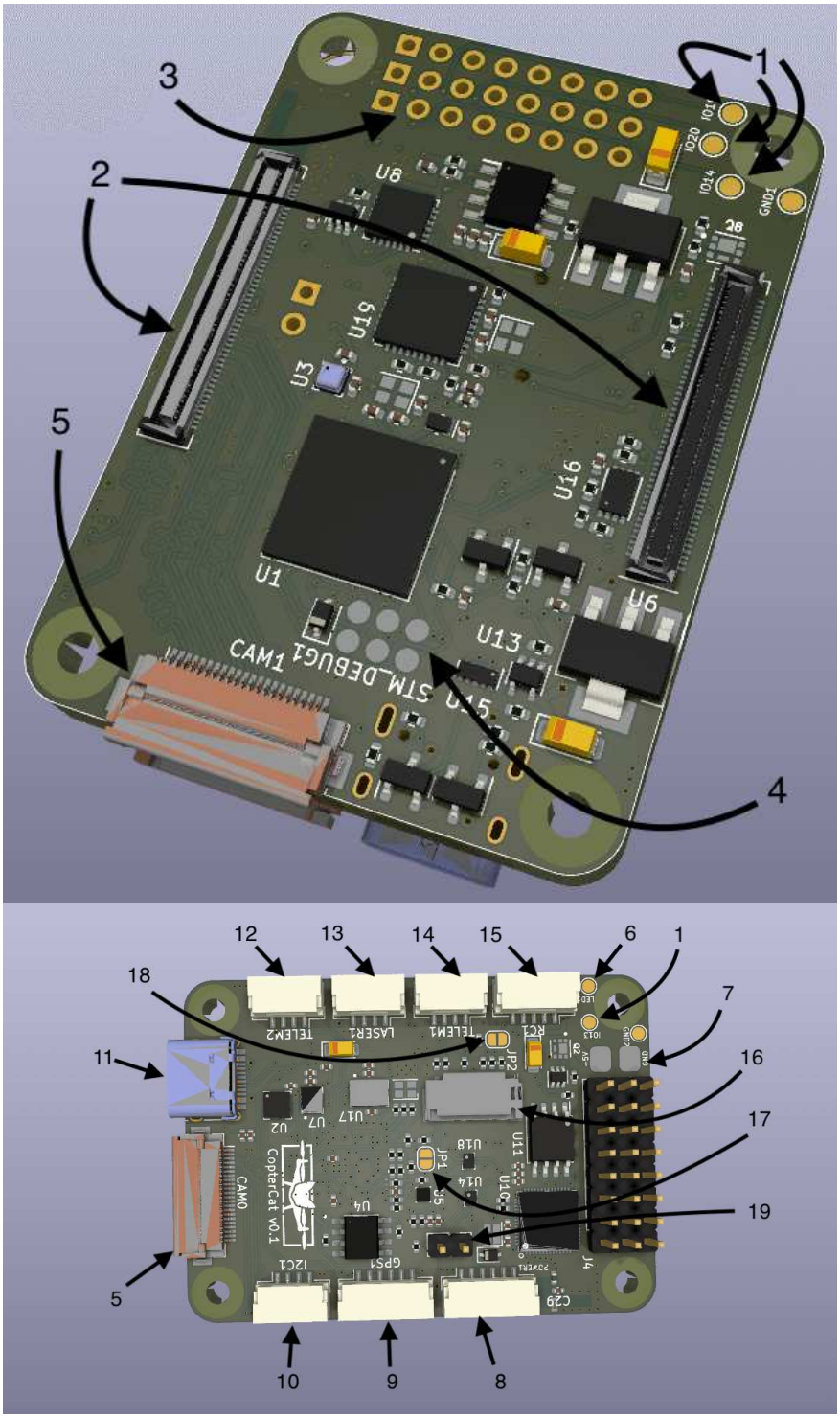
ESP32

- 16MB внешней FLASH (W25Q128JVS).
- 8MB внешней PSRAM (LY68L6400SLIT).
- Встроенная антенна.
- USB-TTL конвертор.

Остальное

- USB-HUB USB2514B.
- USB-PD с физическим переключением.
- Связь ESP32 и STM32 через UART.
- 3 варианта питания.
- 4 универсальных GPIO от ESP32.
- USB Type-C.
- Размеры 40x55 мм, плата 4 слоя.

Разъёмы и перемычки



1. GPIO ESP32 4 порта ввода вывода для подключения внешнего оборудования.
2. Коннекторы RPi CM4.
3. Выводы ESC 8 шт.
4. Контакты программирования и отладки JTAG STM32.
5. Коннекторы камер (шлейф 22 контакта с 0.5 мм расстояния между проводниками).
6. Контакт подключения адресной ленты.
7. Контакты основного питания 5В.

8. JST-6 стандартного шлейфа питания PX4.
9. JST-6 GPS+компас+5В.
10. JST-4 I2C+5В.
11. USB Type-C.
12. JST-4 UART7+5В.
13. JST-4 I2C RPi+3.3В для подключения дальномера.
14. JST-4 UART5+5В.
15. JST-5 Стандартный разъём для подключения приёмника управления.
16. Слот SD карты (для RPi).
17. Джампер BOOT для STM32.
18. Джампер RPIBOOT для прошивки eMMC модуля RPi CM4.
19. Джампер переключения режима работы USB разъёма (при замкнутой перемычке, USB работает как вход HUB и при подключении к компьютеру будут отображаться STM32, ESP32 и RPi CM4 в режиме OTG; при разомкнутой перемычке, USB будет работать для подключения внешних устройств к RPi, например стереокамеры).

Загрузка прошивки

FMU

При первом запуске, в микроконтроллер придётся загрузить PX4-bootloader через порт JTAG. Подробная инструкция здесь.

Для подключения к компьютеру:

1. Замкните перемычку 19.
2. Подключите USB Type-C к компьютеру.
3. Устройство должно появиться в QGC.

Также прошивку можно произвести через RPi:

1. Установите RPi CM4 в коннектор на плате.
2. Разомкните перемычку 19.
3. Устройство появиться в папке `/dev` на RPi.

ESP32

Написать программу можно либо в Arduino IDE, либо в VS Code с плагином esp-idf. Далее скомпилировать и загрузить в микроконтроллер. Загрузить можно двумя способами.

С компьютера:

1. Замкните перемычку 19
2. В подключённых устройствах появиться CP2104
3. Загрузите прошивку в соответствии с инструкцией к выбранной IDE

С RPi CM4:

1. Установите RPi CM4 в коннектор на плате.
2. Разомкните перемычку 19.

3. Скомпилируйте ваш код в .bin формат.
4. Загрузите полученный файл на RPi.
5. Загрузите прошивку в микроконтроллер с помощью esptool.py (описание+установка).

RPi CM4

Слот SD карты работает как на стандартной RPi. Для плат с eMMC порядок загрузки операционной системы не отличается от CM4 IO Board (инструкция).

Общая информация

- Все требуемые для заказа файлы находятся в папке `/gerbers`.
- Проект выполнен в программе KiCAD v6.
- Библиотеки компонентов взяты с сайта snapeda.

Система мониторинга воздуха

CopterHack-2022, команда **C305**.



C305

Информация о команде

Состав команды:

- Мамбетов Руслан - CEO, разработчик встраиваемого ПО, дизайнер печатной платы;
- Филимонов Сергей - программист (интерфейс пользователя, визуализатор);
- Смадыч Никита - инженер-программист (навигация дрона);
- Антонов Георгий - инженер (сборка и настройка дрона, создание 3D-моделей).

Описание проекта

Идея проекта

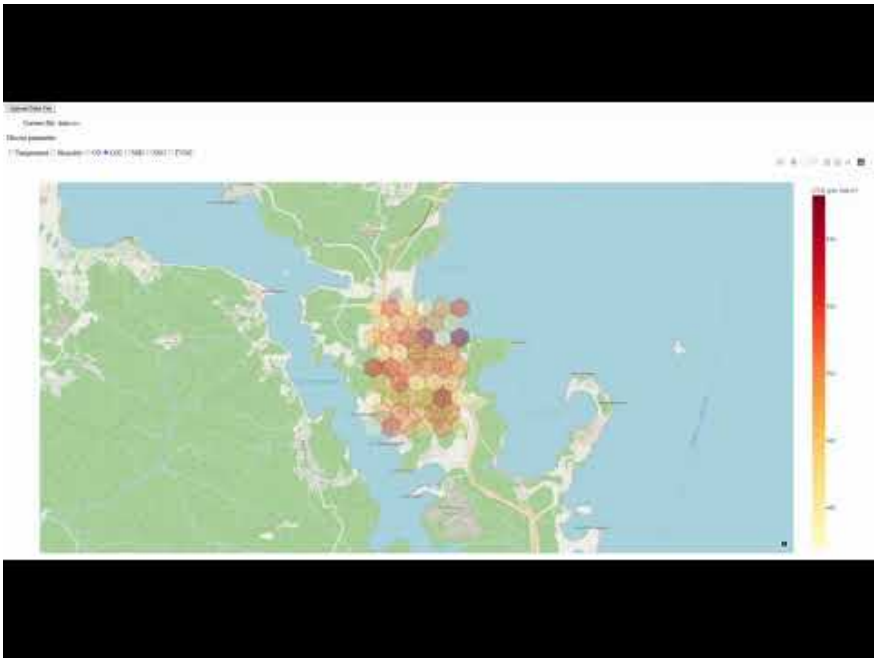
Идея представляемого проекта заключается во внедрении в платформу “Клевера” дополнительных модулей предназначенных для измерения содержания в воздухе газов и органических соединений используемых в промышленности, а также лидара (RPLIDAR), предназначенного для автономной навигации.

Наши нововведения позволят использовать коптер для поиска участков с повышенным содержанием потенциально опасных для человека и окружающей среды газообразных веществ, строить карту загрязненности воздуха и обозначать на ней зоны с критической концентрацией газов. Также дрон сможет выполнять задачи по обнаружению утечек на газовых магистралях. В совокупности это даст возможность обнаруживать локальные зоны выброса опасных газов и впоследствии своевременно их устранять.

Применение лидара обусловлено необходимостью облета дроном возможных препятствий при выполнении миссии, либо работы в замкнутых помещениях, например, локализация зон утечек вредных газов в больших закрытых производственных и складских помещениях.

Проект, в том числе, направлен на мониторинг углеродного следа, являющимся важным фактором в вопросе глобального изменения климата. Статистика показывает, что повышение уровня выбросов парниковых газов в атмосферу за последние годы стремительно растет, что приводит к повышению температуры поверхности планеты и значительному увеличению числа природных катастроф и аномалий. Примером влияния глобального потепления являются и недавние события - аномальная жара летом 2021 года.

Презентационный ролик



Репозиторий проекта

Документация к проекту

- Сборка коптера газоанализатора
- Настройка коптера для автономных полётов
- Система планирования полета
- Пользовательские приложения
- Настройка лидара
- Процесс загрузки прошивки
- Подключение датчиков газов

Ресурсы проекта

- Исходный код программ для Raspberry Pi
- Исходный код наземной станции
- Модели крепления лидара
- Прошивка для газоанализатора (Arduino)
- Прошивка для газоанализатора (STM)
- Gerber файлы платы газоанализатора

Контроль соблюдения ПДД на выделенной полосе с дроном

Автор

- Смирнов Даниил, Telegram: londrwus.

Введение

Автобусная полоса — это часть дороги только для пассажирского транспорта. Выделяют ее специально, чтобы горожане без машин быстрее добирались на работу и обратно домой в час пик. Благодаря выделенным полосам общественный транспорт не стоит в дорожной пробке и удобен для пассажиров. Если водитель обычной легковушки объедет по ней пробку, он получит штраф за выделенную полосу для общественного транспорта

Соблюдение контроля полосы

На данный момент, контроль соблюдения полосы происходит с стационарных камер дорожного движения. Однако, не на всех участках дорог можно расположить подобного вида камеры или камеры могут не работать или вообще не выделять автобусную полосу. Поэтому было создано решения контроля полосы с помощью беспилотника.

Объяснение кода. Написание

Подключаем библиотеки:

```
import rospy
import cv2 #OpenCV
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
from std_srvs.srv import Trigger
from pyzbar import pyzbar
import numpy as np
```

Создаём некоторые переменные:

```
rospy.init_node('flight')
bridge = CvBridge()
dict_flag = []
detect_flag = True
i = -1
```

Для реализации контроля полосы и нахождения машин, автобусов, нам понадобится библиотека OpenCV. Создаём скриншот с изображением с основной камеры для обработки с использованием OpenCV:

```
def lane_control():
    global dict_flag, i, detect_flag, detected_blue_bus
    cv_image = bridge.imgmsg_to_cv2(rospy.wait_for_message('main_camera/image_
    filtered_image = cv2.cvtColor(cv_image, cv2.COLOR_HSV2BGR)
```

На данный момент, автобусы производятся в синем цвете. Поэтому, автобус у нас будет синего цвета.



Прописываем диапазоны цветов для автобуса и машины:

```
# Автобус
lower_blue = np.array([90, 60, 100])
upper_blue = np.array([119, 255, 255])
# Другие машины, допустим, цвет red
lower_red = np.array([160, 20, 50])
upper_red = np.array([190, 255, 255])
```

Данный скрипт требует больших вычислительных мощностей Raspberry Pi. Для ограничения работы скрипта, мы замедляем работу камеры до частоты 10 Гц (`main_camera.launch`):

```
<node pkg="topic_tools" name="cam_throttle" type="throttle" args="messages mai
```

Топик для камеры в этом случае необходимо поменять на `main_camera/image_raw_throttled` с `main_camera/image_raw` .

```
# car
frame = cv2.inRange(filtered_image, lower_red, upper_red)
cnt = cv2.findContours(frame, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
try:
    for c in cnt:
         moments = cv2.moments(c, 1)
         sum_pixel = moments['m01']

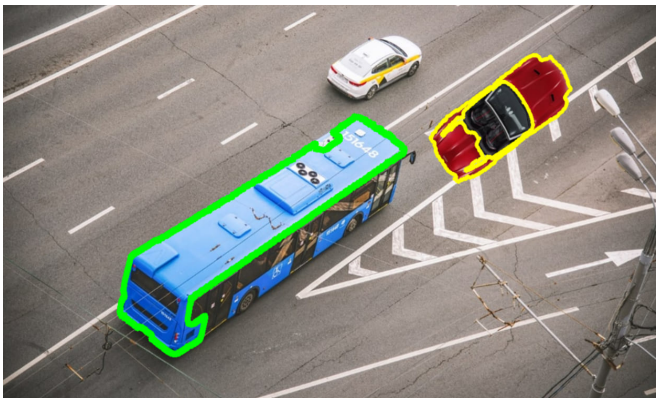
         if sum_pixel >= 100000:
             dict_flag[i][1] = [sum_pixel, int(moments['m10'] / sum_pixel), int
             detected_red_car = True
             cv2.drawContours(image_src, [c], 0, (0,255,255), 6)
except:
    pass

# bus
frame = cv2.inRange(filtered_image, lower_blue, upper_blue)
cnt = cv2.findContours(frame, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
try:
    for c in cnt:
         moments = cv2.moments(c, 1)
         sum_pixel = moments['m01']

         if sum_pixel >= 100000:
             dict_flag[i][0] = [sum_pixel, int(moments['m10'] / sum_pixel), int
             detected_blue_bus = True
             cv2.drawContours(image_src, [c], 0, (0,255,0), 10)
except:
    pass

if detected_red_car == True:
    print("[DEBUG] detected car")
else:
    print("[DEBUG] No cars")
```

Данная программа будет определять машины по её оттенку. Вот что у нас в итоге получилось.



Вот примеры других цветовых диапазонов. Стоит подметить, что цветовые диапазоны вычисляются в HSV, а не в RGB:

```
# Красный в одном диапазоне
red_low1 = np.array([0, 40, 20])
red_high1 = np.array([14, 255, 255])
# Красный в другом
red_low2 = np.array([165, 40, 20])
red_high2 = np.array([180, 255, 255])
# Голубой
lower_blue = np.array([90, 100, 31])
upper_blue = np.array([119, 255, 190])
# Желтый
lower_yellow = np.array([17, 55, 55])
upper_yellow = np.array([32, 200, 200])
#...
```

Иногда надо использовать два значения HSV красного, т. к. они попадают под начало и конец диапазона HSV.

Данная программа будет только обнаруживать, выдавать отчет об объекте, рисовать контуры объектов. Если же мы захотим, чтобы выводить на LED-ленту нарушило ли ПДД данный объект, то можно добавить переменные и добавить функцию:

```
Led = { "red": [255, 0, 0], "blue": [0, 0, 255] }
count_led = 72 #Если же на вашей LED-ленте 72 светодиода, если 58 - ставьте зна
Led_full = [0]*count_led
def led_print():
    global count_led, Led_full
    set_effect(r=0, g=0, b=0)
    rospy.sleep(4)
    a = []
    for i in range(count_led):
        a.append(LEDState(i, Led[Led_full[i][0]][0], Led[Led_full[i][0]][1], Le
    set_leds(a)
```

Дополним функцию `lane_control`:

```
S = sum([ dict_flag[i][0] for j in range (len(dict_flag[i]))])
j, k = 0, 0
print(dict_flag[i][0][0])
for k in range(int(dict_flag[i][0][0]/S * count_led)):
    Led_full[k+j] = ["red", dict_flag[i][0][1], dict_flag[i][0][2]]
j+=k
k = 0
print(j, k )
for k in range(int(dict_flag[i][1][0]/S * count_led)):
    Led_full[k+j] = ["blue", dict_flag[i][1][1], dict_flag[i][1][2]]
j+=k
k = 0
print(j, k)

if abs(Led_full[0][1] - Led_full[count_led/2][1]) > 50:
    Led_full.sort(key=lambda Led_full: Led_full[1]) # сортировка по x
else:
    Led_full.sort(key=lambda Led_full: Led_full[2]) # сортировка по y

print(Led_full)
```

После этого наша светодиодная лента будет обозначать нарушил ли данный объект ПДД или нет.

Но даже после такого количества кода, наш дрон не будет следовать за нарушителем. Давайте сделаем так, чтобы дрон мог преследовать нарушителя постоянно, чтобы он не мог скрыться с места нарушения. Создадим базовые значения

```
# Библиотеки
import tf
import tf2_ros
import geometry_msgs.msg
import tf2_geometry_msgs
import numpy as np
from geometry_msgs.msg import Vector3Stamped, Point, PointStamped

ar = "aruco_13" # ID нарушителя
tfBuffer = tf2_ros.Buffer()
listener = tf2_ros.TransformListener(tfBuffer)

l_z = 2 # Для будущего кода дальномера

def range_callback(msg):
    global l_z
    l_z = msg.range # Вывод Z по дальномеру

rospy.Subscriber('rangefinder/range', Range, range_callback)

def get_aruco_pose(frame_id): # Возвращает координаты, с помощью преобразование
    global tfBuffer, listener
    try:
        trans = tfBuffer.lookup_transform(frame_id, ar, rospy.Time())
    except:
        return None
    pnt_l0 = tf2_geometry_msgs.do_transform_point(PointStamped(point=Point(x=0,
    l0 = np.array([pnt_l0.point.x, pnt_l0.point.y, pnt_l0.point.z])
    return l0

def get_body_pose(frame_id): # Возвращает координаты, с помощью преобразование
    global tfBuffer, listener
    try:
        trans = tfBuffer.lookup_transform(frame_id, "body", rospy.Time())
    except:
        return None
    pnt_l0 = tf2_geometry_msgs.do_transform_point(PointStamped(point=Point(x=0,
    l0 = np.array([pnt_l0.point.x, pnt_l0.point.y, pnt_l0.point.z])
    return l0

def remove_0_vel(vel): # Скорости полета в POSITION
    if np.linalg.norm(vel[:2]) < 0.045:
        vel[0] = 0
        vel[1] = 0
    return vel
```

Создадим функцию, которая будет производить весь этот полет. Для начала сделаем так, чтобы дрон находил нарушителя.

```
def follow_violator():
    z = 1.5

    navigate_wait(z=1.5, speed=1, frame_id="body", auto_arm = True)
    rospy.sleep(2)
    set_effect(r=255, g=255, b=0)
    print("[DEBUG] started navigation")
    navigate_wait(x=0, y=0, z=z, speed=0.8, frame_id=ar, yaw=float('nan'), tole
```

В итоге, мы написали код, но давайте сделаем так, чтобы он постоянно двигался за ним и улучшал его позиционирование.

```
FRQ = 5
r = rospy.Rate(FRQ)
prev_vel = None
prev_pa = None
prev_t = rospy.get_time()
st_t = rospy.get_time()
d = 10
# Вычисляет на сколько надо перемещаться, за счет изменения кадров (прошлог
while d > 1 or (rospy.get_time() - st_t < 0.5):
    pb = get_body_pose("aruco_map")
    pa = get_aruco_pose("aruco_map")
    now = rospy.get_time()
    if prev_pa is None:
        if pb is None:
            r.sleep()
            continue
        navigate(x=pb[0], y=pb[1], z=z, speed=1, frame_id="aruco_map") # Сн
        set_effect(r=150, g=0, b=255, effect='blink')
        prev_pa = pa
        prev_t = now
    else:
        if pb is not None:
            d = np.linalg.norm(pb[:1]-pa[:1])
        if pa is not None:
            set_effect(r=150, g=0, b=255)
            vel = (pa-prev_pa)/(now-prev_t+0.0001)
            vel = np.clip(vel, -0.7, 0.7)

            vel = remove_0_vel(vel)
            if prev_vel is not None:
                vel = vel*0.1 + prev_vel*0.1

            t = pa[:1] + vel[:1]*(0.1/FRQ)*2.7
            set_position(x=t[0], y=t[1], z=z, frame_id="aruco_map")
            prev_pa = pa.copy()
            prev_vel = vel.copy()
            prev_t = now
        else:
            navigate(x=pb[0], y=pb[1], z=z, frame_id="aruco_map")
            set_effect(r=150, g=0, b=255, effect='blink_fast')
            print("NO pa and vel")

    r.sleep()

set_effect(r=255, g=150, b=0)
z_st = 1.5

st_t = rospy.get_time()
z_vel = 0.35

Z = z_st
r = rospy.Rate(20)
# Вычисляет на сколько надо изменить координату Z, за счет изменения кадров
while l_z > 0.09:
    pb = get_body_pose("aruco_map")
    pa = get_aruco_pose("aruco_map")
    now = rospy.get_time()
    if prev_pa is None:
        set_position(x=0, y=0, z=-0.01, frame_id="body")
        set_effect(r=255, g=150, b=0, effect='blink_fast')
        print("NO prev_pa")
        prev_pa = pa
        prev_t = now
    else:
        if pb is not None:
```

```
                d = np.linalg.norm(pb[:1]-pa[:1])
            if pa is not None:
                set_effect(r=255, g=150, b=0)
                if np.linalg.norm(pa - prev_pa) < 0.0001 and prev_vel is not No
                    vel = prev_vel.copy()*0.97
                else:
                    vel = (pa-prev_pa)/(now-prev_t+0.0001)
                if l_z > 0.5:
                    vel = np.clip(vel, -0.7, 0.7)
                else:
                    vel = np.clip(vel, -0.5, 0.5)

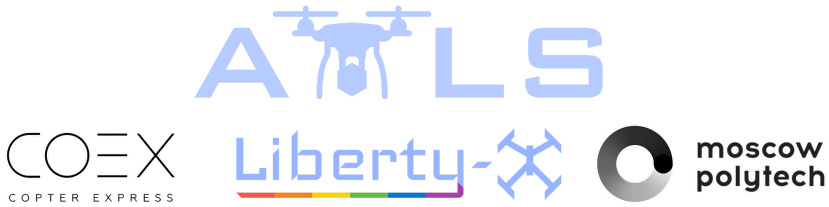
                vel = remove_0_vel(vel)
                if np.linalg.norm(vel[:1]) < 0.02 and d <= 0.07:
                    Z = -(rospy.get_time()-st_t)*(0.1) + z_st
                else:
                    Z = -(rospy.get_time()-st_t)*z_vel + z_st

                t = pa[:1] + vel[:1]*(0.1/FRQ)*2.2
                set_position(x=t[0], y=t[1], z=Z, frame_id="aruco_map")
                prev_pa = pa.copy()
                print("Z = ", Z)
                prev_vel = vel.copy()
                prev_t = now
            else:
                set_position(x=0, y=0, z=-0.01, frame_id="body")
                set_effect(r=255, g=150, b=0, effect='blink_fast')
                print("NO pa and vel")

        r.sleep()
```

Вывод: мы написали довольно короткий, для поставленных перед ним задач, код. Данный код фиксирует правонарушение, регулирует вариативность определения транспортного средства и совершает преследование нарушителя

Autonomous Multirotor Landing System (AMLS)



Цель проекта: Автоматически посадить дрон в движении

Статья о проекте AMLS

В этой статье будет описание нашего проекта, а именно, алгоритмов оптической стабилизации, удержания по GPS, следование по GPS, удержания высоты, системы захватов, крыши, измерения освещённости, скорости движения платформы. Также, после прочтения статьи станет понятно, как работает проект AMLS!

Основной репозиторий проекта на GitHub

https://github.com/XxOinvizioNxX/Liberty-Way

Разработчики

- Павел Нешумов
- Андрей Кабалин
- Владислав Яснецкий



Оглавление

- 0. Как это работает?
  - 0.1. Видео про наш проект
- 1. Удержание по GPS и полёт по точкам
  - 1.1. Чтение данных из UART
  - 1.2. Парсинг протокола UBlox
  - 1.3. Установка текущей путевой точки
  - 1.4. Изменение точек (Чтобы летать по координатам)
  - 1.5. Удержание конкретной позиции
- 2. Следование за платформой
- 3. Компас
- 4. Удержание высоты (барометр)
- 5. Оптическая стабилизация
  - 5.1. Так сложно и так важно
  - 5.2. Первые шаги
  - 5.3. Инверсия
  - 5.4. Версия на Java
  - 5.5. Liberty-Way
  - 5.6. Связь с дроном
  - 5.7. Подвес для камеры
- 6. Платформа Eitude
  - 6.1. Система захватов
  - 6.2. Крыша
  - 6.3. Спидометр
  - 6.4. Датчики уровня освещённости
- 7. Заключение

---

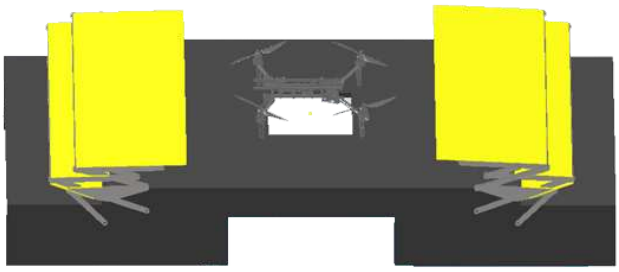
0. Как это работает?

Система состоит из двух частей:

- Дрона



- И платформы, или подвижной (на автомобиле), или статичной (на постамате)



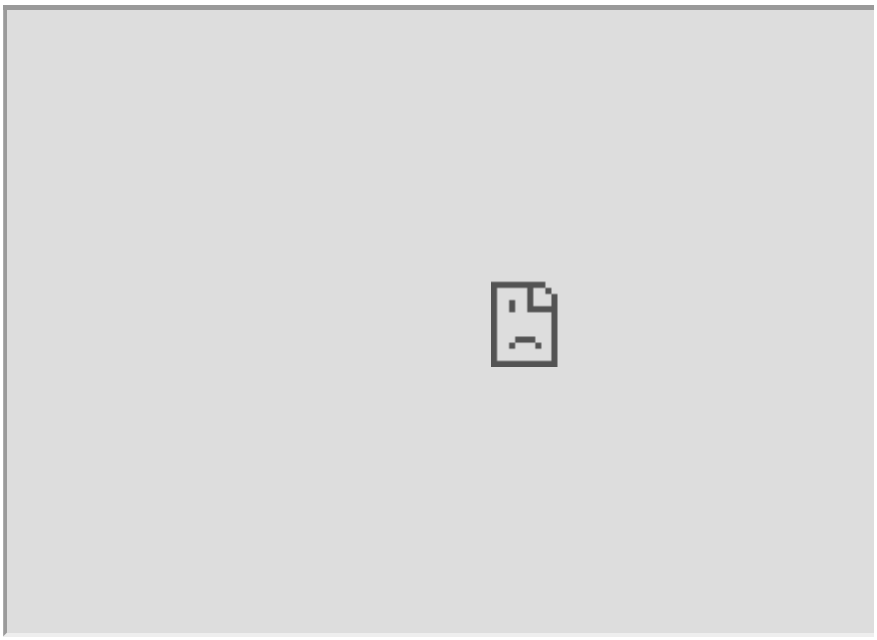
Порядок действий выглядит так:

- В начале, дрон с посылкой находится далеко от платформы. По сотовой связи или другому радиоканалу, ему отправляются GPS координаты платформы (на дроне установлен модуль Liberty-Link, способный корректировать его положение, независимо от прошивки полётного контроллера. (Модуль ставится в разрыв линии от приёмника (пульта) до полётника).
- Дрон летит на эти координаты. В процессе полёта они могут обновляться (но не часто, чтобы не нагружать канал).
- Когда дрон подлетает близко к платформе, включается стабилизация по GPS. Тут уже данные передаются по ближней радиосвязи и с большой частотой, чтобы дрон успевал за платформой.
- Вместе с этим, происходит снижение дрона (на платформе и дроне есть барометры). Снижение происходит до высоты 1.5-2 метра над платформой.
- В момент снижения включается оптическая (прецизионная) стабилизация с платформы. Как только камера увидит метку, алгоритм

"захватит" дрон.

- После включения оптической стабилизации, GPS отходит на второй план и является страхующим (на случай, если что-то пойдёт не так, снова включается GPS стабилизация).
- Во время оптической стабилизации на дроне закреплена ArUco метка, которую видит камера на платформе и по ближней радиосвязи посылает на дрон корректирующие значения.
- Одновременно с оптической стабилизацией включается и посадка. Этот алгоритм искусственно плавно снижает setpoint по высоте (Z) до некоторого порога.
- Как только снижение достигло нужного значения, подаётся команда на платформу для активации механических захватов, которые должны поймать дрон.
- После поимки дрона, происходит его обслуживание, разгрузка посылки и закрытие крыши над платформой (для защиты от неблагоприятных погодных условий).
- На этом посадка завершена!

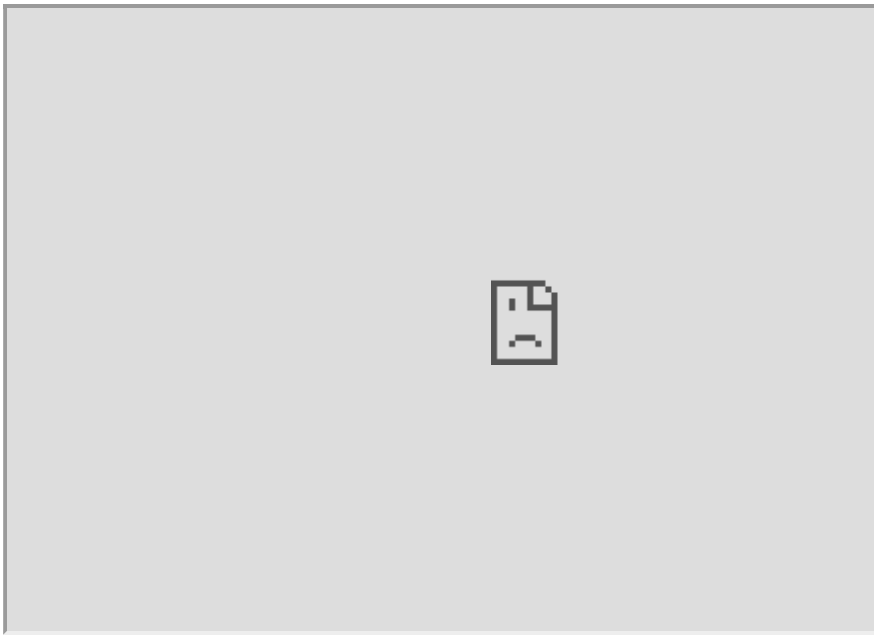
Видео про наш проект (кликабельно)



1. Удержание по GPS и полёт по точкам

Как уже было сказано ранее, на дроне установлен "универсальный" модуль Liberty-Link, принимающий команды с платформы и корректирующий положение дрона, вмешиваясь в сигнал с пульта управления (подробнее об этом в следующих пунктах).

В Liberty-Link будет встроенный GPS модуль, и, соответственно, возможность поддержания положения по GPS и следования по точкам. Результат работы алгоритма поддержания позиции по GPS (кликабельно):



GPS-модуль будет использоваться из семейства UBlox (например, UBlox Neo-M8). Установлено будет 1 или 3 (для минимизации погрешности) модуля.



Модули работают по UART-интерфейсу. Настроены на частоту отправки 5 раз в секунду. Прошивка Liberty-Link будет читать данные с модулей и вычислять координаты текущего положения.

1.1. Чтение данных из UART

Чтение данных с модуля в буфер выглядит так:

```
// Read data from the GPS module
while (GPS_serial.available() && new_line_found == 0) {
    // Stay in this loop as long as there is serial information from the GPS av
    char read_serial_byte = GPS_serial.read();
    if (read_serial_byte == '$') {
        // Clear the old data from the incoming buffer array if the new byte ec
        for (message_counter = 0; message_counter <= 99; message_counter++) {
            incoming_message[message_counter] = '-';
        }
        // Reset the message_counter variable because we want to start writing
        message_counter = 0;
    }
    // If the received byte does not equal a $ character, increase the message_
    else if (message_counter <= 99)
        message_counter++;

    // Write the new received byte to the new position in the incoming_message
    incoming_message[message_counter] = read_serial_byte;

    // Every NMEA line end with a '*'. If this character is detected the new_li
    if (read_serial_byte == '*') new_line_found = 1;
}
```

1.2. Парсинг протокола UBlox

После, из заполненного буфера вычисляются широта, долгота, тип корректировки (2D, 3D) и количество спутников. Парсинг GPS данных протокола UBlox выглядит так:

```
// If the software has detected a new NMEA line it will check if it's a valid
if (new_line_found == 1) {
    // Reset the new_line_found variable for the next line
    new_line_found = 0;
    if (incoming_message[4] == 'L' && incoming_message[5] == 'L' && incoming_me
        // When there is no GPS fix or latitude/longitude information available
        // Set some variables to 0 if no valid information is found by the GPS
        l_lat_gps = 0;
        l_lon_gps = 0;
        lat_gps_previous = 0;
        lon_gps_previous = 0;
        number_used_sats = 0;
    }
    // If the line starts with GA and if there is a GPS fix we can scan the lin
    if (incoming_message[4] == 'G' && incoming_message[5] == 'A' && (incoming_m
        // Filter the minutes for the GGA line multiplied by 10
        lat_gps_actual = ((int)incoming_message[19] - 48) * (long)10000000;
        lat_gps_actual += ((int)incoming_message[20] - 48) * (long)1000000;
        lat_gps_actual += ((int)incoming_message[22] - 48) * (long)100000;
        lat_gps_actual += ((int)incoming_message[23] - 48) * (long)10000;
        lat_gps_actual += ((int)incoming_message[24] - 48) * (long)1000;
        lat_gps_actual += ((int)incoming_message[25] - 48) * (long)100;
        lat_gps_actual += ((int)incoming_message[26] - 48) * (long)10;
        // To convert the minutes to degrees we need to divide the minutes by 6
        lat_gps_actual /= (long)6;
        // Add the degrees multiplied by 10
        lat_gps_actual += ((int)incoming_message[17] - 48) * (long)100000000;
        lat_gps_actual += ((int)incoming_message[18] - 48) * (long)10000000;
        // Divide everything by 10
        lat_gps_actual /= 10;

        // Filter the minutes for the GGA line multiplied by 10
        lon_gps_actual = ((int)incoming_message[33] - 48) * (long)10000000;
        lon_gps_actual += ((int)incoming_message[34] - 48) * (long)1000000;
        lon_gps_actual += ((int)incoming_message[36] - 48) * (long)100000;
        lon_gps_actual += ((int)incoming_message[37] - 48) * (long)10000;
        lon_gps_actual += ((int)incoming_message[38] - 48) * (long)1000;
        lon_gps_actual += ((int)incoming_message[39] - 48) * (long)100;
        lon_gps_actual += ((int)incoming_message[40] - 48) * (long)10;
        // To convert the minutes to degrees we need to divide the minutes by 6
        lon_gps_actual /= (long)6;
        // Add the degrees multiplied by 10
        lon_gps_actual += ((int)incoming_message[30] - 48) * (long)1000000000;
        lon_gps_actual += ((int)incoming_message[31] - 48) * (long)100000000;
        lon_gps_actual += ((int)incoming_message[32] - 48) * (long)10000000;
        // Divide everything by 10
        lon_gps_actual /= 10;

        if (incoming_message[28] == 'N')
            // When flying north of the equator the latitude_north variable wil
            latitude_north = 1;
        else
            // When flying south of the equator the latitude_north variable wil
            latitude_north = 0;

        if (incoming_message[42] == 'E')
            // When flying east of the prime meridian the longiude_east variabl
            longiude_east = 1;
        else
            // When flying west of the prime meridian the longiude_east variabl
            longiude_east = 0;

        // Filter the number of satillites from the GGA line
        number_used_sats = ((int)incoming_message[46] - 48) * (long)10;
        number_used_sats += (int)incoming_message[47] - 48;
```

```
        if (lat_gps_previous == 0 && lon_gps_previous == 0) {
            // If this is the first time the GPS code is used
            // Set the lat_gps_previous variable to the lat_gps_actual variable
            lat_gps_previous = lat_gps_actual;
            // Set the lon_gps_previous variable to the lon_gps_actual variable
            lon_gps_previous = lon_gps_actual;
        }

        // Divide the difference between the new and previous latitude by ten
        lat_gps_loop_add = (float)(lat_gps_actual - lat_gps_previous) / 10.0;
        // Divide the difference between the new and previous longitude by ten
        lon_gps_loop_add = (float)(lon_gps_actual - lon_gps_previous) / 10.0;

        // Set the l_lat_gps variable to the previous latitude value
        l_lat_gps = lat_gps_previous;
        // Set the l_lon_gps variable to the previous longitude value
        l_lon_gps = lon_gps_previous;

        // Remember the new latitude value in the lat_gps_previous variable for
        lat_gps_previous = lat_gps_actual;
        // Remember the new longitude value in the lat_gps_previous variable fo
        lon_gps_previous = lon_gps_actual;
    }

    // If the line starts with SA and if there is a GPS fix we can scan the lin
    if (incoming_message[4] == 'S' && incoming_message[5] == 'A')
        fix_type = (int)incoming_message[9] - 48;

}
```

1.3. Установка текущей путевой точки

Далее, с полученными данными и происходит самая магия. Для включения поддержания текущего положения достаточно установить флаг `waypoint_set = 1;` и установить текущие координаты как waypoint:

```
l_lat_waypoint = l_lat_gps;
l_lon_waypoint = l_lon_gps;
```

После этого, начнётся вычисление ошибки в координатах и корректировка с помощью PD - регулятора. Для D - составляющей используется rotation memory.

1.4. Изменение точек (Чтобы летать по координатам)

Если просто задать новые `l_lat_waypoint` и `l_lon_wayoint`, находящиеся на большом удалении от дрона, он не сможет нормально прилететь и стабилизироваться на этих координатах. Для плавного изменения можно использовать переменные `l_lat_gps_float_adjust` и `l_lon_gps_float_adjust`. Это переменные типа `float`, изменяя которые, `l_lat_waypoint` и `l_lon_waypoint` будут плавно меняться.

Например, если в основном цикле постоянно прибавлять некую величину к этим переменным:

```
l_lat_gps_float_adjust += 0.0015;
```

При установленном waypoint, дрон будет плавно перемещаться в заданном направлении. В дальнейшим это будет использоваться для плавного ускорения и замедления во время движения к точке.

1.5. Удержание конкретной позиции

```
if (waypoint_set == 1) {
    //If the waypoints are stored

    // Adjust l_lat_waypoint
    if (l_lat_gps_float_adjust > 1) {
        l_lat_waypoint++;
        l_lat_gps_float_adjust--;
    }
    if (l_lat_gps_float_adjust < -1) {
        l_lat_waypoint--;
        l_lat_gps_float_adjust++;
    }

    // Adjust l_lon_waypoint
    if (l_lon_gps_float_adjust > 1) {
        l_lon_waypoint++;
        l_lon_gps_float_adjust--;
    }
    if (l_lon_gps_float_adjust < -1) {
        l_lon_waypoint--;
        l_lon_gps_float_adjust++;
    }

    // Calculate the latitude error between waypoint and actual position
    gps_lon_error = l_lon_waypoint - l_lon_gps;
    // Calculate the longitude error between waypoint and actual position
    gps_lat_error = l_lat_gps - l_lat_waypoint;

    // Subtract the current memory position to make room for the new value
    gps_lat_total_average -= gps_lat_rotating_mem[gps_rotating_mem_location];
    // Calculate the new change between the actual pressure and the previous me
    gps_lat_rotating_mem[gps_rotating_mem_location] = gps_lat_error - gps_lat_e
    // Add the new value to the long term average value
    gps_lat_total_average += gps_lat_rotating_mem[gps_rotating_mem_location];

    // Subtract the current memory position to make room for the new value
    gps_lon_total_average -= gps_lon_rotating_mem[gps_rotating_mem_location];
    // Calculate the new change between the actual pressure and the previous me
    gps_lon_rotating_mem[gps_rotating_mem_location] = gps_lon_error - gps_lon_e
    // Add the new value to the long term average value
    gps_lon_total_average += gps_lon_rotating_mem[gps_rotating_mem_location];

    // Increase the rotating memory location
    gps_rotating_mem_location++;
    if (gps_rotating_mem_location == 35)
        // Start at 0 when the memory location 35 is reached
        gps_rotating_mem_location = 0;

    // Remember the error for the next loop
    gps_lat_error_previous = gps_lat_error;
    gps_lon_error_previous = gps_lon_error;

    //Calculate the GPS pitch and roll correction as if the nose of the multico
    //The Proportional part = (float)gps_lat_error * gps_p_gain.
    //The Derivative part = (float)gps_lat_total_average * gps_d_gain.
    gps_pitch_adjust_north = (float)gps_lat_error * gps_p_gain + (float)gps_lat
    gps_roll_adjust_north = (float)gps_lon_error * gps_p_gain + (float)gps_lon_

    if (!latitude_north)
        // Invert the pitch adjustmet because the quadcopter is flying south of
        gps_pitch_adjust_north *= -1;
    if (!longiude_east)
        // Invert the roll adjustmet because the quadcopter is flying west of t
        gps_roll_adjust_north *= -1;
```

```
    //Because the correction is calculated as if the nose was facing north, we
    gps_roll_adjust = ((float)gps_roll_adjust_north * cos(angle_yaw * 0.017453
    gps_pitch_adjust = ((float)gps_pitch_adjust_north * cos(angle_yaw * 0.0174

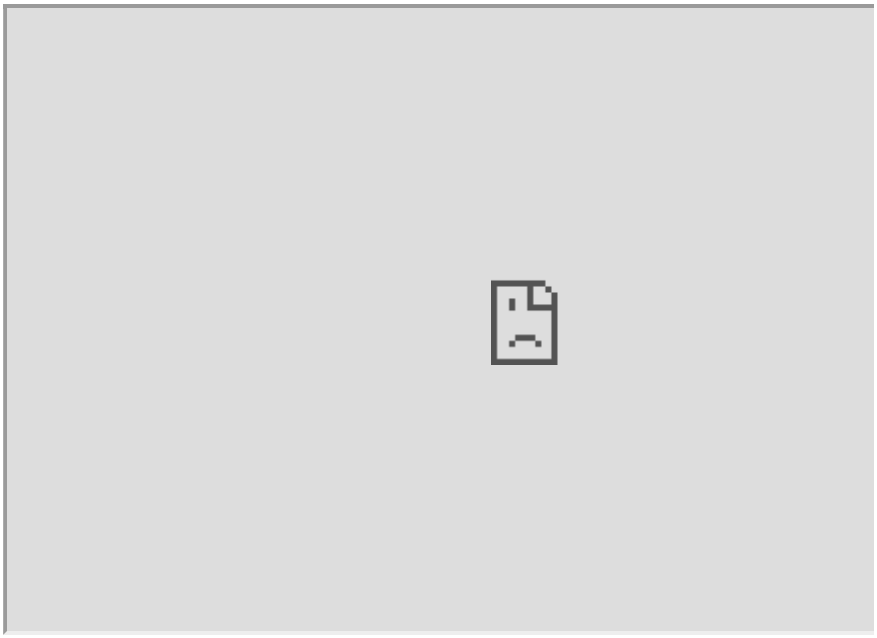
    //Limit the maximum correction to 300. This way we still have full control
    if (gps_roll_adjust > 300) gps_roll_adjust = 300;
    if (gps_roll_adjust < -300) gps_roll_adjust = -300;
    if (gps_pitch_adjust > 300) gps_pitch_adjust = 300;
    if (gps_pitch_adjust < -300) gps_pitch_adjust = -300;
}
```

2. Следование за платформой

Основной задачей стабилизации по GPS-координатам стала разработка алгоритма предсказания положения дрона. Самым простым способом представилось использовать математический расчет следующего положения дрона. Это вычисляется для наиболее точного позиционирования дрона в отношении посадочной платформы.

Для начала был придуман простейший алгоритм расчета коэффициента изменения координат. Реализация производилась на языке Python. На этапе тестирования данного алгоритма встала проблема симуляции генерации GPS-координат. Дабы разрешить эту проблему, было испробовано много различных ресурсов: от открытых исходных кодов самодельных навигаторов до попытки использовать API Google Maps, Yandex Maps или 2GIS. И лишь спустя семестр мы додумались до простого изменения значений по некоторой дельте с отрисовкой в MatPlotLib либо PyQtGraph. До этого всё тестирование алгоритма производилось с использованием инструментария прошивки PX4, симулятора движения дрона Gazebo. Как следствие было преодолено много формальностей в вопросах общения с симулятором и увеличением производительности.

Видео работы алгоритма предсказания GPS координат (кликабельно)



Конечный результат ошибки предсказанных координат достиг диапазона от 0 до 70 см.

---

3. Компас

До момента оптической стабилизации (во время GPS стабилизации), для вычисления вектора корректировки по GPS требуется знать точный угол по компасу. Для этого используется встроенный в GPS модуль компас.

Т.к. во время полёта меняются углы крена и тангажа, требуется корректировать значения с компаса. В общем плане, вычисления угла с компаса выглядит так:

```
// The compass values change when the roll and pitch angle of the quadcopter ch
// The 0.0174533 value is phi/180 as the functions are in radians in stead of d
compass_x_horizontal = (float)compass_x * cos(angle_pitch * -0.0174533) + (floa
compass_y_horizontal = (float)compass_y * cos(angle_roll * 0.0174533) + (float)

// Now that the horizontal values are known the heading can be calculated. With
// Please note that the atan2 uses radians in stead of degrees. That is why the
if (compass_y_horizontal < 0)actual_compass_heading = 180 + (180 + ((atan2(comp
else actual_compass_heading = (atan2(compass_y_horizontal, compass_x_horizontal

// Add the declination to the magnetic compass heading to get the geographic no
actual_compass_heading += declination;
// If the compass heading becomes smaller then 0, 360 is added to keep it in th
if (actual_compass_heading < 0) actual_compass_heading += 360;
// If the compass heading becomes larger then 360, 360 is subtracted to keep it
else if (actual_compass_heading >= 360) actual_compass_heading -= 360;
```

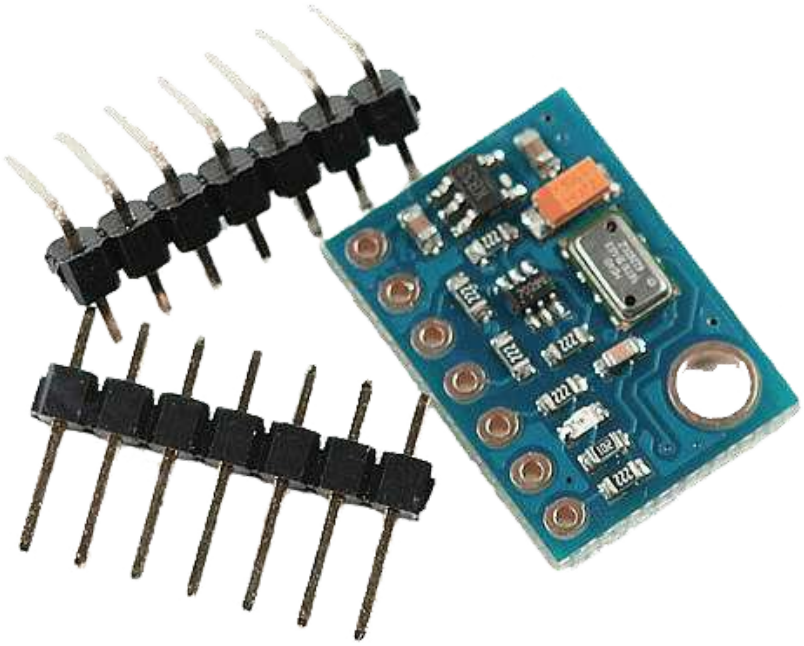
Понятно, что угол с компаса можно использовать и для поддержания угла рыскания дрона в целом. При полётах по точкам это, возможно, будет реализовано. Но данный момент, острой необходимости в этом нет, т.к. после начала оптической стабилизации, её алгоритм способен корректировать дрон независимо от его угла рыскания. Также, во время оптической стабилизации угол автоматически корректируется.

---

4. Удержание высоты (барометр)

До момента оптической стабилизации (во время GPS стабилизации), наш модуль Liberty-Link будет иметь возможность поддержания высоты при помощи барометра.

На платформе, так же, как и в Liberty-Link будут установлены следующие барометры MS5611.



Согласно документации, разрешение по высоте составляет 10см. Алгоритм будет брать значения давления и, пропуская его через ПИД-регулятор, стабилизировать высоту дрона, изменяя Throttle (3-ий канал).

Видео работы алгоритма удержания высоты по барометру (кликабельно):

Во время полёта по точкам, setpoint давления будет уменьшаться, для повышения высоты (лететь по прямой безопаснее на большой высоте, чтобы ни во что не врезаться). А во время стабилизации по GPS (когда дрон

находится уже близко к платформе), дрону будет задан setpoint по давлению такой, чтобы соответствовать ~1.5-2м высоты над платформой.

5. Оптическая стабилизация

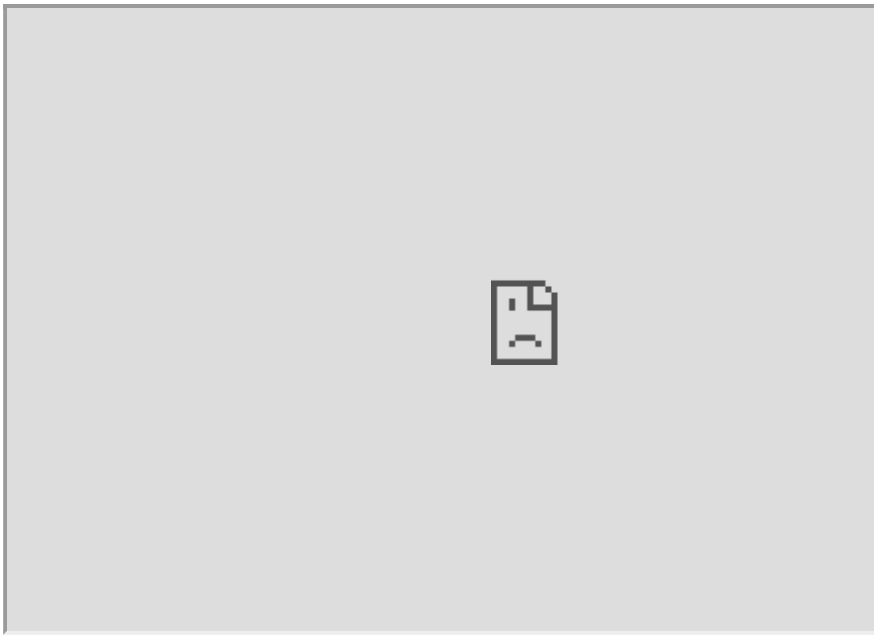
5.1. Так сложно и так важно

Оптическая стабилизация - самая важная и сложная часть нашего проекта. Только благодаря этим алгоритмам в наших условиях возможно достаточно точно удерживать дрон над платформой. Текущая версия алгоритма оптической стабилизации вместе с описанием для повторения, доступна в нашем основном репозитории на GitHub. В дальнейшем, к ней добавится и стабилизация по GPS.

5.2. Первые шаги

Так как мы не знали насколько реальным окажется выполнение этой далеко не простой задачи, первое что мы сделали, это определились, с помощью чего возможно точно стабилизировать дрон в пространстве. И тут, практически единогласно победил вариант с оптической стабилизации при помощи меток дополненной реальности. Во первых, это достаточно бюджетно, не нужны дорогостоящие системы GPS RTK, а, во вторых, дает требуемую точность. Одной из самых первых идей - было приделать Raspberry Pi к дрону, как это сделано на платформе Клевер, и стабилизироваться по метке на платформе.

Тест прототипа первой оптической стабилизации (кликабельно):



Но, проведя пару тестов от этой идеи мы быстро отказались. Для начала, Raspberry Pi очень слабая для быстрого вычисления такого объема данных, во вторых, сама идея установки на каждый дрон компьютера выглядит

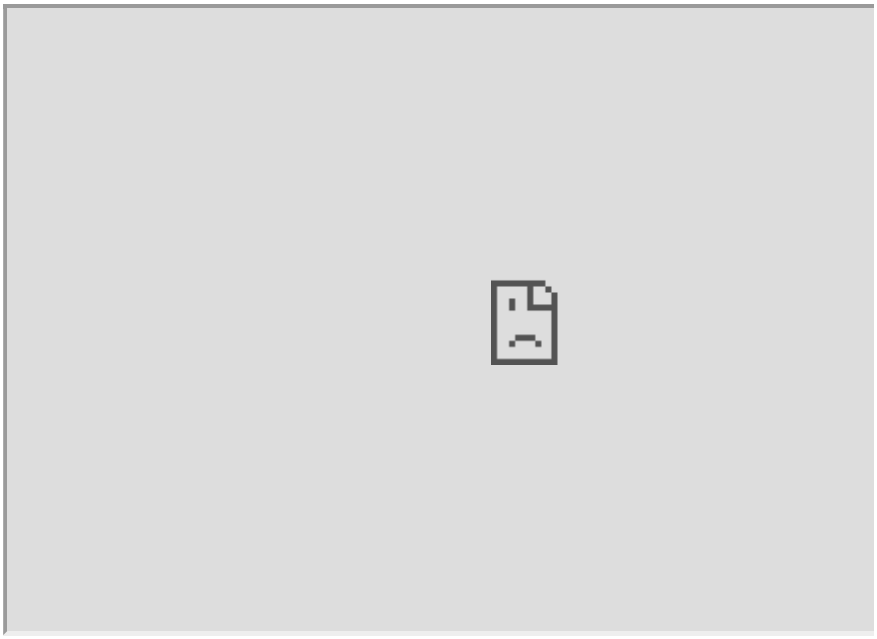
нерациональной.

Также, у нас были промежуточные прототипы, например, попытки использовать цветовые маркеры (окружности различных цветов), но эти идеи не оказались достаточно работоспособными.

5.3. Инверсия

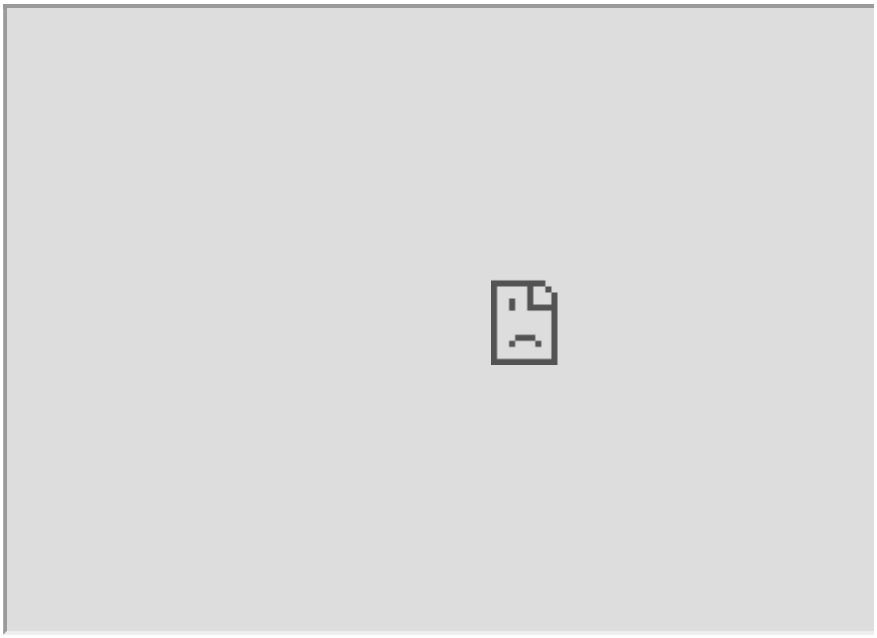
Так мы и пришли к текущему виду оптической стабилизации, когда камера с мощным компьютером расположены на платформе, а на дроне лишь ArUco 4x4 метка и модуль, управляющий им.

Самые первые тесты, в этом примере даже нет оценки положения маркера (pose estimation)(кликабельно):



Далее, были внедрены алгоритмы Pose Estimation благодаря библиотеке OpenCV. Первые тесты показали что мы на верном пути!

Pose Estimation Pyhton (кликабельно):

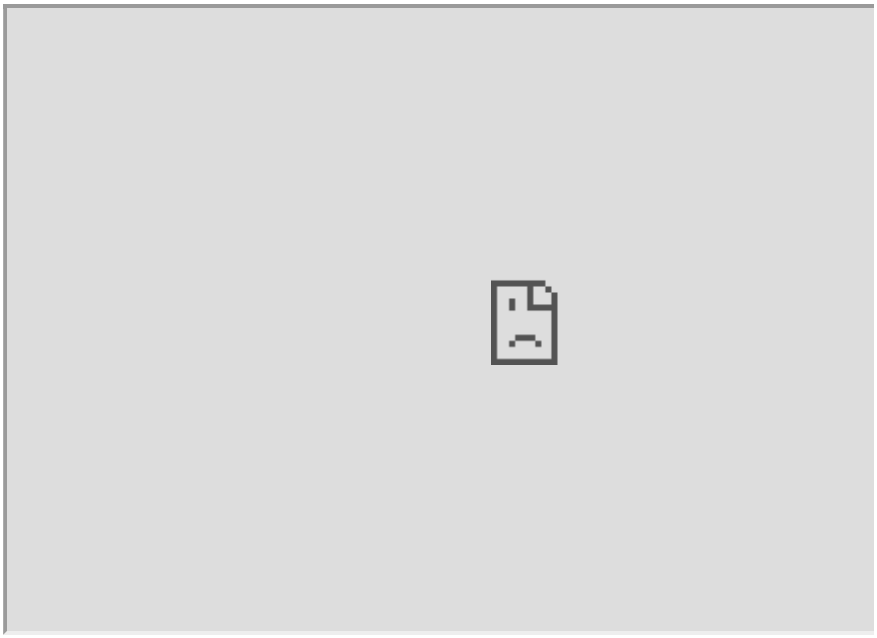


Но, по прежнему, алгоритмы были далеки от идеала. Например, т.к. код писался на Python (https://github.com/XxOinvizioNxX/Liberty-X_Point), производительность была не велика, также, не было нормального контроля потоков. Поэтому, пришлось что-то менять.

5.4. Версия на Java

Взвесив все ЗА и ПРОТИВ, было решено переписать всю оптическую стабилизацию на Java. Так и появилась первая версия Liberty-Way. На этот раз было решено подойти к ООП основательно, и, после небольшой настройки получился отличный алгоритм стабилизации и посадки.

Тест посадки на Liberty-Way v.beta_0.0.1 (кликабельно):

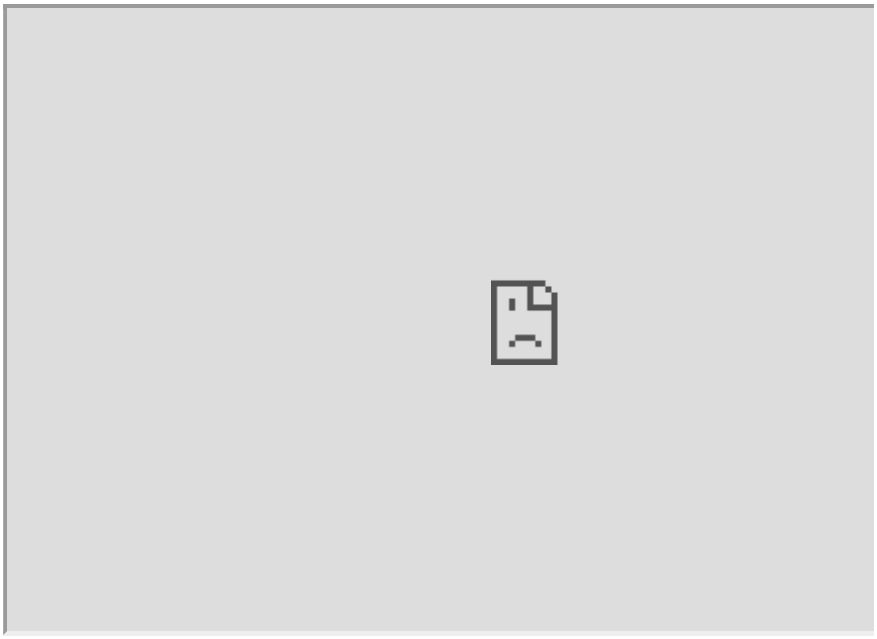


5.5. Liberty-Way

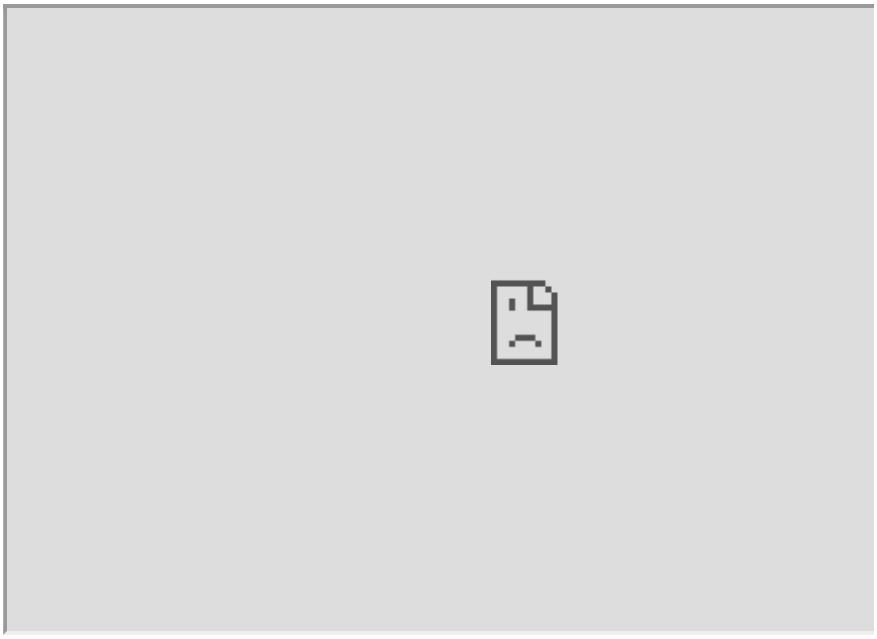
Далее последовало много доработок и исправлений ошибок. В результате, Liberty-Way представляет собой кроссплатформенное приложение, управляющееся через веб сарвар, что очень удобно для настройки и отладки. Также, в последних версиях (beta_1.0.3 - beta_1.1.2) был внедрён blackbox (для записи логов), а также общение с платформой и много других необходимых алгоритмов.

Полное описание, включая все настройки, запуск и т.д. вы можете найти в нашем репозитории на GitHub: https://github.com/XxOinvizioNxX/Liberty-Way.

Видео работы статичной стабилизации (кликабельно):

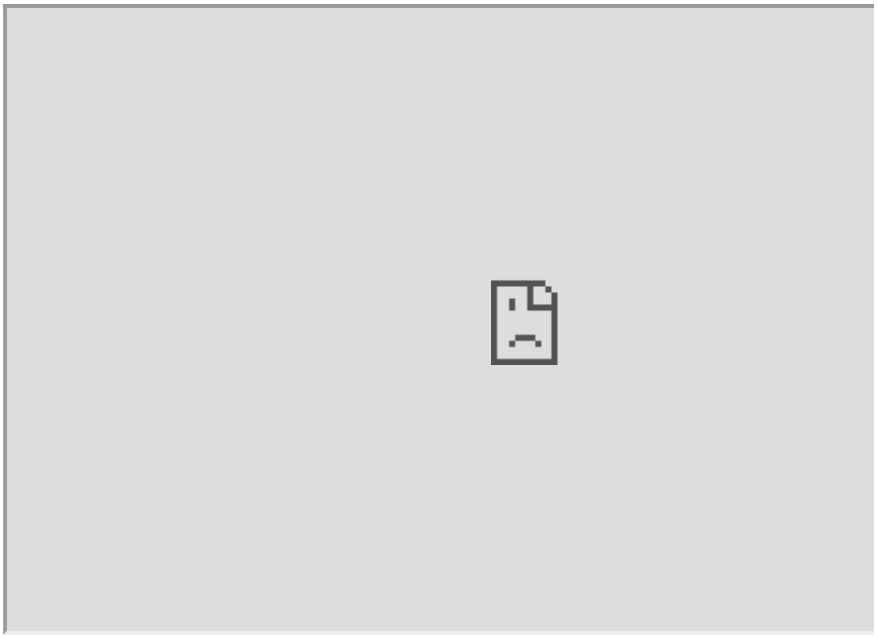


Liberty-Way может даже стабилизировать "брошенный" дрон (кликабельно):



Да, на видео есть небольшой баг с поворотом, в новом релизе он исправлен

И, конечно же, работа в движении (тестировалось ещё на beta_0.0.3) (кликабельно):



Все основные настройки удобно вынесены в отельный JSON-файлы (settings, PID), что позволяет без пересборки приложения быстро менять нужные параметры. Фактически, для запуска приложения, достаточно скачать последний релиз, распаковать архив и запустить через соответствующий вашей ОС лаунчер.

5.6. Связь с дроном

Liberty-Way подключается к модулю Liberty-Link, установленному на дроне, и, корректирует его положение, управляя напрямую первыми четырьмя основными каналами пульта. За один цикл (каждый фрейм с камеры) на модуль отправляются 12 байт данных корректировки:

Byte N	0	1	2	3	4	5	6	7	8	9	10	11
Description	Roll byte 1	Roll byte 2	Pitch byte 1	Pitch byte 2	Yaw byte 1	Yaw byte 2	Altitude byte 1	Altitude byte 2	Service Info	Check byte	Data suffix 1 (L)	Data suffix 2 (X)
Structure	1000-2000		1000-2000		1000-2000		1000-2000		0-3	XOR check sum	L	X
Example in HEX	0x05	0xDC	0x05	0xDC	0x05	0xDC	0x05	0xDC	0x00	0x00	0x4C	0x58
Example in DEC	1500		1500		1500		1500		0	0	76	88

- **Roll bytes** - Корректировка крена (1000-2000)
- **Pitch bytes** - Корректировка тангажа (1000-2000)
- **Yaw bytes** - Корректировка рыскания (1000-2000)
- **Altitude bytes** - Корректировка газа (высоты)
- **Service info** - Состояние дрона (0 - Корректировка отключена, 1 - Стабилизация, 2 - Снижение по барометру (команда не внедрена), 3 - Отключение моторов)
- **Check byte** - XOR чек-сумма
- **Data suffix** - уникальная пара ASCII символов, чтобы обозначить конец пакета

На стороне дрона (модуля Liberty-Link), чтение данных происходит следующим образом:

```
while (Telemetry_serial.available()) {
    tdc_receive_buffer[tdc_receive_buffer_counter] = Telemetry_serial.read();
    if (tdc_receive_byte_previous == 'L' && tdc_receive_buffer[tdc_receive_buf
        tdc_receive_buffer_counter = 0;
        if (tdc_receive_start_detect >= 2) {
            tdc_check_byte = 0;
            for (tdc_temp_byte = 0; tdc_temp_byte <= 8; tdc_temp_byte++)
                tdc_check_byte ^= tdc_receive_buffer[tdc_temp_byte];
            if (tdc_check_byte == tdc_receive_buffer[9]) {
                direct_roll_control = (uint32_t)tdc_receive_buffer[1] | (uint32
                direct_pitch_control = (uint32_t)tdc_receive_buffer[3] | (uint3
                direct_yaw_control = (uint32_t)tdc_receive_buffer[5] | (uint32_
                direct_throttle_control = (uint32_t)tdc_receive_buffer[7] | (ui
                direct_service_info = (uint32_t)tdc_receive_buffer[8];

                if (direct_roll_control > 1100 && direct_roll_control < 1900 &&
                    direct_pitch_control > 1100 && direct_pitch_control < 1900
                    direct_yaw_control > 1100 && direct_yaw_control < 1900 &&
                    direct_throttle_control > 1100 && direct_throttle_control <
                    /*flight_mode == 2 &&*/ channel_7 > 1500) {
                    tdc_timer = millis();
                    tdc_working = 1;
                }
                else
                    tdc_working = 0;
            }
            else {
                direct_roll_control = 1500;
                direct_pitch_control = 1500;
                tdc_working = 0;
            }
        } else
            tdc_receive_start_detect++;
    }
    else {
        tdc_receive_byte_previous = tdc_receive_buffer[tdc_receive_buffer_count
        tdc_receive_buffer_counter++;
        if (tdc_receive_buffer_counter > 11)tdc_receive_buffer_counter = 0;
    }
}
if (millis() - tdc_timer >= 500) {
    tdc_working = 0;
}
if (tdc_working && direct_service_info == 2 && !return_to_home_step)
    return_to_home_step = 3;
if (!tdc_working)
    return_to_home_step = 0;
if (!tdc_working || direct_service_info < 1) {
    direct_roll_control = 1500;
    direct_pitch_control = 1500;
    direct_yaw_control = 1500;
    direct_throttle_control = 1500;
}
```

В результате, имеются 4 переменные:

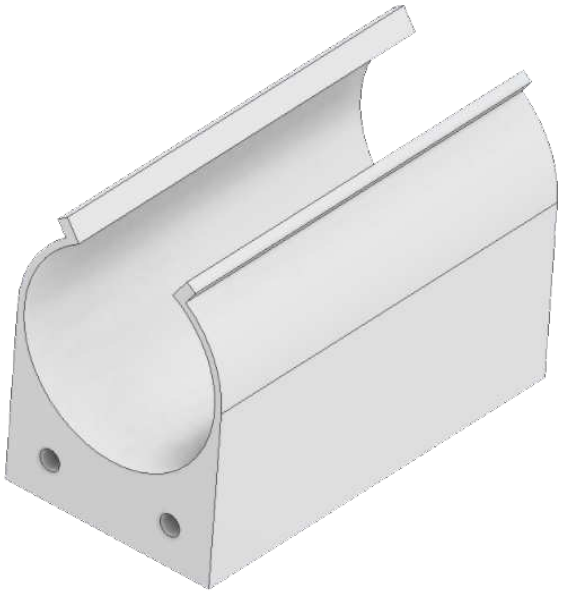
```
direct_roll_control
direct_pitch_control
direct_yaw_control
direct_throttle_control
```

Которые напрямую прибавляются к данным, поступающим с пульта управления. Вероятно, в дальнейшем, будут добавлены и другие данные, как минимум, для работы с GPS. Следите за обновлениями в нашем репозитории.

5.7. Подвес для камеры

Для эксплуатации нашей системы в реальных условиях, требуется минимизировать тряску камеры, чтобы не потерять метку на дроне. Для этого, была разработана 3Д-модель крепления подвеса от дрона к нашей платформе для стабилизации обычной веб камеры

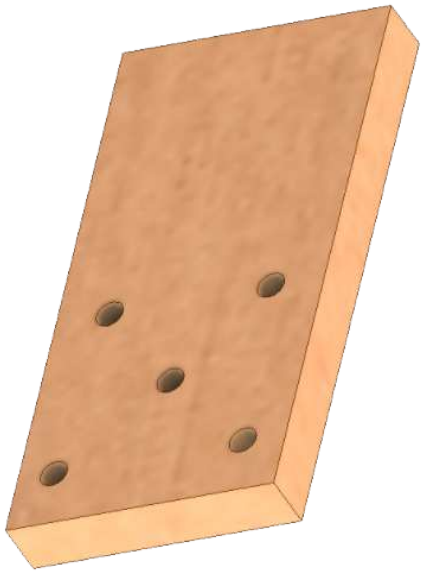
Крепление для камеры:



Крепление провода (ферритового фильтра на проводе) камеры:



Крепление защёлок "крабиков" на подложку подвеса:



Примерный вид сборки всего механизма подвеса:



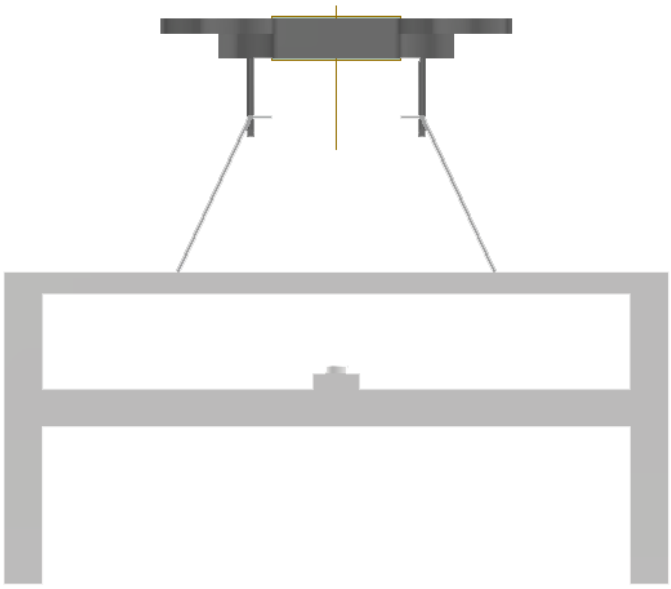
6. Платформа Eitude

Платформа - взаимосвязанная система для посадки дрона. Управляться платформа планируется через Serial-интерфейс, с помощью G-Code команд: Текущий код платформы можно в репозитории Eitude на GitHub: https://github.com/XxOinvizioNxX/Eitude.

6.1. Система захватов

Но ведь логично, что в реальности, без системы захватов невозможно посадить дрон, не повредив никого. Пока что, у нас есть прототип в виде 3Д модели. Для реализации захвата дрона будут изготовлены 4 длинных захвата с крючками на концах и по мере посадки коптера на платформу, шасси будут захватываться этими 4-мя крючками и удерживаться по мере снижения и после него.

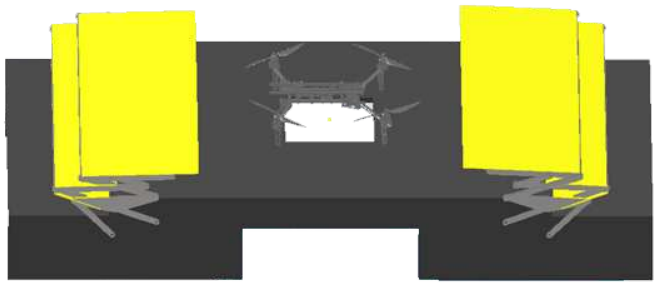


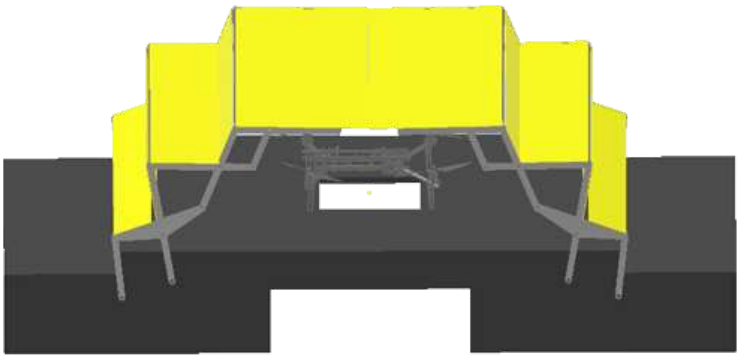


6.2. Крыша

Для защиты от неблагоприятных погодных условий, мы разработали механизм крыши - ножничные механизмы, обтянутые брезентом, которые находятся на краях платформы и после успешной посадки механизмы с обеих сторон платформы будут закрываться и защищать дрон от внешнего воздействия, Сама конструкция крыши делает её достаточно лёгкой и

прочной, а ножничный механизм позволяет просто складывается и раскладывается при этом сборка такого механизма будет простой и надежной.





6.3. Спидометр

Для будущей посадки на быстро движущуюся платформу, очень полезно знать скорость её движения. На данный момент на платформе нет GPS модуля, или иных способов измерить абсолютную скорость. Поэтому, для временного решения этой проблемы, решено было вычислять скорость по ускорению, используя акселерометр. Для примера, MPU6050. IMU-модуль через мягкую подложку установлен на прототип платформы и прикрыт крышкой для защиты от ветра. Алгоритм стабилизации (Liberty-Way) посылает на платформу запрос `L1` для проверки скорости. В качестве ответа возвращается `S0 L<скорость в км/ч>` .



Тест спидометра (внутри серого круга нижний правый параметр (SPD) - скорость в км/ч) (кликабельно):

Для вычисления скорости, берётся ускорение за маленькие промежутки времени, перемножается со временем, получая моментальную скорость. Которая постоянно прибавляется к предыдущей сумме:

```
void speed_handler(void) {
    gyro_signalen();

    // Filter accelerations
    acc_x_filtered = (float)acc_x_filtered * ACC_FILTER_KOEFF + (float)acc_x *

    speed = acc_x_filtered;
    // Convert acceleration to G
    speed /= 4096.0;
    // Convert to m/s^2
    speed *= 9.81;
    // Multiply by dt to get instant speed in m/ms
    speed *= (millis() - speed_loop_timer);

    // Reset timer
    speed_loop_timer = millis();

    // Convert to m/s
    speed /= 1000.0;
    // Convert to km/h
    speed *= 3.6;

    // Accumulate instatnt speed
    speed_accumulator += speed;

    if (!in_move_flag) {
        // If the platform is not moving, reset the speed
        speed_accumulator = speed_accumulator * SPEED_ZEROING_FACTOR;
    }
}
```

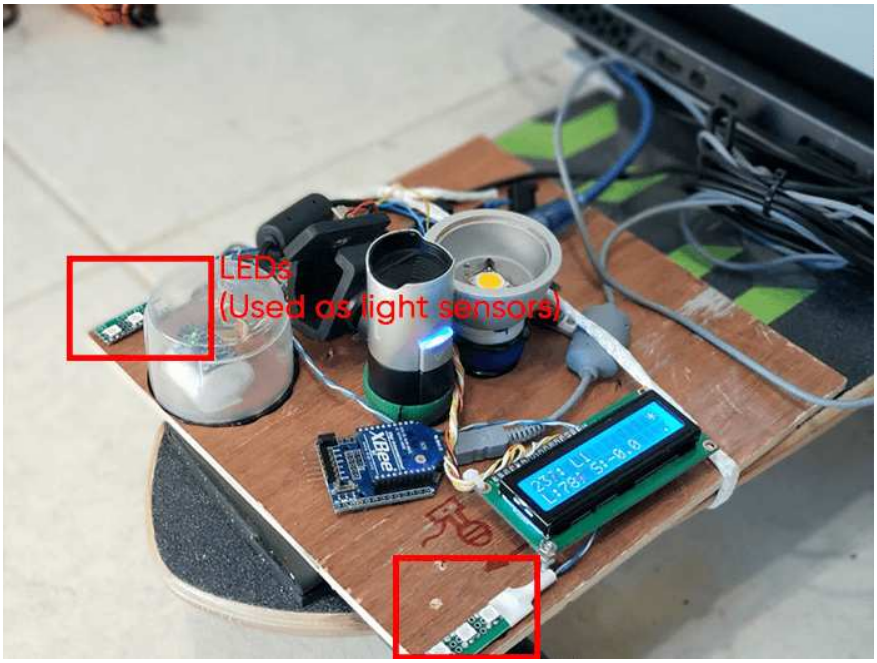
Несмотря на наличие различных фильтров, из-за погрешности скорость может не "вернуться" в 0, поэтому, также производится замер вибраций, и, если они меньше порога, считается что платформа стоит и постепенно обнуляется скорость.

Полный код спидометра можно найти в репозитории Eitude на GitHub: https://github.com/XxOinvizioNxX/Eitude

6.4. Датчики уровня освещённости

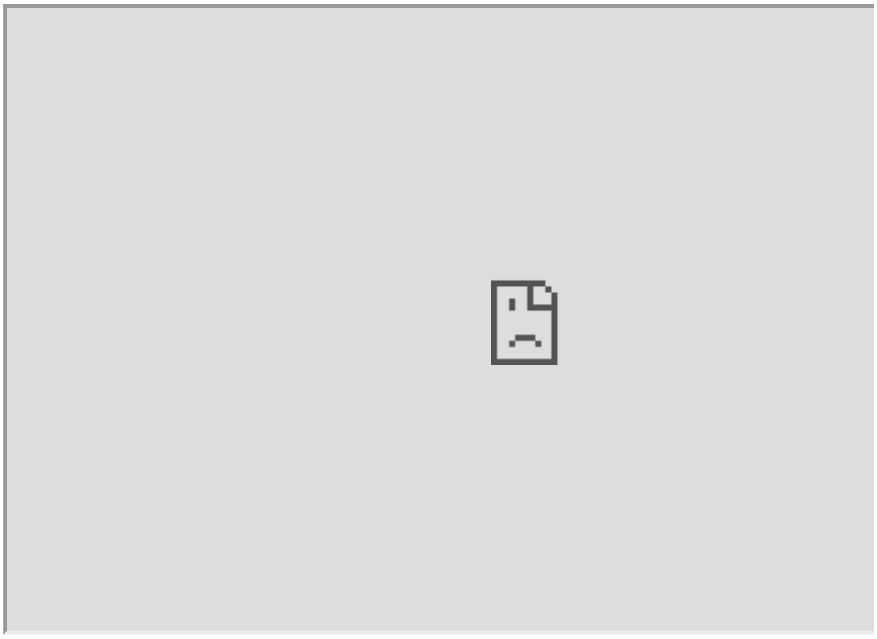
Т.к. наша платформа должна работать в различных окружающих условиях, а оптическая стабилизация очень требовательна к видимости ArUco маркера, важно иметь автоматическую систему измерения выдержки камеры по уровню освещённости, а, при её нехватке, даже включать дополнительную подсветку. В долгосрочной перспективе в качестве датчиков света планируется использовать специализированные сенсоры, например, BH1750.

В текущем варианте прототипа, используются 6 светодиодов в качестве датчика света и, встроенный в микроконтроллер, АЦП. Алгоритм стабилизации (Liberty-Way) посылает на платформу запрос `L0` для проверки уровня освещённости. В качестве ответа возвращается `S0 L<освещённость>`.



Тест определения уровня освещённости с помощью светодиодов (кликабельно):

Тест регулировки выдержки и включения дополнительной подсветки (кликабельно):



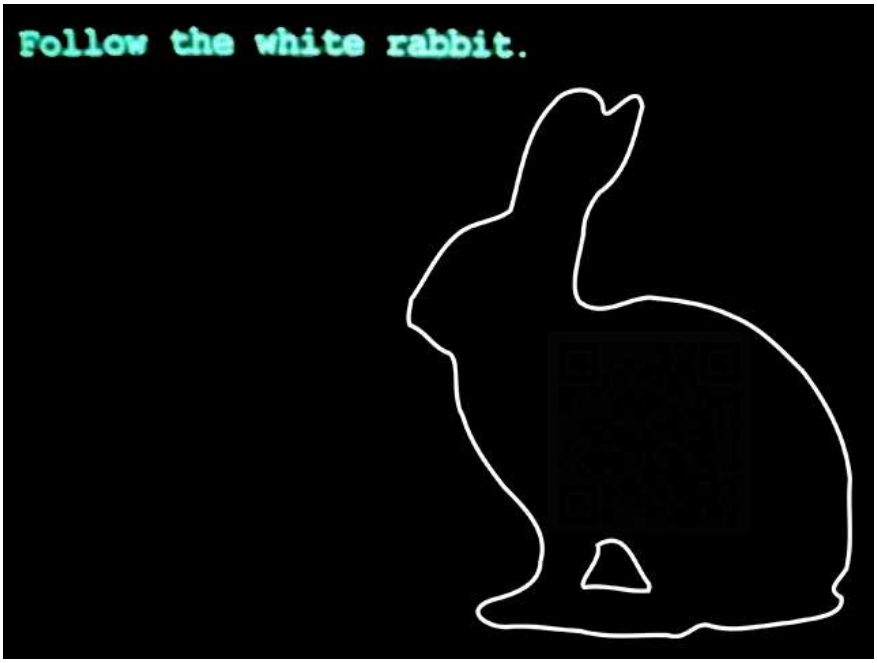
7. Заключение

На данный момент, имеется отлаженный прототип оптической стабилизации, GPS удержания, стабилизации высоты по барометру, платформы и множество 3Д-моделей, жаждущих реализации. Проект автоматической посадки дрона на движущуюся платформу ещё не закончен.

Следите за нашими апдейтами:

- На репозитории GitHub: https://github.com/XxOinvizioNxX/Liberty-Way.
- И на нашем YouTube-канале: https://www.youtube.com/channel/UCqN12Jzy-1eJLkcA32R0jdg.

В дальнейшем, мы планируем сделать ещё много нового и интересного!


Follow the white rabbit.

Разработка системы для управления БПЛА с помощью шлема виртуальной реальности

CopterHack-2021, команда: **ProCleVeR**.

Команда

- Давыденко Галина, e-mail: galyadavydenko@yandex.ru.

Введение

Сейчас существует несколько способов управления квадрокоптером: первый и самый простой, управление через аппаратуру, у данного метода имеется несколько недостатков, управление идет до тех пор, пока человек может видеть квадрокоптер, или же пока не будет потерян сигнал. Второй способ – FPV, такое управление уже более удобно и наиболее распространено, по сравнению с предыдущим. В данном случае осуществляется не только управление коптером, но и также получения видео-изображения по дополнительному видео-каналу в режиме реального времени. Третий способ, автономный полет - позволяет БПЛА работать в среде, куда не проникает сигнал GPS и без оператора.

Рассмотрев все способы управления, я выявила, что похожим на систему, которую я разрабатываю, будет FPV. В моей разработке присутствует несколько компонентов, квадрокоптер, шлем виртуальной реальности и манипуляторы. И сравнив, то что я хочу получить в итоге и то что есть на данный момент, выявилось, что у моей разработки будут преимущества, например, как я считаю главным минусом FPV управления является то, что коптер не сможет летать на большие расстояния из-за сигнала аппаратуры.

Разработка

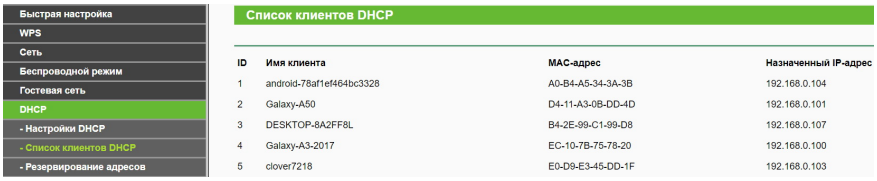
Было принято решение делать систему такой: управление квадрокоптером будет проходить через манипуляторы, а также через шлем виртуальной реальности. Какое же управление идет через шлем? На шлем будет перенесен поворот квадрокоптера по рысканью. При помощи поворота головы, будет поворачиваться коптер.

Настройка Clover OS

Настройка включает в себя переключение Raspberry из режима точки доступа в режим клиента. На начало работы была установлена следующая операционная система. После установки можно было приступить непосредственно к настройкам системы. Как перевести Raspberry Pi в режим клиента, рассказывается в статье: Настройка Wi-Fi. После того, как была произведена данная настройка Raspberry будет автоматически подключаться к Wi-Fi, после можно подключаться к Raspberry по SSH, также в дальнейшем подключение к Wi-Fi пригодится для подключения к серверу и передачи данных между клиентом и серверу (в разрабатываемой мной системе клиентом является квадрокоптер и сервером – компьютер).

Подключение и проверка подключения

Для начала проверим и попробуем подключиться к Raspberry по сети Wi-Fi. Узнать подключается ли Raspberry, а также узнать его IP-address для дальнейшей работы. Подключаемся к маршрутизатору по локальному адресу 192.168.0.1, затем переходим к списку подключённых устройств и находим устройство с названием: cloverXXXX, где X – любое число.



Удаленное управление

Для дальнейшей работы, будет проходить несколько тестов. Первые два теста используют светодиоды. 1 – отправление и получение данных с проверкой на светодиоде. Данные отправляются с сервера и приходят клиенту, клиент также отправляет данные о своем состоянии серверу.

Для начала подключим библиотеку, используемую при работе со светодиодами – RPi.GPIO. Затем поле того как было получено сообщение от сервера, включаем(выключаем) светодиод, в строке мы будем указывать порт, к которому подключен светодиод, а также значение 1 или 0 в зависимости от того, что требуется сделать со светодиодом. Познакомится с программным управление светодиода можно познакомится здесь.

Далее рассмотрим вариант управлении непосредственно через среду Unity, которая и использовалась при разработке системы. Для данного теста было написано два кода один из низ написан на C# и является сервером в данном подключении, другой на Python – клиент.

Откроем соединение и подключимся к клиенту по протоколу TCP при помощи следующих строк:

```
IPEndPoint ipPoint = new IPEndPoint(IPAddress.Parse("192.168.0.107"), 9090);
socketServer = new Socket(AddressFamily.InterNetwork, SocketType.Stream, Protoc
socketServer.Bind(ipPoint);
```

Для прослушивания каналов используем метод:

```
socketServer.Listen(10);
```

Так как потребуется начать асинхронную операцию, создадим объект асинхронных событий.

```
SocketAsyncEventArgs  e = new  SocketAsyncEventArgs();
```

Для того, чтобы определить нажата ли клавиша, будем использовать следующее:

```
Input.GetKey(KeyCode.Space)
```

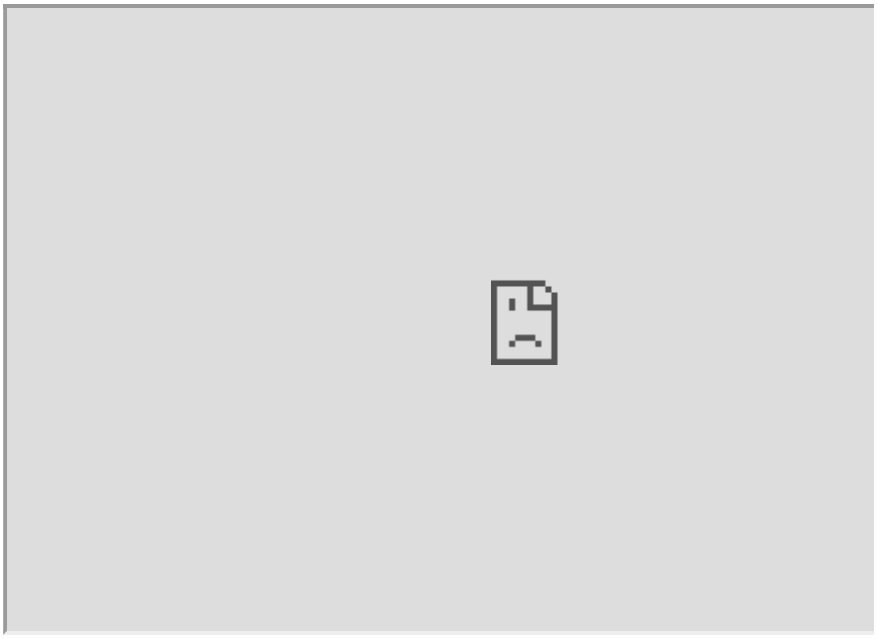
Для отправки используем:

```
socketClient.Send(Encoding.ASCII.GetBytes("1"));
```

Для принятия данных используем:

```
socketClient.Receive(Encoding.ASCII.GetBytes("1024"));
```

Видео демонстрации работоспособности результата:



Отправка изображения и передача видео в среду Unity

Для начала будем отправлять пакет данных, который содержит в себе информацию: тип передаваемых данных и если это изображение, то его размер. Это делается потому, что клиент(квадрокоптер) помимо изображения будет отправлять данные, например, местоположение, заряд аккумулятора, мощность и так далее. Для этого, было необходимо различать пакеты. В программе это реализуется следующим образом:

```
socketClient.Receive(buffer);
Array.Copy(buffer, 0, image, i, buffer.Length > size - i ? size - i : buffer.Le
```

Для вывода изображения используем:

```
Texture2D  tex = new  Texture2D(2, 2);
tex.LoadImage(image);
GetComponent<Renderer>().material.mainTexture = tex;
```

В свою очередь клиент отправляет изображение, которое предварительно загрузили на Raspberry.

Для определения размера передаваемого изображения используем:

```
filesize = os.stat(filename).st_size
```

Пакуем данные:

```
d = struct.pack('>bI', 0, filesize)
```

Также для отправки данных может использоваться другой метод:

```
s.sendall(bytes_read)
```

Перейдем к передаче видеопотока и его отображении:

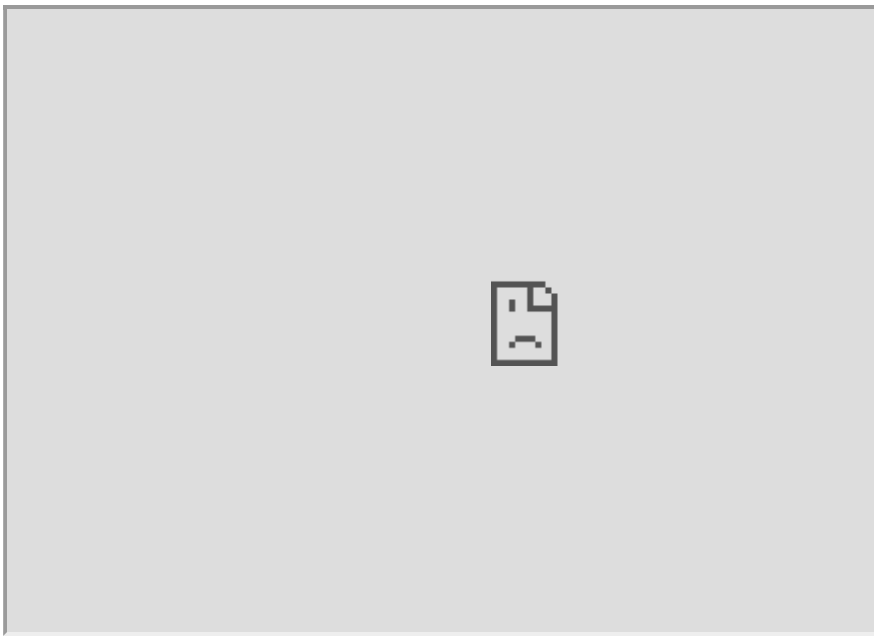
Для вывода напишем следующие строчки кода:

```
yield return new WaitWhile(() => socketClient.Available < size);
Debug.Log(socketClient.Available);
socketClient.Receive(image, 0, image.Length, SocketFlags.None);
Texture2D tex = new Texture2D(2, 2);
tex.LoadImage(image);
GetComponent<Renderer>().material.mainTexture = tex;
```

У клиента добавляем передачу видео с подключенной камеры:

```
ret, frame = cam.read() # считываем изображения с камеры
result, frame = cv2.imencode('.jpg', frame, encode_param) # записываем в переме
client_socket.send(struct.pack("<bI", 1, size))
client_socket.sendall(frame)
```

Видеодемонстрация работоспособности результата:



Функции при управлении

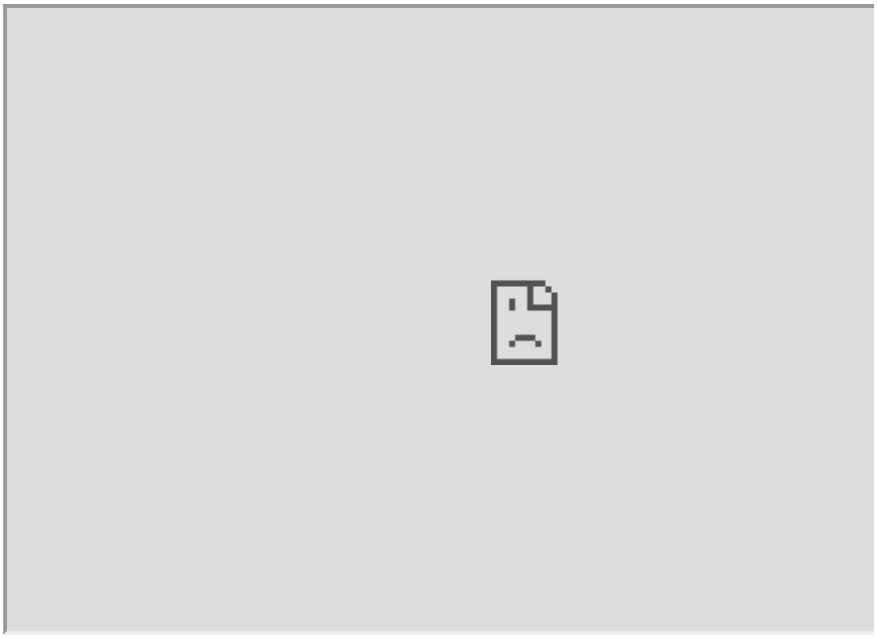
- Индикация управления.



- Поворот квадрокоптера по рысканью при помощи шлема виртуальной реальности.

Тестовые запуски системы

- Запуск без индикации.
- Запуск с индикацией.



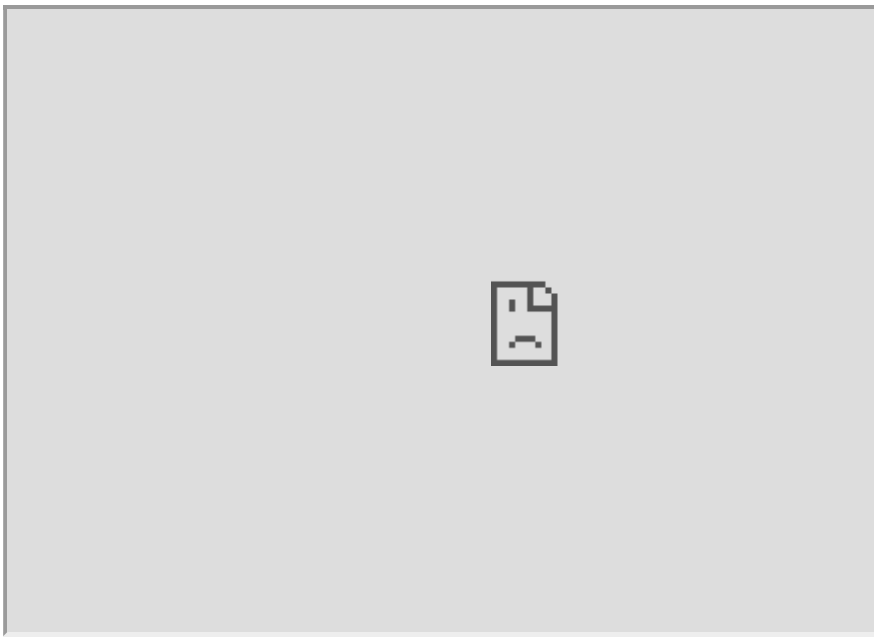
clever-show

Программное обеспечение для запуска шоу дронов под управлением Raspberry Pi с пакетом COEX Clover и полётного контроллера с прошивкой PX4.

Создайте анимацию в Blender, сконвертируйте её в полётные пути дронов, настройте дроны и запустите своё собственное шоу дронов!

Репозиторий проекта: https://github.com/CopterExpress/clever-show.

Демонстрационное видео



12 дронов выступают в Электротеатре Станиславский, Москва.

Разработчики

- Артур Голубцов, инженер программист COEX, https://github.com/goldarte
- Артём Васюник, стажёр программист COEX, https://github.com/artem30801

Innopolis Open 2020 - команда L22_ÆRO

Команда

- Юрьев Василий.
- Оконешников Дмитрий.

Описание задачи финала

Внедрение новых технологий происходит в различных отраслях экономики, в том числе и в сельском хозяйстве. Дроны или БПЛА не стали исключением. Благодаря применению беспилотников оценка состояния сельскохозяйственных территорий и анализ компонентов ландшафта стали более доступными и эффективными.



Финальная задача Innopolis Open 2020 была посвящена мониторингу сельскохозяйственных территорий и состояла из следующих элементов:

- Взлет (с QR-кода) и посадка (на цветной маркер небольшого размера).
- Распознавание зашифрованного сообщения в QR-коде.
- Распознавание цвета объектов (цветных маркеров – условное обозначение сельхоз угодий).
- Определение их координат (расположение на поле изменяется).

- Отчет по полученным данным.

Код

Код на GitHub: https://github.com/vas0x59/ior2020_uav_L22_AERO.

Основной код

При реализации кода в первоначальной концепции использовались свои типы сообщений, множество нод и других возможностей ROS, для обеспечения этого функционала необходимо создавать пакет и компилировать его, но из-за специфики соревнований (использование одной SD-карты для все команд) весь код был объединен в один файл. Данный подход усложнил отладку, но упростил запуск на площадке.

Элементы программы:

1. Взлет.
2. Распознавание QR-кода.
3. Поиск цветных маркеров.
4. Посадка.
5. Генерация отчета и видео.

Итоговыми координатами маркеров являются автоматически сгруппированные и усредненные данные из системы распознавания полученных за весь полет. Для покрытия всей территории была выбрана траектория "Зиг-заг". Для отладки применен симулятор Gazebo.

Цветные маркеры

`l22_aero_vision/src/color_r_c.py`

Для обработки изображения с камеры и детектирования объектов мы использовали функции из библиотеки OpenCV.

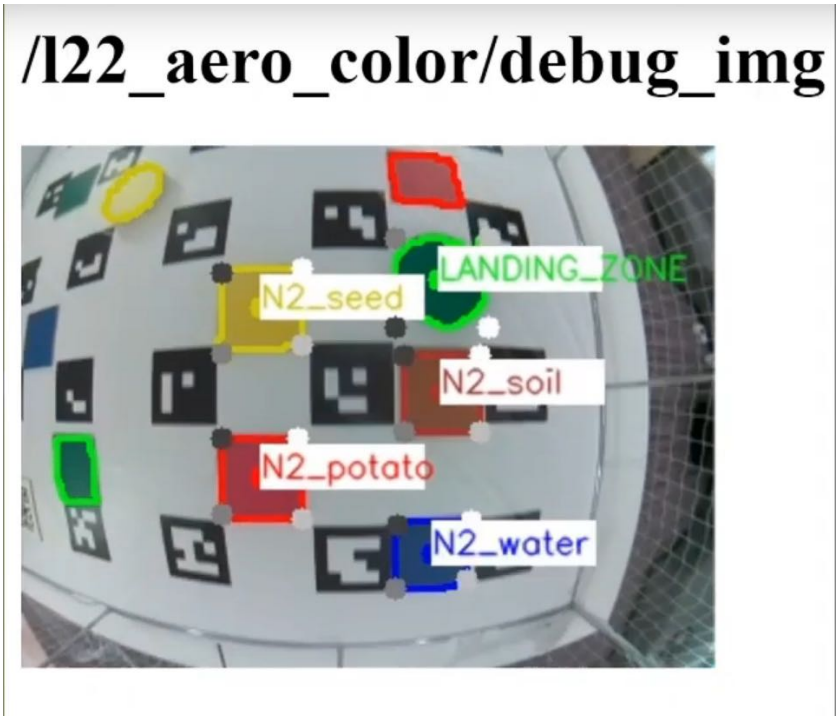
Алгоритм:

1. Получение изображения и параметров камеры.
2. Построение маски по определенному диапазону цветов (в формате HSV).
3. Детектирование контуров цветных объектов.
4. Определение типа объекта, получение ключевых точек объекта на изображении.
5. Определение положения квадратов и кругов с помощью solvePnP основываясь на реальных размерах объектов и точках на изображении (OpenCV Docs).
6. Отправка результата в топики `/l22_aero_color/markers` и `/l22_aero_color/circles` (координаты относительно `main_camera_optical`).

Во время разработки были созданы свои типы сообщений, а также сервис для настройки параметров детектора во время посадки. (`ColorMarker` , `ColorMarkerArray` , `SetParameters`).

Для определения положения цветных объектов в системе координат поля была использована библиотека TF (http://wiki.ros.org/tf)

Из-за искажений по краям изображения от fisheye-объектива все распознанные контуры находящийся рядом с краем изображения игнорируются. Во время посадки данный фильтр отключается. Определение типа объекта производиться с помощью функций анализа контуров (approxPolyDP - кол-во вершин; `minAreaRect` , `contourArea` - соотношение площади описанного квадрата и площади контура + соотношение сторон).



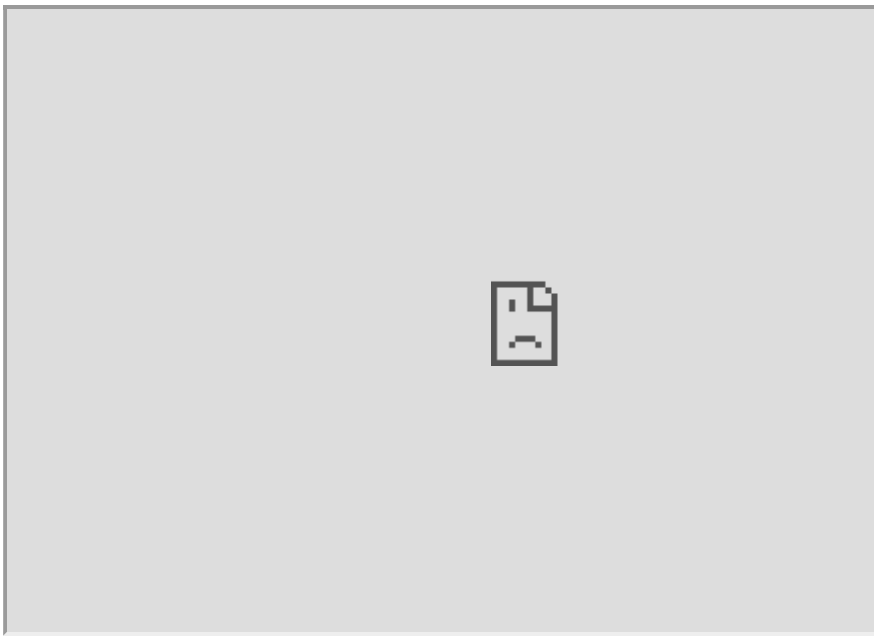
Примеры распознавание маркеров:



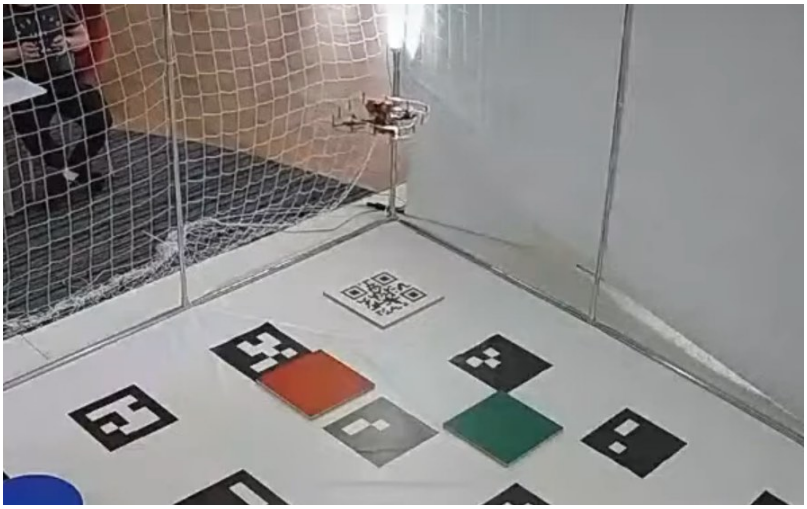
Визуализация в RViz

`l22_aero_vision/src/viz.py`

Для отладки распознавания объектов создан скрипт визуализирующий координаты маркеров в среде RViz.



QR-код



Для выполнения задачи распознавания QR-кода была использована библиотека PyZbar. С целью повышения результативности и точности распознавания QR-кода полеты производились на небольшой высоте по точкам, расположенным вокруг данного объекта.

Посадка

Посадка выполняется в 3 этапа:

1. Перелет к предполагаемой зоне посадки и зависание на высоте 1.5 м.
2. Спуск до высоты в 0.85 м с 3 корректировками по координатам маркера относительно `aruco_map`.
3. Спуск в течение нескольких секунд с постоянной корректировкой по координатам маркера посадки в системе координат `body` (так как ArUco-маркеры могут быть уже не видны), вместо `navigate` используется `set_position`.

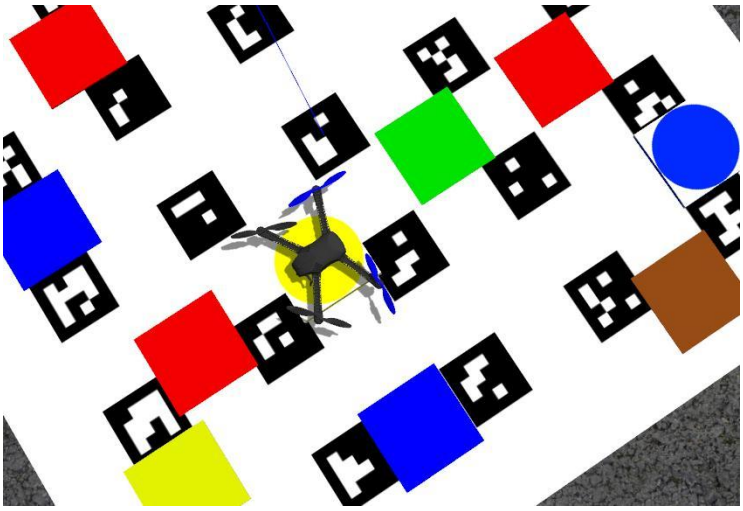
Gazebo

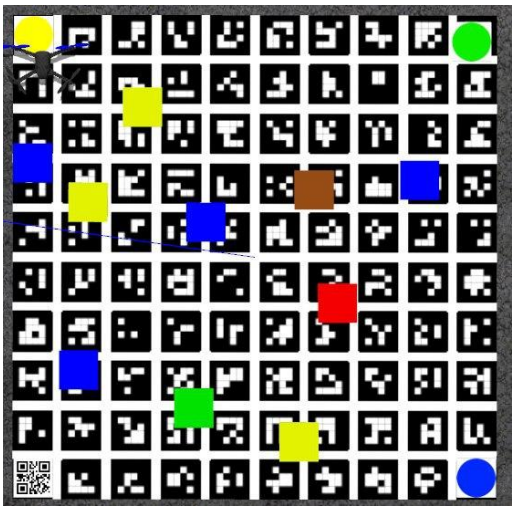
По причине отсутствия возможности тестирования кода на своем реальном дроне было принято решение воспользоваться симулятором Gazebo.

Для запуска пакета ПО Клевера в симуляторе можно использовать набор скриптов или оригинальную инструкцию от PX4.

Для Innopolis Open было создано несколько тестовых сцен. ior2020_uav_L22_AERO_sim.

Также использование симулятора ускорило отладку полного выполнения кода, так как запуск производился с real time factor=2.5.





При тестировании выявлены некоторые проблемы (некорректное положение `aruco_map`) с использованием дисторсии в плагине камеры, поэтому в симуляторе использовалась камера типа Pinhole (без искажений от объектива).

ROS

Созданные ноды, топики, сообщения и сервисы.

Nodes

- `l22_aero_vision/color_r_c.py` - распознавание цветных объектов.
- `l22_aero_vision/viz.py` - визуализация в RViz.
- `l22_aero_code/full_task.py` - основной код.

Topics

- `/l22_aero_color/markers` (`l22_aero_vision/ColorMarkerArray`) - список прямоугольных маркеров.
- `/l22_aero_color/circles` (`l22_aero_vision/ColorMarkerArray`) - список круглых маркеров.
- `/l22_aero_color/debug_img` (`sensor_msgs/Image`) - изображение для отладки.
- `/qr_debug` (`sensor_msgs/Image`) - изображение для отладки.

Messages

`ColorMarker`

```
string color
int16 cx_img
int16 cy_img
float32 cx_cam
float32 cy_cam
float32 cz_cam
float32 size1
float32 size2
int16 type
```

`ColorMarkerArray`

```
std_msgs/Header header
l22_aero_vision/ColorMarker[] markers
```

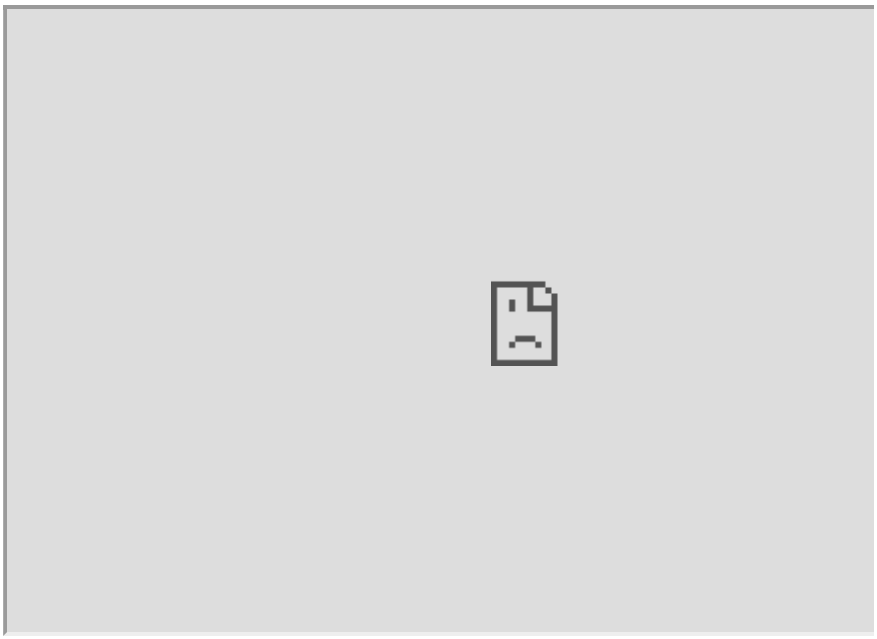
Services

`SetParameters`

```
float32 rect_s1
float32 rect_s2
float32 circle_r
int32 obj_s_th
int32 offset_w
int32 offset_h
---
```

Олимпиада НТИ 2020 – Летательная робототехника (P4DF2)

Решение задания командного зачёта на финале Олимпиады НТИ 2020 по треку Летательная робототехника командой P4DF2.



Авторы

Команда P4DF2:

- Игорь Сидорин (@maerans)
- Даниил Руфин (@Daniil_P4R)

Описание задачи финала

Глобальные эпидемии всегда застают человечество врасплох, и заставляют менять привычный образ жизни. Но в этот раз на борьбу с распространением вируса встали современные технологии! На фоне эпидемии в Китае начали использовать дроны, которые патрулируют улицы и отчитывают прохожих, которые гуляют, не надевая защитную маску, а также повсеместно доставляют медикаменты.

Пока тесты на наличие вируса только начинают массово распространяться, а вакцина от вируса находится в разработке – самое время задуматься о том, как быстро и безопасно производить распространение вакцины. Помочь в этом смогут, конечно же, дроны.

Своевременное выявление вируса у жителей планеты и вакцинация позволит спасти тысячи жизней. Так, на финале участникам предлагается проработать решение для БПЛА и используя БПЛА. Используя БПЛА, разработать решение для следующих задач:

1. Получение данных о наличии людей на улицах (камеры в общественных местах позволяют определить места массового скопления людей, а сервер передать БПЛА координаты мест, где они находятся).
2. Выявление заболевших среди прохожих. Определение людей с повышенной температурой (зависит от цвета объекта расположенного под коптером, это зелёный, жёлтый или красный цвет). Обеспечение заболевших людей экспресс-тестами.
3. Обработка информации, собранной в пункте 2.
4. Сбор результатов экспресс-теста (для этого мы используем QR-коды) на наличие вируса и возвращение БПЛА на точку взлета.

Программная часть

Для распознавания QR-кодов мы использовали библиотеку pyZBar. Она уже установлена в последнем образе для Raspberry Pi.

Скрипт будет занимать 100% процессора. Для искусственного замедления работы скрипта можно запустить throttling кадров с камеры, например, в 5 Гц (`main_camera.launch`):

```
<node pkg="topic_tools" name="cam_throttle" type="throttle"
   args="messages main_camera/image_raw 5.0 main_camera/image_raw_throttled"/>
```

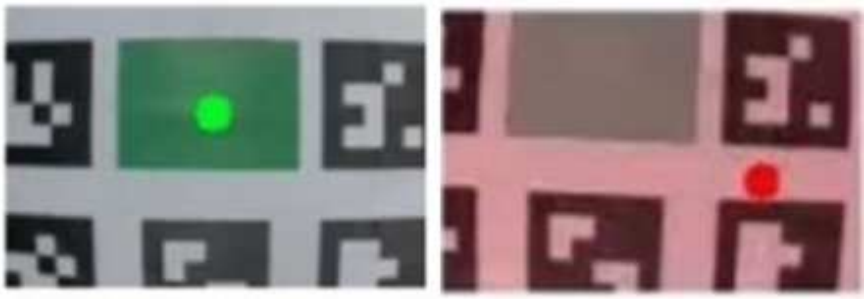
Топик для подписчика в этом случае необходимо поменять на `main_camera/image_raw_throttled.`

Для распознавания цвета мы использовали OpenCV.

Проблемы

Обратите внимание:

1. Пока коптер не увидит QR-код, он не полетит дальше.
2. На распознание цвета влияет освещение, поэтому желательно избавиться от солнечного света в помещении, так как солнечный свет делает изображение с неправильной цветопередачей.



Код

```
from __future__ import print_function
import rospy
import cv2 as cv
from clever import srv
from std_srvs.srv import Trigger
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
from clever.srv import SetLEDEffect
from pyzbar.pyzbar import decode as qr_read
from threading import Thread

# inits
rospy.init_node('flight')
bridge = CvBridge()

# proxys
set_effect = rospy.ServiceProxy('led/set_effect', SetLEDEffect)
get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)

# pubs
color_debug = rospy.Publisher("/color_debug", Image)
qr_debug = rospy.Publisher("/qr_debug", Image)


def lenta():
    print('blink purple')
    set_effect(effect='blink', r=255, g=0, b=255)
    rospy.sleep(5)
    set_effect(r=0, g=0, b=0)
    print('blink off')


def lenta_r():
    print('blink red')
    set_effect(effect='blink', r=255, g=0, b=0)
    rospy.sleep(5)
    set_effect(r=0, g=0, b=0)
    print('blink off')


def check_temp(data):
    global cap  # var for waiting the capture
    frame = bridge.imgmsg_to_cv2(data, 'bgr8')[80:160, 100:220]  # get frame
    hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

    # get binarized images in each color
    red = cv.inRange(hsv, (165, 70, 158), (255, 209, 255))
    yellow = cv.inRange(hsv, (10, 80, 88), (49, 220, 225))
    green = cv.inRange(hsv, (26, 28, 60), (135, 162, 225))

    # count non-zero pixels
    color = {'r': cv.countNonZero(red),
             'y': cv.countNonZero(yellow),
             'g': cv.countNonZero(green)}

    temperature[n] = max(color, key=color.get)  # get max key
    print(n, color, '     ', temperature[n])
```

```
    # draw circle in centor of colored spot (only need color)
    try:
        if temperature[n] == 'r':
            moments = cv.moments(red, 1)  # get moments for find the center
            dM01 = moments['m01']
            dM10 = moments['m10']
            dArea = moments['m00']
            x = int(dM10 / dArea)
            y = int(dM01 / dArea)
            cv.circle(frame, (x, y), 5, (0, 0, 255), -1)  # draw
        if temperature[n] == 'y':
            moments = cv.moments(yellow, 1)
            dM01 = moments['m01']
            dM10 = moments['m10']
            dArea = moments['m00']
            x = int(dM10 / dArea)
            y = int(dM01 / dArea)
            cv.circle(frame, (x, y), 5, (0, 255, 255), -1)
        if temperature[n] == 'g':
            moments = cv.moments(green, 1)
            dM01 = moments['m01']
            dM10 = moments['m10']
            dArea = moments['m00']
            x = int(dM10 / dArea)
            y = int(dM01 / dArea)
            cv.circle(frame, (x, y), 5, (0, 255, 0), -1)
    except ZeroDivisionError:
        print('zero')

    color_debug.publish(bridge.cv2_to_imgmsg(frame, 'bgr8'))  # publish to topi

    # led and print if covid
    if max(color, key=color.get) == 'y' or max(color, key=color.get) == 'r':
        t = Thread(target=lenta)
        t.daemon = True
        t.start()
        print('sbrosheno')

    # unsubscribe from topic (get only one capture)
    image_sub.unregister()
    cap = True


def qr_check(data):
    global cap
    frame = bridge.imgmsg_to_cv2(data, 'bgr8')
    barcodes = qr_read(frame)  # read the barcode using zbar
    if barcodes:
        print(barcodes[0].data)

        # draw rect and publish to topic
        (x, y, w, h) = barcodes[0].rect
        cv.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)
        qr_debug.publish(bridge.cv2_to_imgmsg(frame, 'bgr8'))

        if barcodes[0].data == 'COVID - 19' or barcodes[0].data == 'COVID - 20
            t = Thread(target=lenta_r)
            t.daemon = True
            t.start()

        cap = True
        image_sub.unregister()


# coords of each point
coords = {1: [0.295, 0.295, 1],
```

```
            3: [0.295, 0.885, 1],
            5: [0.295, 1.475, 1],
            7: [0.295, 2.065, 1],
            9: [0.59, 2.655, 1],
            8: [0.885, 2.065, 1],
            6: [0.885, 1.475, 1],
            4: [0.885, 0.885, 1],
            2: [0.885, 0.295, 1]}

# dict for temperatures
temperature = {}

# copter's way
path = [1, 3, 5, 7, 9, 8, 6, 4, 2]

# take off
print()
print('take off')
navigate(x=0, y=0, z=1.5, speed=0.5, frame_id='body', auto_arm=True)
rospy.sleep(1.3)
telem = get_telemetry(frame_id='aruco_map')
navigate(x=telem.x, y=telem.y, z=1.5, frame_id='aruco_map')
rospy.sleep(13)

# go using our way
for n in path:
    cap = False
    print()
    print('flight to', n, coords[n])
    navigate(x=coords[n][0], y=coords[n][1], z=coords[n][2], frame_id='aruco_ma
    rospy.sleep(4)
    image_sub = rospy.Subscriber('main_camera/image_raw', Image, check_temp, qu
    while not cap:  # wait the capture
        rospy.sleep(0.5)
    rospy.sleep(3)

# home
print()
print('flight to home')
navigate(x=0, y=0., z=1.5, frame_id='aruco_map')
rospy.sleep(4)
print('land')
land()

print()
print(temperature)

print()
print('wait 2m')
rospy.sleep(120)

# take off
print()
print('take off')
navigate(x=0, y=0, z=1.5, speed=0.5, frame_id='body', auto_arm=True)
rospy.sleep(1.3)
telem = get_telemetry(frame_id='aruco_map')
navigate(x=telem.x, y=telem.y, z=1.5, frame_id='aruco_map')
rospy.sleep(3)

for n in path:
    if temperature[n] == 'r' or temperature[n] == 'y':  # if was temperatute hi
        cap = False
        print()
        print('flight to', n, coords[n])
        # flight to this point
```

```
        navigate(x=coords[n][0], y=coords[n][1], z=coords[n][2], frame_id='aruc
        rospy.sleep(3)
        image_sub = rospy.Subscriber('main_camera/image_raw', Image, qr_check,
        rospy.sleep(4)
        if not cap:
            print('spusk')
            navigate(x=coords[n][0], y=coords[n][1], z=0.7, speed=0.5, frame_id
            while not cap:
                rospy.sleep(0.5)
        image_sub.unregister()  # unsubscribe of topic in each situation
        rospy.sleep(3)

    else:  # it it was good we won't fly there
        print()
        print(n, 'healthy at first')

# home
print()
print('flight to home')
navigate(x=0, y=0., z=1.5, frame_id='aruco_map')
rospy.sleep(5)
print('land')
land()
```

Если вы хотите использовать эту программу с другими координатами, то их можно изменить в coords, а последовательность полёта по ним, задать в `path`.

Инженерная часть

Все чертежи, модели и программу для тестирования можно найти в архиве.

Наименование выполняемых работ

Разработка дополнительного, модульного устройства для транспортировки и доставки хрупкого, малогабаритного груза (экспресс-теста).

Цель выполнения работ

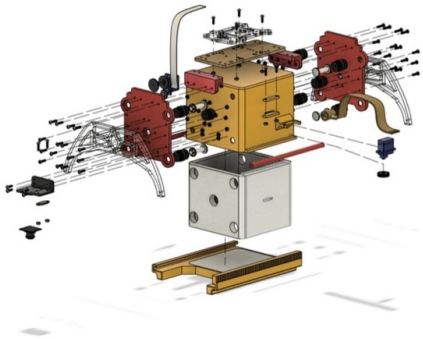
Цель выполнения работ – расширение функциональных возможностей программируемого квадрокоптера “COEX Клевер 4 Code”, при помощи устройства транспортировки груза, для возможности оперативной, бесконтактной доставки квадрокоптером экспресс-тестов и вакцины. Целевая аудитория – скорая помощь, эпидемиологи, МЧС, лица с риском заболевания.

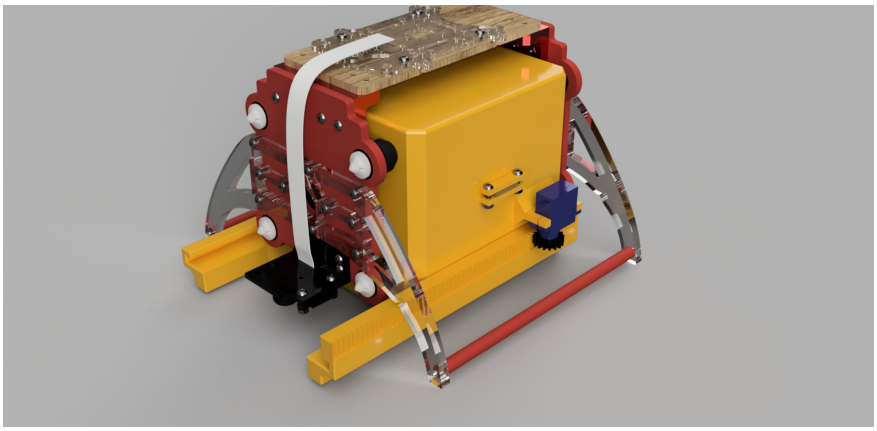
Минимальные требования к устройству

- Модульность;
- возможность изготовления посредством 3D печати;
- возможность осуществления сборки устройства посредством винтового/шпоночного соединения;

- способность вмещать/захватывать груз 10x10x10 см., весом не менее 200 г.;
- возможность транспортировки хрупкого груза при помощи данного устройства;
- возможность реализации бесконтактной доставки груза, не нарушая его целостности, при помощи данного устройства;
- совместимость с программируемым квадрокоптером “COEX Клевер 4 Code”;
- возможность монтажа на нижнюю деку.

Инструкция по созданию устройства





3D-Печать

1. Боковое крепление блока

Функция: Крепление. Материал: PLA/ABS (или аналог). Линии стенки 2. Заполнение 10%. Количество: 2шт. Примечание: Можно вырезать на лазере или на фрезере.

2. Демпфер

Функция: Амортизация. Материал: TPU/FLEX (или аналог). Заполнение 100%. Количество: 10шт.

3. Штифт

Материал: PLA/ABS(или аналог). Заполнение 100%. Количество: 8шт.
Примечания: Вы можете купить их отдельно.

4. Крышка штифта

Материал: PLA/ABS(или аналог). Заполнение 100%. Количество: 8шт.
Примечания: Вы можете купить их отдельно.

5. Крепления камеры

Функция: Крепление камеры и датчика к захвату груза. Материал: PLA/ABS(или аналог). Заполнение 100%. Количество: 1шт.

6. Крышка

Функция: Подвижная крышка захвата. Материал: PLA/ABS (или аналог). Линии стенки 2. Заполнение 10%. Количество: 1шт.

7. Внутренний куб

Функция: Основная часть захвата в которую помешается груз. Материал: PLA/ABS (или аналог). Линии стенки 2. Заполнение 10%. Количество: 1шт.

8. Распорка

Функция: Помогает удерживать ножки. Материал: PLA/ABS (или аналог). Заполнение 50%. Количество: 2шт. Примечание: Лучше использовать деревянный пруток. Высверлив в нем с двух сторон отверстия под винты м3.

9. Катушка

Функция: Для затягивания ленты. Материал: PLA/ABS (или аналог). Заполнение 100%. Количество: 1шт.

10. Шестерня

Функция: Насадка на сервопривод для открытия крышки. Материал: PLA/ABS (или аналог). Заполнение 100%. Количество: 1шт.

11. Уголки

Функция: Соединение между верхнем креплением и боковым креплением . Материал: PLA/ABS (или аналог). Заполнение 50%. Количество: 2шт.

Лазерная резка

1. Верхнее крепление

Функция: Переходное крепления захвата. Материал: Дерево 4мм. Количество: 1шт.

2. Боковое крепление

Функция: Переходное крепления захвата. Материал: Дерево 5мм. Количество: 2шт.

3. Ножка левая

Функция: Опорный элемент. Материал: Поликарбонат 6мм. Количество: 2шт.

4. Ножка правая

Функция: Опорный элемент. Материал: Поликарбонат 6мм. Количество: 2шт.

5. Зажим ремня

Функция: Зажим ремня. Материал: Поликарбонат 3мм.

Дополнительные материалы и устройства

Крепеж

- Винт M3-12mm - 28
- Винт M3-14mm - 8
- Винт M3-10mm - 8
- Винт M2-8mm - 1
- Винт M2-6mm - 4
- Гайка 3mm - 44
- Гайка 2mm - 4

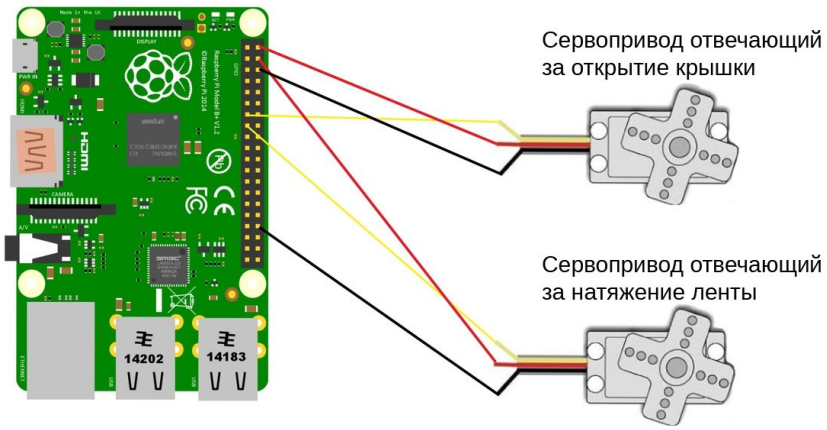
Оборудование

1. **Лента тканевая ширина 18мм**. Примечание- можно использовать любой кусок ткани порезанный на полоски толщиной в 18мм. Длина - 30см.
2. **Штифт и крышка для него**. Количество - 8шт.
3. **Монолитный поликарбонат прозрачный**. размер листа - 200 мм × 200 мм × 6 мм
4. **Поролон толщиной 10мм**. Примечание - можно просто обрезать бытовую губку по нужным размерам.
5. **Сервоприводы Feetech FS90R 360 degrees**. Количество - 2шт.
6. **Демпфер 17.5x7x20mm**. Количество - 10шт.

Сборка

Теперь когда у вас есть все нужные части захвата, можно начинать сборку. Все винтовые соединения рекомендуется фиксировать с помощью синего loctite.

- В первую очередь нужно вырезать из губки или поролона прокладки подходящие по размеру. Для этого можно напечатать на листе А4 приложенный чертеж и аккуратно вырезать деталь.
- Далее при помощи клея четыре губки вклеиваются внутрь куба как показано на чертеже.
- Оставшиеся губка вклеивается в крышку.
- При помощи гаек, винтов м3-12мм прикручиваем ножки к боковым креплениям
- Крепление камеры прикрутить к боковому креплению как показано на чертеже. Теперь это будет передняя часть захвата.
- Прикрутите уголки к боковым пластинам используя винты 14мм.
- Вставьте демпферы в специализированные отверстия центрального куба.
- Закрепите на демпферах боковые пластины
- Закрепите демпферы при помощи штифтов (проденьте штифт в демпфер с внутренней стороны и закрепите его крышечкой)
- Проденьте ленту через куб закрепив в зажиме.
- Проденьте ленту в катушку и прихватите клеем.
- Прикрутите сервопривод с катушкой к специальному креплению как показано на чертеже. И вкрутите винт м2 с другой стороны.
- Прикрутите ко второму сервоприводу шестерню.
- Присоедините крышку к кубу.
- Прикрутите второй сервопривод к специальному креплению.
- Прикрепите камеру и датчик высоты.
- Прикрепите нижнюю пластину Клевера к верхней пластине захвата.
- Прикрутите верхнюю пластину к уголкам.
- Подключите сервоприводы. Красные провода подключаем к 5V, черные к GND, сигнальный провод от сервопривода, отвечающего за открытие крышки к GPIO27, а сервопривод отвечающий за натяжение ленты к GPIO22.



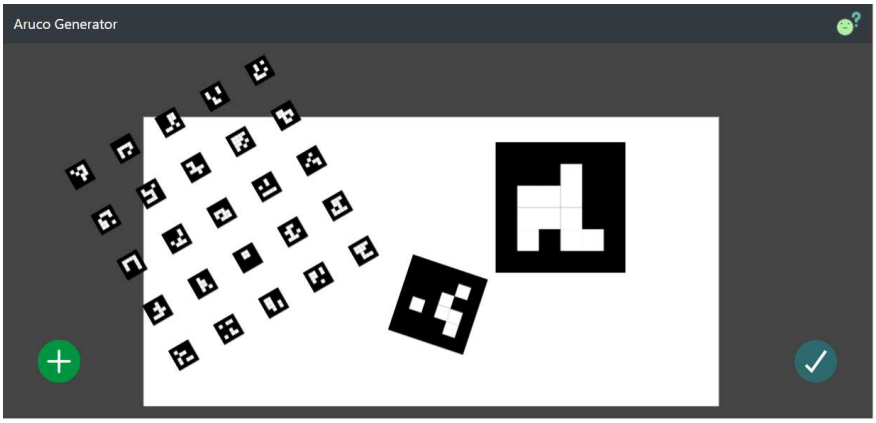
Если вы хотите использовать захват при полете в автономном режиме, то следует поменять параметры в `main_camera.launch` на следующие:

```
<node pkg="tf2_ros" type="static_transform_publisher" name="main_camera_frame"
```

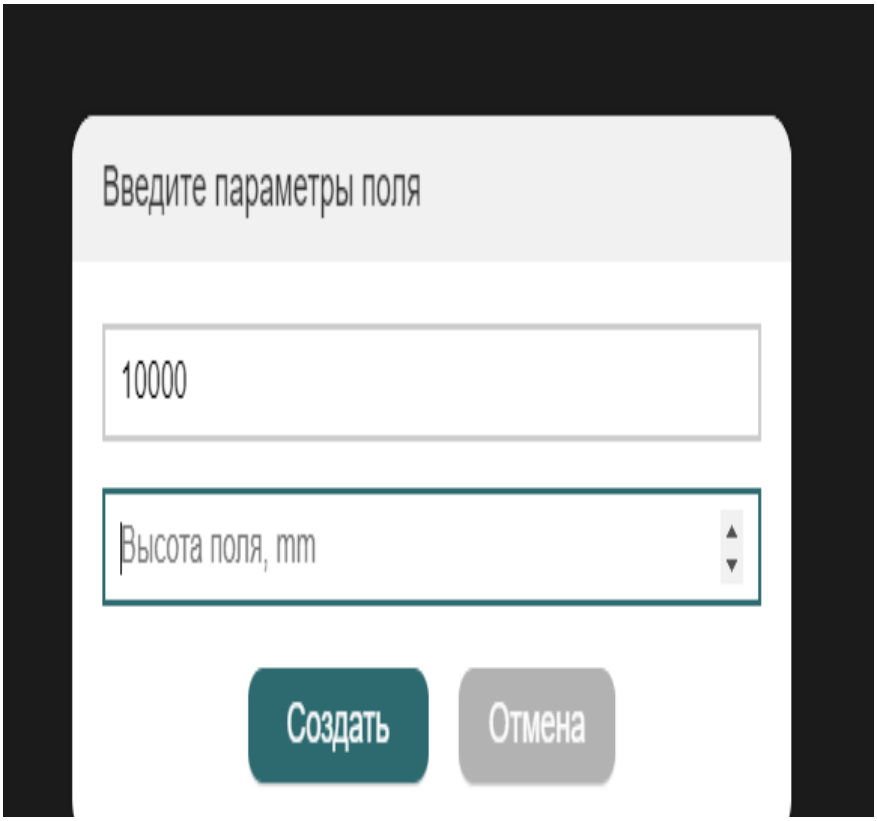
Генератор ArUco карт

Автор проекта: @tenessinum.

Начиная с образа версии *0.16* изменился подход к созданию карт маркеров: маркеры больше не привязаны к сетке и каждый из них теперь можно повернуть на любой угол вокруг всех трёх осей. Вместе с этим изменился и способ задания карт маркеров. Теперь карта загружается из текстового файла (подробнее в статье **Навигация по картам ArUco-маркеров**). Для упрощения процесса создания текстового файла был создан *конструктор полей*.



Создание поля

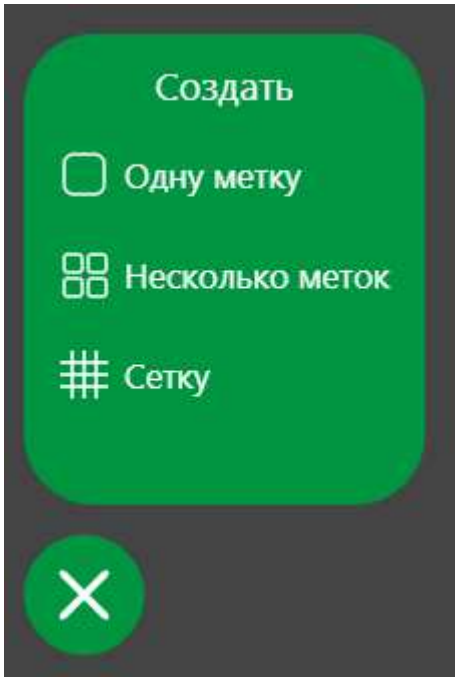




Инструмент творения

Нажав на плюсик в левом углу экрана откроется меню в котором можно создать какой-либо элемент.

- **Одна метка** - просто одинокая метка в поле
- **Несколько меток** - группа меток выстроенная в сетку
- **Сетка - инструмент** к которому можно привязывать метки



Экспорт

Карту можно экспортировать в двух форматах: ***txt*** (для Клевера) и ***svg*** (для печати)

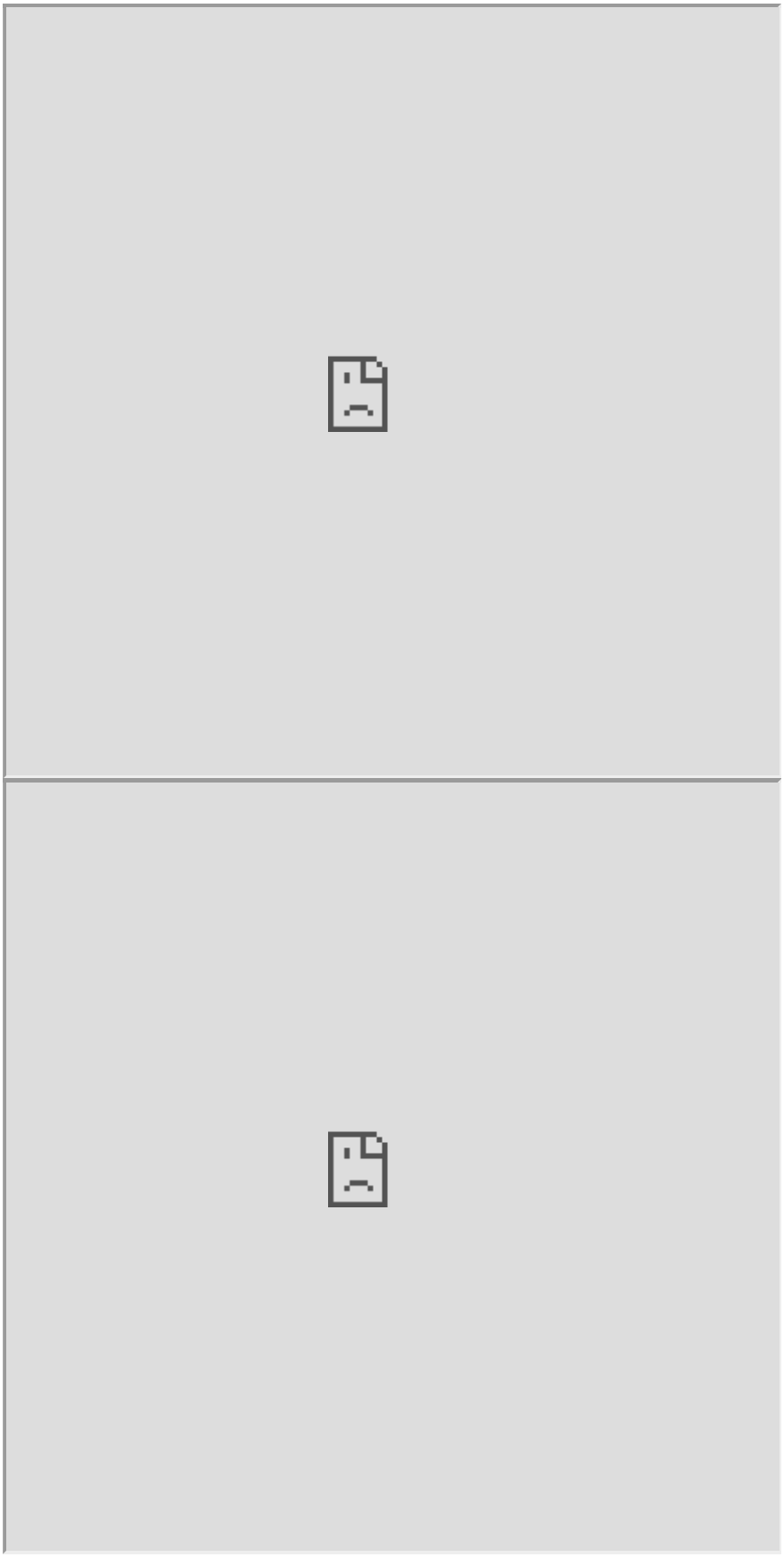


Аэротакси

Проект команды Human Express на проектной программе "Большие Вызовы"

Суть проекта

Крупные города страдают из-за пробок и перегруженности транспорта. Пробки на дорогах и перегруженность транспорта влечёт за собой многие неудобства. Одной из таких проблем является отсутствие возможности быстро добраться из точки А в точку Б. При этом воздушное пространство практически не используется. Предлагаемое решение заключается в создании системы, которая в режиме реального времени наблюдает и контролирует движение беспилотных летательных аппаратов. Такое решение приводит к полной автоматизации процесса полёта и исключает возможность воздушно-транспортных происшествий. В результате проделанной работы удалось сделать систему, которая состоит из нескольких дронов и сервера. Сервер прокладывает маршрут дронам из начальной точки в заданную по проложенным дорогам. Также в работу сервера входит логистика, благодаря которой беспилотники не сталкиваются в полёте.



Настройка сервера

Чтобы скачать проект на сервер выполните команду

```
git clone https://github.com/Tennessium/HUEX
```

Перед началом работы с системой необходимо подключить устройство к сети WiFi (например, раздать со смартфона) установить на него необходимые библиотеки

```
cd HUEX/server
pip install -r requirements.txt
```

В случае необходимости вы можете изменить высоту эшелонов и разрешенные IP-адреса для доступа к Центру управления полетами в файле *server/consts.py* и настроить поле с метками в *server/static/map.txt*. Теперь можно запустить сервер написав в командную строку

```
python manage.py runserver 0.0.0.0:8000
```

Чтобы перейти на веб страницу наберите в адресной строке ip адрес сервера в локальной сети и укажите порт 8000 (`http://ip:8000`). Как узнать ip адрес устройства

Настройка коптеров

В первую очередь подготовьте SD-карту с образом Clover (Инструкция)

Чтобы скачать проект на Raspberry Pi в коптере выполните команду

```
git clone https://github.com/Tennessium/HUEX
```

Перед началом работы с системой необходимо перевести коптеры в режим клиента и подключить к сети WiFi. Вы можете воспользоваться этим мануалом

Однако, для упрощения развертывания системы на нескольких коптерах, рекомендуется использование нашего скрипта, лежащего в папке *copter/setup/*

- Перейдите в папку

```
cd HUEX/copter/setup/
```

- Используя любой редактор, в файле *networkData.txt* измените SSID и пароль сети
- Запустите скрипт

```
sudo bash networkEdit.sh
```

- Перезагрузите Raspberry Pi

Произведите установку и настройку ROS-пакета для LED-ленты

```
cd ~/catkin_ws/src
git clone https://github.com/bart02/ros-led-lib.git led
cd led
```

Воспользовавшись nano ledsub.py, измените переменную LED_COUNT на число светодиодов на вашей ленте

```
chmod +x ledsub.py
cd ~/catkin_ws
catkin_make
sudo systemctl enable /home/pi/catkin_ws/src/led/led.service
sudo systemctl start led
```

Установите необходимые пакеты

```
cd HUEX/clever
pip install -r requirements.txt
```

В файле *copter/consts.py* укажите IP-адрес сервера.

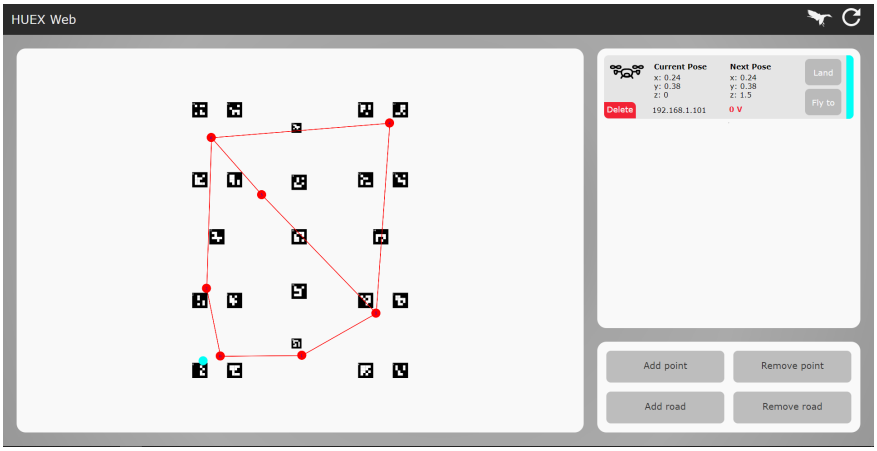
Для запуска основного скрипта воспользуйтесь нашим systemd-сервисом.

```
sudo systemctl enable /home/pi/HUEX/clever/setup/taxi.service
sudo systemctl start taxi.service
```

Скрипт будет запускаться автоматически при старте системы. Для остановки можно воспользоваться командой

```
sudo systemctl stop taxi.service
```

Веб-интерфейс центра управления полётами



В данном веб интерфейсе можно следить за полётами всех дронов на карте (масштабировать с помощью колёсика, передвигать с помощью Alt). При нажатии на лебедя в правом верхнем углу все коптеры аварийно садятся, А при нажатии на значок "обновить" все коптеры автоматически удаляются, что приводит к удалению всех команд и посадке активных на текущий момент коптеров. С помощью инструментов в правом нижнем углу можно строить новые основания и рёбра.

Веб-интерфейс заказа

Чтобы перейти на веб страницу наберите в адресной строке `http://ip:8000/m`, где *ip* - адрес сервера в сети.

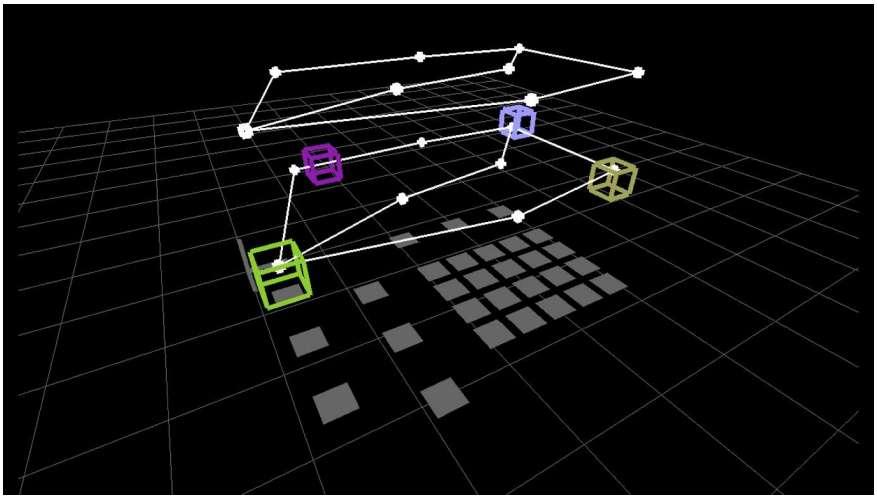
Визуализатор

Для наглядности работы системы был разработан 3D-визуализатор воздушного пространства. Он отображает систему эшелонов, поле ArUco-маркеров; позиции коптеров и направления их движения в real-time.

Запускать визуализатор можно после старта сервера. Компьютер должен быть подключен к той же сети, что и сервер.

```
cd viz
python main.py
```

Программа автоматически подгрузит карты маркеров и эшелонов. Если одна из них изменяется в процессе эксплуатации, перезагрузите программу для внесения изменений.



> Камеру можно передвигать при помощи клавиш *WASD* и поворачивать при помощи стрелок.

DDoS-скрипт

Во время смены был написан простой скрипт, предназначенный для тестирования логистики. Его суть заключается в бесконечном заказе такси между случайными точками. Перед тем как запустить программу *DDos.py* замените параметр *static_path* в пятой строке на *ip* вашего сервера.

API

На случай если вы захотите реализовать свою "обёртку" вы можете реализовать взаимодействие с сервером по средствам *HTTP/HTTPS* запросов

/get

Возвращает телеметрию всех доступных коптеров. Пример:

```
{
  "message": "OK",
  "drones": [
    {
      "ip": "192.168.1.101",
      "led": "#FF0000",
      "status": "land",
      "pose": {
        "x": 0.24,
        "y": 0.38,
        "z": 0,
        "yaw": 0
      },
      "voltage": 4.12,
      "nextp": {
        "led": "#FF0000",
        "status": "land",
        "pose": {
          "x": 0.24,
          "y": 0.38,
          "z": 1.5,
          "yaw": 0
        }
      }
    }
  ]
}
```


Где

- **ip** - ip адрес коптера
- **led** - цвет светодиодной ленты
- **status** - *fly* или *land* - текущий статус коптера
- **pose** - позиция коптера (*x*, *y*, *z*, и *yaw*)
- **voltage** - напряжение на одной банке
- **nextp** - отдаваемая коптеру команда на полёт (*led*, *status*, *pose* как выше)

/static/roads.json

Текущая карта дорог первого эшелона Пример:

```
{
  "points": [
    {
      "x": 2.1,
      "y": 3.5
    },
    {
      "x": 0.6,
      "y": 0.4
    },
    {
      "x": 2.4,
      "y": 0.5
    }
  ],
  "lines": [
    {
      "1": 2,
      "2": 1
    },
    {
      "1": 1,
      "2": 0
    }
  ]
}
```

Где

- **poits** - массив вершин графа и их координат
- **lines** - массив рёбер графа (*1* - точка из которой выходит ребро, *2* - точка куда это ребро направлено)

/ask_taxi?o=x&t=y

Заказ такси из точки под номером *x* в точку под номером *y*. Точки *x* и *y* берутся из **/static/roads.json**

/get_dist?o=x&t=y

Возвращает расстояние и цену за пролёт между точками. Пример:

```
{"dist": 5.3, "cost": 309}
```

Где

- **dist** - Дистанция в метрах
- **cost** - Цена в рублях (150₽ + 30₽ × *n* метров)

Шаровая защита коптера

Введение

Наверное, летать в помещениях приходилось каждому, кто брал в руки коптер. Подобные полеты сопряжены с немалым риском повредить коптер о стены и различные предметы. Даже полеты на относительно больших пространствах связаны с рисками удариться о препятствие: на пути коптера может встать ствол дерева или здание — что уж говорить о полетах в замкнутых пространствах. Подобные «краш-тесты» не очень приятный момент, который может оказаться в лучшем случае причиной потери внушительной суммы денег на ремонт, а в худшем — и вовсе утраты коптера. Тем более неприятны такие ситуации для новичка, который не может своевременно увернуться от препятствия и только учится летать.

Это все подвигло нас к поиску решений. К сожалению, перерыв весь интернет, мы не нашли достаточно легкого и простого в изготовлении решения для простых пользователей, а главное — такого, которое будет всем по карману. Например, защита по контуру пропеллеров неплохо предохраняет сами пропеллеры, но при малейшем касании о препятствие коптер переворачивается и падает. В общем, защита либо не оберегала коптер полностью, либо выглядела несуразно и была слишком узко доступна.



Нами было принято сложное решение: придется делать ее полностью самим и почти с нуля, а также поставлена цель сделать ее простой в изготовлении и максимально легкой.

Разработка

В результате поиска решения, удовлетворяющего всем нашим требованиям, мы остановились на нескольких схожих вариантах. Было решено сделать защиту в форме полуправильного многогранника (примерами могут служить

фуллерен, молекула углерода, или фигура пентакисдодекаэдр) — его мы и выбрали как самый приятный глазу. Кроме того, такая защита легко масштабируема под нужный размер.

При создании такой фигуры используются два вида ребер (далее — лучей): короткие и длинные, их длины рассчитываются исходя из нужного диаметра вписанной в многогранник сферы. Для лучшего понимания я вставлю все необходимые формулы ниже из «Википедии».

Если «короткие» рёбра пентакисдодекаэдра имеют длину a, то его «длинные» рёбра имеют длину $\frac{1}{6}\left(9-\sqrt{5}\right)a \approx 1,13a$

Радиус вписанной сферы (касающейся всех граней многогранника в их инцентрах) при этом будет равен

$$r = \frac{1}{2}\sqrt{\frac{1}{109}\left(477+194\sqrt{5}\right)}\,a \approx 1,4453319a,$$

С угловыми соединениями (фитингами), тоже не все просто: их также два вида — с пятью гранями при вершине (пять лучей исходят из вершины) и с шестью гранями (шесть лучей исходят из вершины).

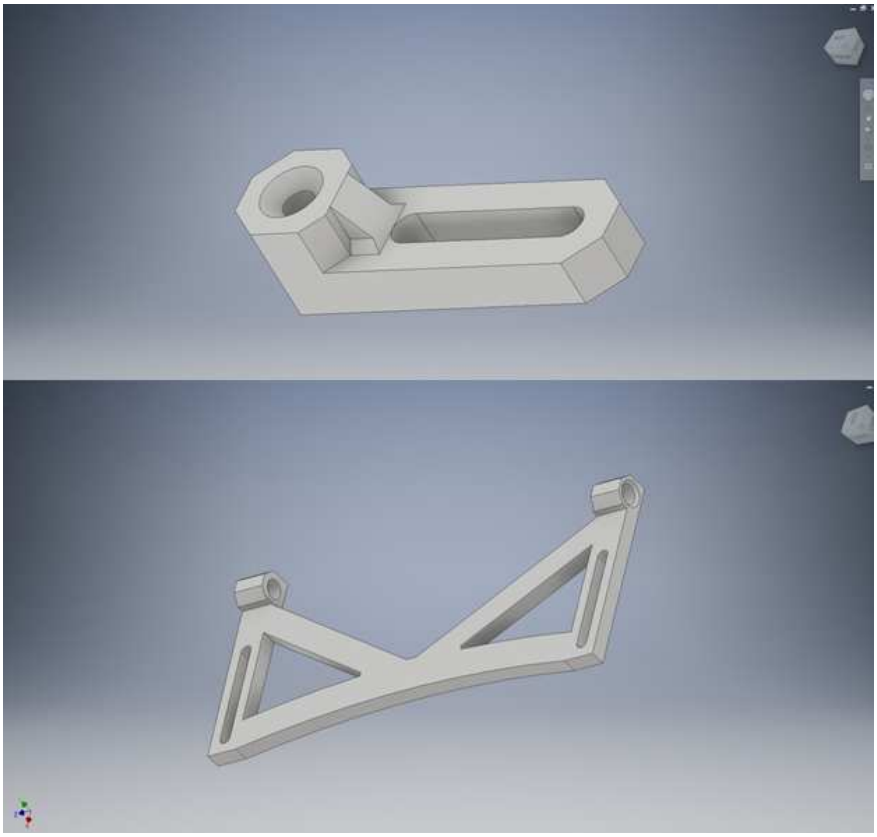
Первые модели

Была составлена спецификация для удобства контроля процесса изготовления, и мы приступили к моделированию.

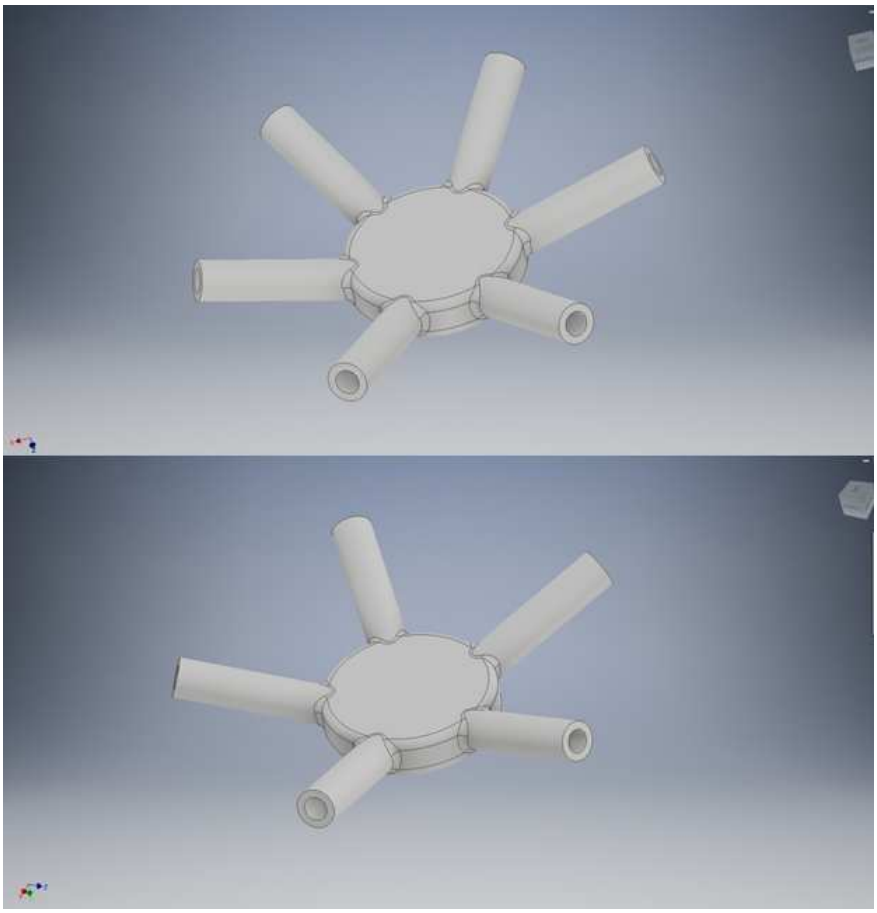
Сделав несложные расчеты под нужные размеры, мы построили модели в Inventor CAD.

В ходе проектирования мы столкнулись с проблемами в моделировании угловых соединений, но они были решены упрощением конструкции, а разность углов компенсируется гибкостью материалов. Таким образом, все соединения сидят в небольшом натяге.

Перв. примен.	Справ. №	Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Приме-чание
						5-лучик	12	
						6-лучик	20	
						Луч 12.5см	60	
						Луч 14см	30	
						Крепёж осн.	2	
						Крепёж доп.	2	
						Луч коптера с изм.	2	
						Стандартные изделия		
						Винт М3х16мм	6	
						Гайка М3	6	



(Элементы крепления защиты к корпусу)

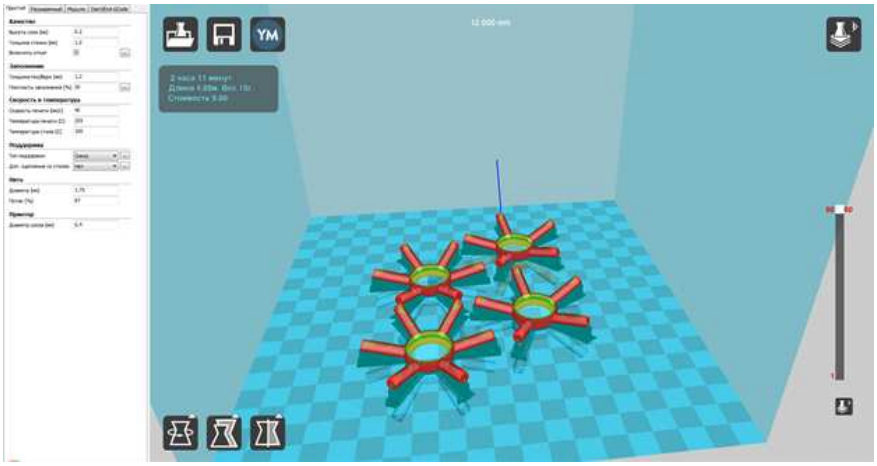


Материалы

В ходе проектирования встал вопрос, из чего же все-таки сделать такую защиту, чтобы получилось легко и прочно. Ответ пришел, как всегда, совершенно неожиданно. На глаза попались шпажки из бамбука: они достаточно тонкие, чтобы не повлиять на аэродинамику, имеют достаточную гибкость и при этом достаточно прочные. Далее возник вопрос, из чего и как делать фитинги. Конечно же, 3D-печать! 3D-принтер — это вообще незаменимая вещь, тем более для тех, кто любит что-то делать сам. К тому же они из-за не самой высокой цены получили достаточно широкое распространение. На таком принтере можно делать изделия почти любой сложности. То, что надо!

Готовые модели переводим в .stl, закидываем в слайсер (в нашем случае — Cura), вводим настройки под конкретный принтер и пластик и ставим на печать.

Для уменьшения веса был выбран ABC-пластик как один из самых легких и доступных.



Шпажки были порезаны на расчетные длины и подготовлены к последующей работе.

Сборка и установка

После того как все было напечатано и порезано, настало время собирать защиту.



Сборка здесь самый ответственный момент, так как требует специального алгоритма.

Из пятилучевого фитинга выходят только короткие лучи, в то время как из шестилучевого — только каждая вторая длинная.

Сборка:

1. Вначале собираем все пятилучевые вершины.
2. На каждый луч, исходящий из пятилучевой вершины, надеваем шестилучевую.
3. Соединяем между собой шестилучевые фитинги длинными шпажками.
4. Присоединяем уже собранные пятилучевые вершины к шестилучевым, учитывая, что в шестилучевом фитинге короткие и длинные лучи чередуются.
5. Повторяем процесс для каждой пятилучевой вершины, пока шар не соберется.

После сборки разделяем шар на две полусферы, устанавливаем крепления на коптер, смотрим, что все подходит.



(Пример установки креплений)

Теперь полусферы можно проклеивать. Между собой полусферы не склеиваются — это нужно для установки коптера внутрь. Мы в качестве клея использовали растворитель для пластиков «Дихлорэтан», но с тем же успехом можно использовать и любой быстросохнущий клей для полимеров.

После высыхания защита готова к установке и первым пробным полетам!



(Пока еще без креплений)



Первые полеты

Мы делали защиту под коптер «Клевер 2», являющийся обучающим пособием по сборке и настройке квадрокоптеров, и на него она устанавливается без доработок. Защита весит на 70 г больше (139 грамм), чем стандартная, и на управляемость и время полета почти не влияет.

Отдельно стоит сказать, что излишние вибрации, если таковые имеются, можно убрать путем более жесткого крепления защиты к коптеру.

В итоге получилась необычная защита для коптера с небольшим весом и интересным дизайном, открывающая новые горизонты для полетов в тех местах, где летать для коптера раньше было опасно.

Система засечки (хронометража) для дронов

CopterHack-2021, команда **Atomic Ferrets**.

Наша команда состоит из двух человек - Стецкий Сергей и Давлетшин Денис.

По всем вопросам обращаться на почту (bashirianboy@gmail.com) или Телеграм.

Идея

На сегодняшний день существует множество засечек для дронов, но тех, которые производят в России, мало, а таких, которые бы имели приемлемую цену, еще меньше. Поэтому было решено сделать собственную систему засечки, которая давала бы весь нужный функционал, и производство которого было бы не столь затратным.

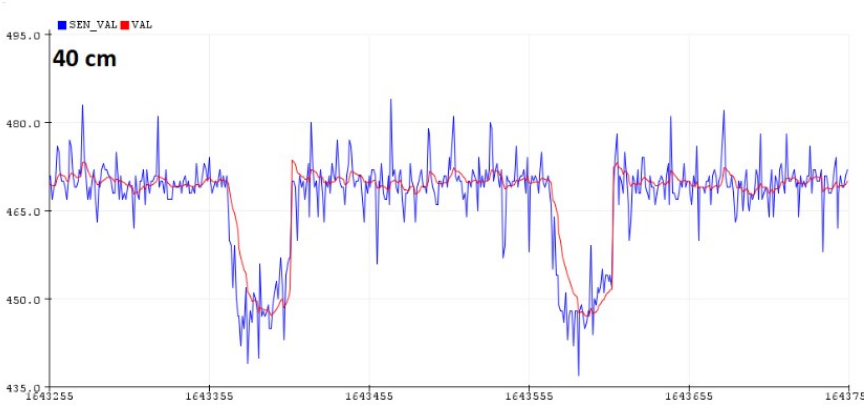
Ознакомиться с кодом проекта вы можно через репозиторий: https://github.com/stinger000/CopterHack2020_IR_LED.

Принцип работы

Прелесть нашей системы в том, что ее концепция проста - использование чувствительного фотодатчика - фотодиода, и светодиода. Фотодатчик и ИК светодиод крепятся на ворота друг напротив друга, и светодиод все время направлен на датчик. Когда дрон пролетает через ворота, он на какое-то время перекрывает собой светодиод, и показания на датчике тут же меняются. На этом и основан принцип работы. Остается только подключить датчик к схеме, способной считывать световой поток, падающий на него, и затем отправлять сигнал о детектировании пролета на компьютер. Для наших целей мы использовали Arduino.

Для Arduino написан код, который обрабатывает постоянно поступающие на него сигналы с фотодиода. Затем используется имитация фильтра низких частот, сглаживающий шумы, которые по сравнению с реальным сигналами более низкочастотные. Фильтрация шумов помогает в том случае, когда датчик и светодиод разнесены далеко друг от друга, и световой поток в этом случае может смешаться с шумом.

На рисунке ниже изображена типичная временная диаграмма сигнала. "Ямы" образуются во время пролета через засечку. Синим цветом показан "сырой" сигнал, а красным - обработанный с помощью low-pass фильтра. Подбором параметров фильтра можно добиться того, чтобы принципиальный вид сигнала не менялся, а шумы сглаживались.



Алгоритм работает так, что пролет засчитывается поле того, как будет обнаружен спад и последующий подъем сигнала, а также амплитуда сигнала должна больше некоторого выставленного вручную минимума (зависит от шумов собранной системы).

Затем по Serial-порту Arduino отправляет специальный символ, который обозначает обнаружение пролета.

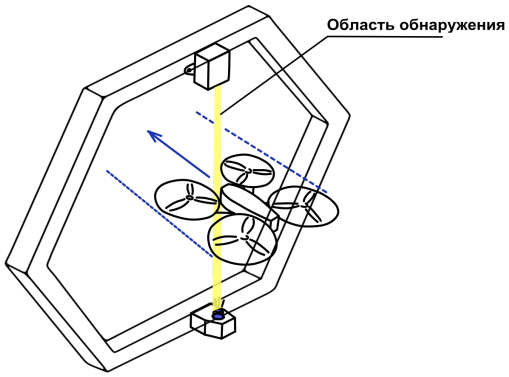
Компьютер, к которому подключена засечка, может считывать этот сигнал, и затем выводить на любой интерфейс.

Результат работы

Вы можете посмотреть короткую видеопрезентацию:

Использование

Данный проект можно использовать для любых квадрокоптеров достаточно больших размеров. Условие, которое необходимо выполнить, это то, чтобы дрон при пролете перекрывал собой светодиод. То есть дрон должен попадать в довольно узкий диапазон. Это наглядно проиллюстрировано на рисунке ниже:



Для начала работы скачайте репозиторий с кодом обработки пролетов и репозиторий с пользовательским интерфейсом:

```
git clone https://github.com/stinger000/CopterHack2020_IR_LED
git clone https://github.com/stinger000/CopterHack2020_IR_LED_Desktop_GUI
```

Зайдите в папку Signal_handler репозитория `CopterHack2020_IR_LED` и откройте в Arduino IDE файл `AnalogReadSerial_filtered_v2.ino`.

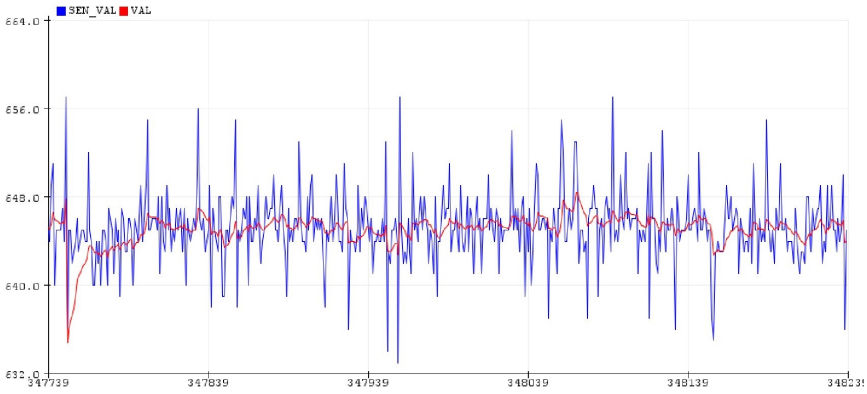
Надо раскомментировать строки:

```
//debug in plotter
//Serial.print(sensorValue);   // uncomment this 4 rows
//Serial.print(" ");
//Serial.println(value);
//...
```

и закомментировать строку вывода символа в Serial-порт:

```
// ...
if ( rising_edge & falling_edge  )
{
    // ...
    // Send time by serial port to GUI
    Serial.println(millis()); //comment this line
    //...
}
```

Перепрошейте ваше Arduino этим файлом. Затем проверьте сигнал в плоттере.



По нему задать величину шума, получившуюся для вашей системы: за это отвечает переменная `noise`.

Например на рисунке ниже величина шумов (для отфильтрованного красного сигнала) достигает 10 единиц.

Затем закомментируйте и раскомментируйте обратно строки кода и перепрошейте Arduino.

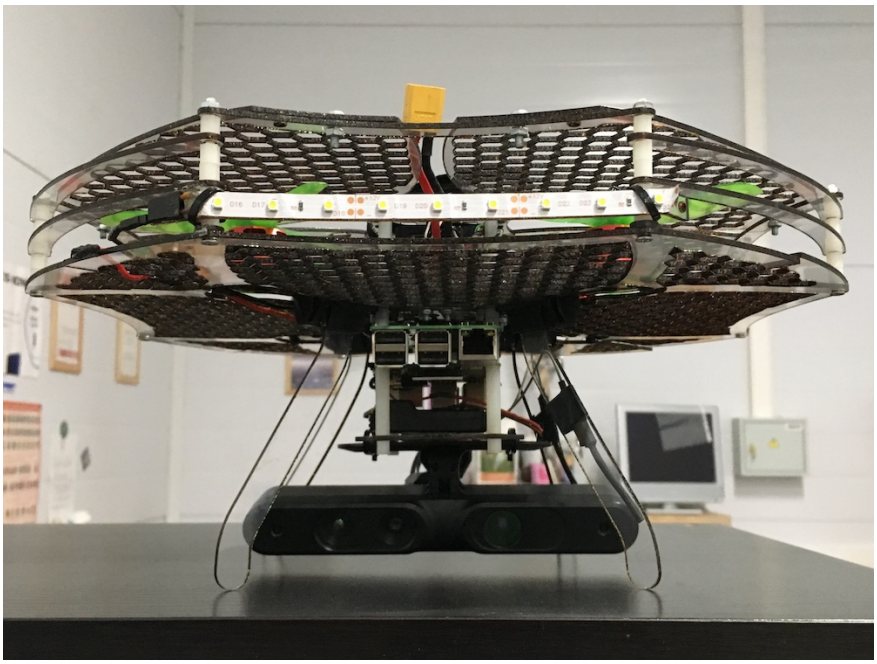
В терминале запустите файл с графическим интерфейсом (# такой то файл):

```
cd CopterHack2020_IR_LED_Desktop_GUI
python main.py
```

Затем подключите Arduino через USB-кабель в ваш компьютер и нажмите "Connect".

Начало и конец отсчета замеров производится кнопками "start" и "stop".

Дрон для 3D-сканирования человека



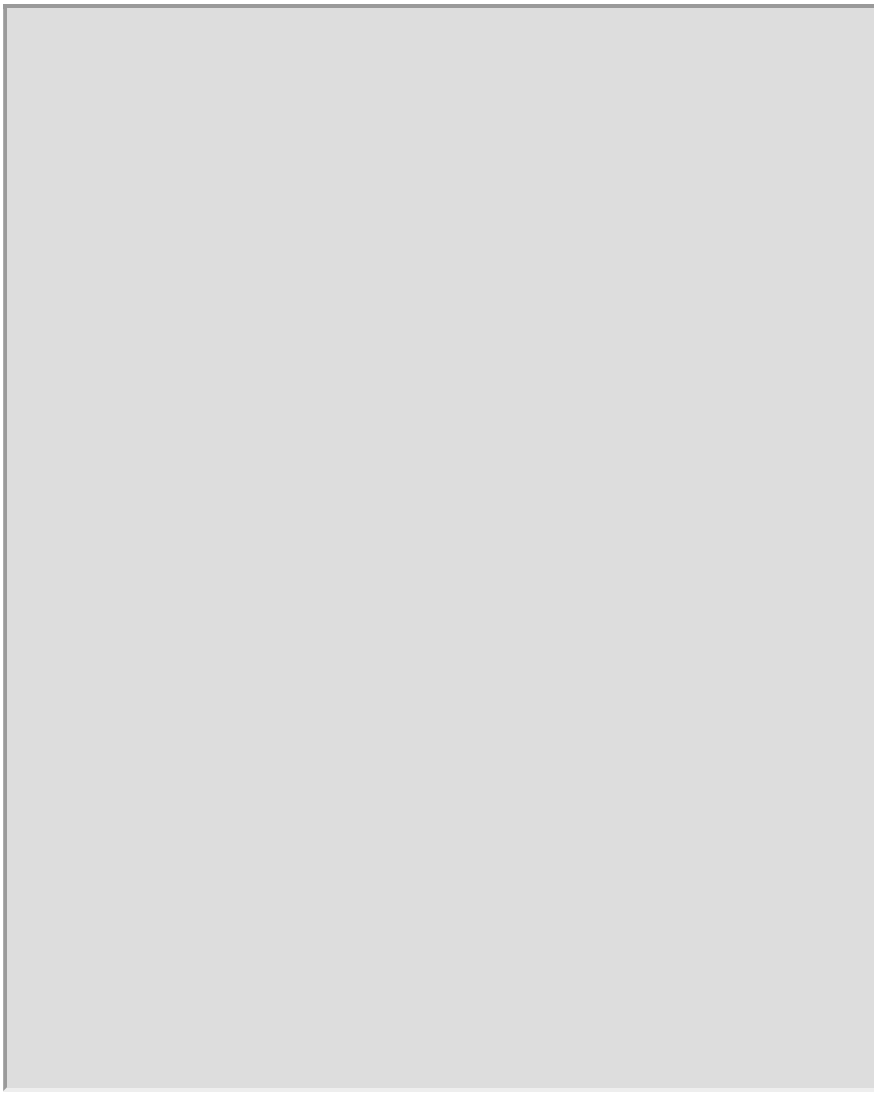
Проект был создан совместно с компанией Texel, которая разрабатывает стационарные 3D-сканеры для создания модели человека.

Коллеги из Texel предоставили модуль, состоящий из Raspberry Pi с установленным ПО для сканирования и камеры для захвата 3D-изображения PrimeSense.

С нашей стороны был предоставлен квадрокоптер Клевер 3 с установленным оборудованием для автономного полета, а также написана полетная программа.

Мы провели множество тестов и настроек, внесли много изменений в конструкции дрона, чтобы в итоге добиться результата.

Видео



Команда

Над проектом работали:

- Тимофей Кондратьев [Copter Express] - сборка дрона, написание и отладка программы, проведение тестов;
- Антон Мальцев [Copter Express] - моделирование защиты пропеллеров;
- Андрей Посконин [Texel] - импорт ПО Texel на Raspberry Pi, совместные тесты;

Система распознавания лиц

Введение

В последнее время системы распознавания лиц используются все шире, область применения этой технологии поистине огромна: от обычных селфи-дронов до дронов-полицейских. Ее интеграция в различные устройства проводится повсеместно. Сам процесс распознавания реально завораживает, и это сподвигло меня сделать проект связанный именно с этим. Целью моего стажерского проекта является создание простой open source-ной системы распознавания лиц с квадрокоптера Клевер. Данная программа берет изображения с камеры квадрокоптера, а его обработка происходит уже на компьютере. Поэтому все оставшиеся инструкции выполняются на ПК.

Разработка

Первой задачей было найти алгоритм самого распознавания. В качестве пути решения проблемы было выбрано использовать готовое API для Python. Данное API сочетает в себе ряд преимуществ: скорость и точность распознавания, а также простота использования.

Установка

Для начала нужно установить все необходимые библиотеки:

```
pip install face_recognition
pip install opencv-python
```

Затем скачать сам скрипт из репозитория:

```
git clone https://github.com/mmkuznecov/face_recognition_from_clever.git
```

Объяснение кода

Подключаем библиотеки:

```
import face_recognition
import cv2
import os
import urllib.request
import numpy as np
```

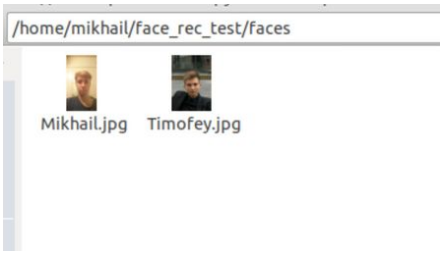
Данный кусок кода предназначен для Python 3. В Python 2.7 подключаем urllib2 вместо urllib:

```
import urllib2
```

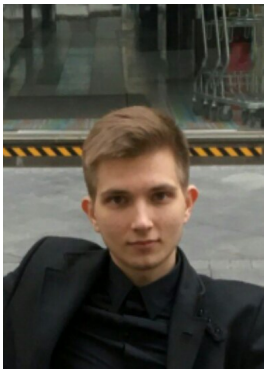
Создаем список кодировок изображений и список имен:

```
faces_images=[]
for i in os.listdir('faces/'):
    faces_images.append(face_recognition.load_image_file('faces/'+i))
known_face_encodings=[]
for i in faces_images:
    known_face_encodings.append(face_recognition.face_encodings(i)[0])
known_face_names=[]url
for i in os.listdir('faces/'):
    i=i.split('.')[0]
    known_face_names.append(i)
```

Дополнение: все изображения хранятся в папке faces в формате name.jpg







Инициализируем некоторые переменные:

```
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
```

Берем изображение с сервера и преобразуем его в cv2 формат:

```
req = urllib.request.urlopen('http://192.168.11.1:8080/snapshot?topic=/main_car
arr = np.asarray(bytearray(req.read()), dtype=np.uint8)
frame = cv2.imdecode(arr, -1)
```

Для Python 2.7:

```
req = urllib2.urlopen('http://192.168.11.1:8080/snapshot?topic=/main_camera/ima
arr = np.asarray(bytearray(req.read()), dtype=np.uint8)
frame = cv2.imdecode(arr, -1)
```

Объяснение дальнейшего кода можно найти на GitHub'е используемого API в комментариях к следующему скрипту

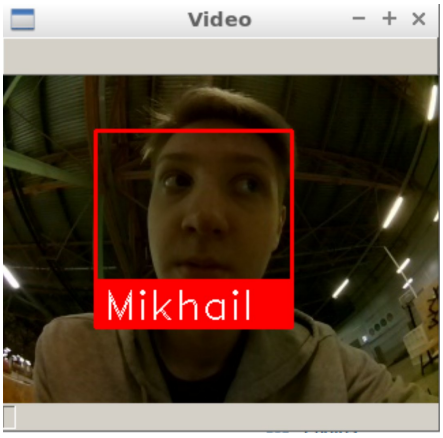
Использование

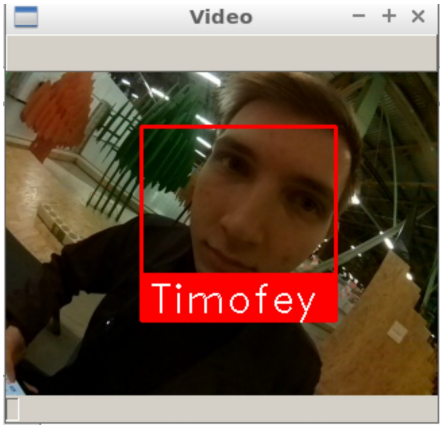
Достаточно подключиться к "Клеверу" через Wi-Fi и проверить, корректно ли работает видеострим с камеры.

Затем просто запускаем скрипт:

```
python recog.py
```

И на выходе:





Возможные трудности

При запуске скрипта может выскочить следующая ошибка:

```
    known_face_encodings.append(face_recognition.face_encodings(i)[0])
IndexError: list index out of range
```

В этом случае постарайтесь переделать изображения в папке faces, возможно из-за плохого качества программа не распознает лиц на изображениях.

Использование калибровки

Для повышения точности распознавания можно использовать калибровку камеры. Модуль для калибровки можно установить, используя специальный пакет. Инструкцию по установке и использованию можно найти в файле calibration.md. Программа с использованием калибровочного пакета называется recog_undist.py

Краткое пояснение кода:

Подключаем установленный пакет:

```
import clever_cam_calibration.clevercamcalib as ccc
```

Добавляем следующие строки:

```
height_or, width_or, depth_or = frame.shape
```

Таким образом получаем информацию о размере изображения, где height_or-это высота оригинального изображения в пикселях, а width_or-ширина. Затем исправляем искажения оригинального изображения и получаем уже его параметры:

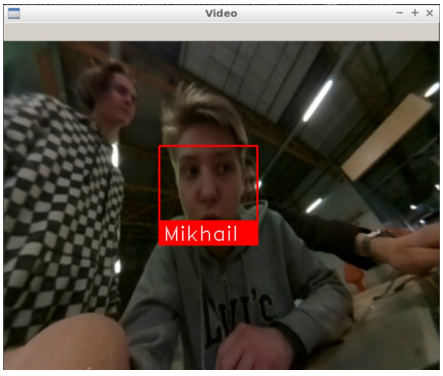
```
if height_or==240 and width_or==320:
    frame=ccc.get_undistorted_image(frame,ccc.CLEVER_FISHEYE_CAM_320)
elif height_or==480 and width_or==640:
    frame=ccc.get_undistorted_image(frame,ccc.CLEVER_FISHEYE_CAM_640)
else:
    frame=ccc.get_undistorted_image(frame,input("Input your path to the .yaml f
height_unz, width_unz, depth_unz = frame.shape
```

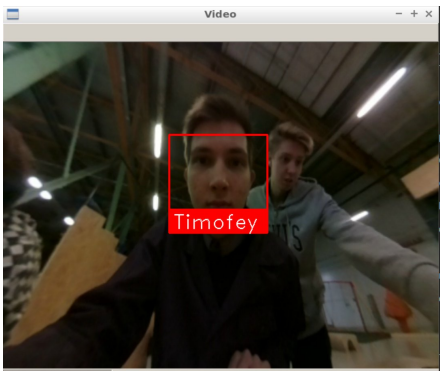
В данном случае мы передаем аргумент ccc.CLEVER_FISHEYE_CAM_640, т.к. разрешение изображения в приведенном примере составляет 640x480, также можно использовать ccc.CLEVER_FISHEYE_CAM_320 для разрешения 320x240, в противном случае необходимо в качестве второго аргумента передать путь до калибровочного .yaml файла.

И, наконец, возвращаем изображение к изначальному размеру:

```
frame=cv2.resize(frame,(0,0), fx=(width_or/width_unz),fy=(height_or/height_unz)
```

Благодаря этому можно значительно повысить точность распознавания, т.к. обрабатываемое изображение будет уже не так сильно искажено.





Управление дроном силой мысли

Введение

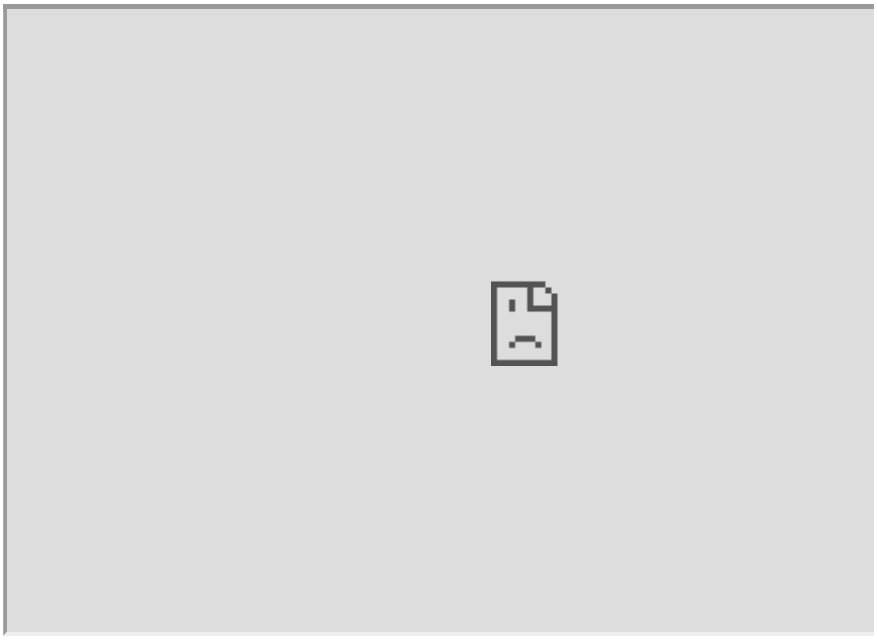
Нейронные сети и дроны достаточно быстро развиваются в мире технологий. В управлении квадрокоптером за основу была взята искусственная нейронная сеть, состоящая из искусственных нейронов (программируемая система, которая имитирует подобие биологических нейронов).

Нейронная сеть разделяется на несколько слоев:

1. входной слой;
2. скрытый слой;
3. выходной слой.

Целью данного проекта было создание управления Клевер 4 с помощью нейрошлема от компании «NEUROBOTICS». Мы использовали NeuroPlay-8M с 8-ми канальной системой беспроводной регистрации ЭЭГ данных человека на (гелевых) электродах. Программное обеспечение имеет поддержку устройств серии Нейробелт с интерфейсом Bluetooth 4.0 (BLE).

Видеодемонстрация:



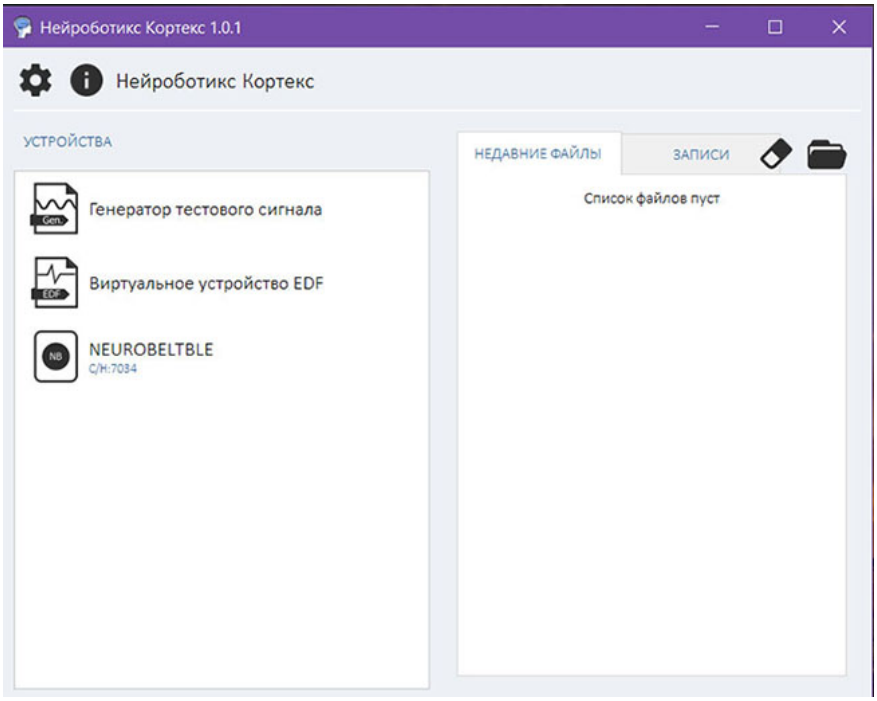
Установка

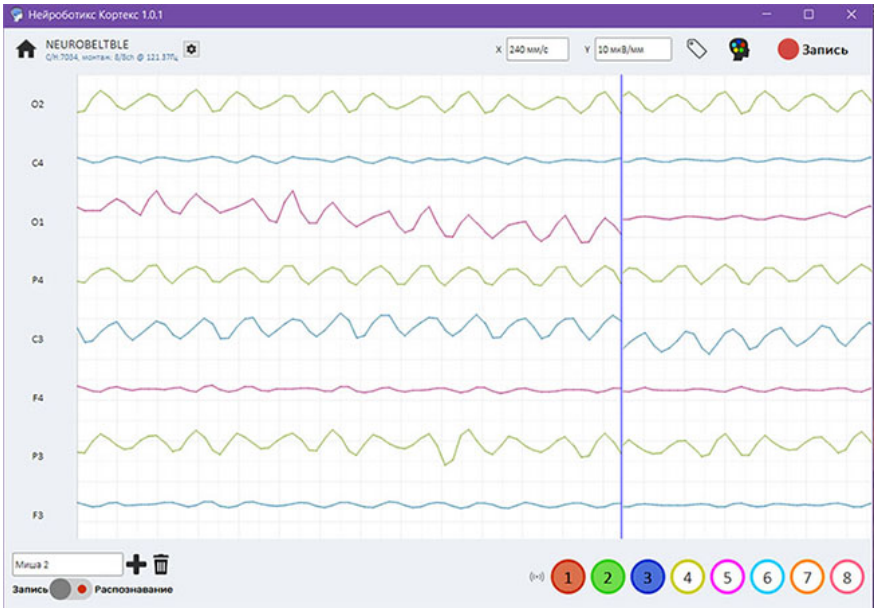
Для начала нужно установить необходимые программы:

1. Cortex.
2. BioEcho.

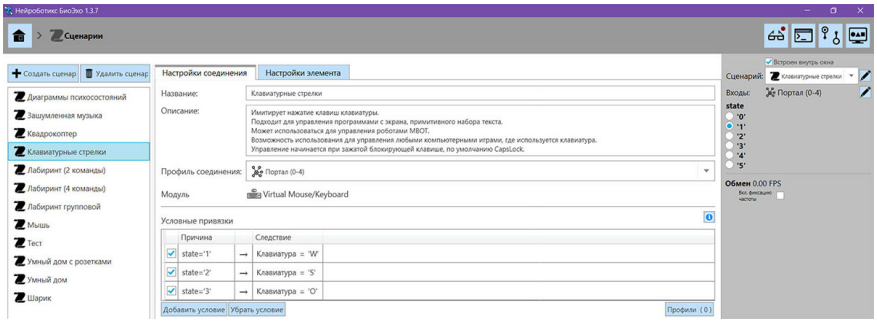
Использование

Программное обеспечение Cortex: классифицирует сигнал мозга в эмоцию.





Программное обеспечение BioEcho: эмуляция физической клавиатуры.



Далее данные поступают на клавиатуру с помощью BioEcho и предаются в ROS.

Основной код работает на Raspberry Pi и передает команды в ROS, имитируя нажатие соответствующих клавиш. Для управления дроном оператор использует устойчивые внутренние физиологические и психологические реакции, при которых произвольное внимание концентрируется на том или ином процессе и состоянии. В рамках реализации проекта были использованы следующие состояния:

1. Напряжение – вперед (клавиша «w»).
2. Радость – yaw на 90° (клавиша «s»).
3. Расслабление – удержание позиции (клавиша «o»).

Для корректного перехода между эмоциями мы использовали «нейтральное» состояние, в качестве которого может выступить любой эмоциональный процесс, который не был задействован в управлении дроном. Также нужно запомнить, что оператор не использует мышцы и движение глаз. Скорость переключения между эмоциями оператора составляет около секунды, а между командами движения более двух секунд.

Код

Код для управления квадрокоптером, написанный на языке Python, находится на сайте GitHub.

Система распознавания и подсчета количества объектов

Введение

Системы компьютерного зрения все шире используются для решения повседневных задач в самых различных сферах, начиная от промышленности, заканчивая медициной. Алгоритмы распознавания образов позволяют идентифицировать объекты, определять их тип, предоставляют необходимую качественную и количественную информацию. Целью данного проекта было создание системы подсчета количества объектов (людей, машин), используя технологии компьютерного зрения.

Установка

Для начала нужно установить все необходимые библиотеки:

```
pip install opencv-contrib-python imutils matplotlib dlib
```

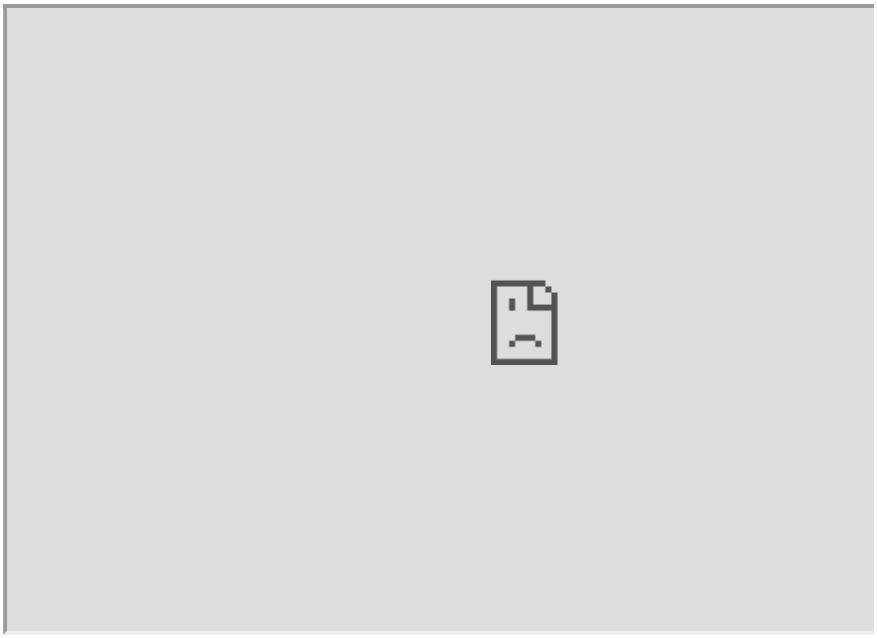
Библиотека dlib устанавливается достаточно долго, так что не стоит пугаться того, что процесс зависает.

Затем скачать программу из репозитория:

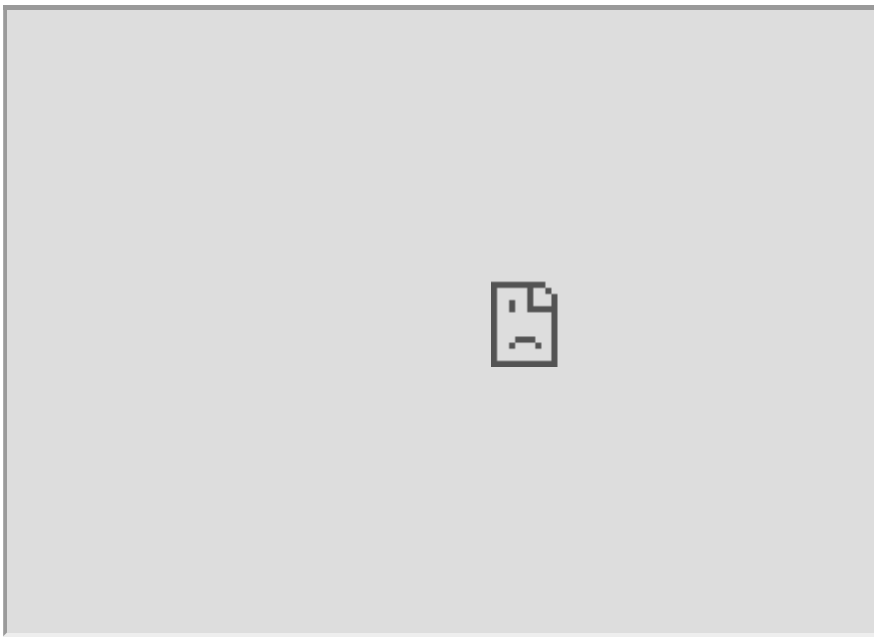
```
git clone https://github.com/mmkuznecov/objects_counting_from_clever.git
```

Примеры работы

Система может быть адаптирована для разных условий, съёмка может производиться как из статичного положения:



Так и в динамическом полёте:



Использование

Подключитесь к Клеверу и проверьте, передается ли изображение с камеры. Для использования скрипта просто перейдите в папку, куда был скачан скрипт и пропишите в консоли, где op - название выходного видео:

```
python count.py -o op.avi
```

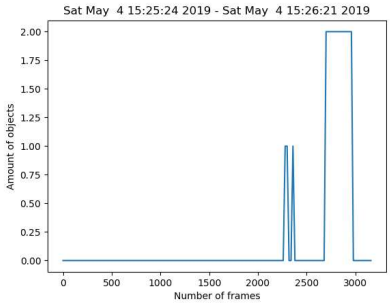
По умолчанию программа будет считать количество людей. Для того чтобы прописать распознавание только определенного объекта, нужно приписать в конце тип объекта для распознавания, например, следующая команда позволит распознавать машины:

```
python count.py -o op.avi -t car
```

Полный список распознаваемых объектов приведен ниже:

background, aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor

Итогом работы программы будет записанное видео с метками, отмечающими распознавание объектов и их номер, а также .png изображение, на котором показана информация о времени записи видео, количестве распознанных объектов в разное время.



По вопросам пишите в Telegram @mmkuznecov.

Версия с HD-Link

Если имеется настроенная система HD-Link для передачи видео в хорошем качестве на большие расстояния, можно использовать следующие скрипты для обработки поступающего видео.

Для установки и настройки оборудования можно обратиться к документации. Там можно найти подробную информацию о настройке конфигурации, а также ссылку для скачивания образа для Raspberry Pi. Подключиться к GroundPi можно либо по Ethernet, либо по Wi-Fi, предварительно настроив в образе соответствующие параметры подключения. Во втором случае, после подключения к питанию GroundPi, должна появиться сеть Open.HD. Пароль для подключения к ней **wifiopenhd**

Для поштучного подсчета

Для запуска этой версии необходимо скачать программу из репозитория:

```
git clone https://github.com/mmkuznecov/HD_Link_counting.git
```

На пк должен быть установлен Python. Для установки всех необходимых модулей, нужно перейти в папку, куда была скачана программа и, чтобы установить все необходимые модули, прописать в консоли:

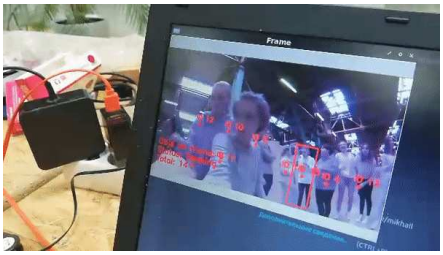
```
pip install -r requirements.txt
```

Перед тем, как запускать программу, надо убедиться, что ПК связан с GroundPi по Ethernet или Wi-Fi.

Если работать из под операционной системы Windows, чтобы запустить работу программы, достаточно дважды кликнуть по файлу run.bat.

На обработанных кадрах отображается количество распознанных с начала работы объектов (Total), а также количество объектов в конкретном кадре (Objs on frame).

Пример работы:



На изображении есть еще один параметр - Status, но он был убран.

Для обработки больших групп людей

Скачиваем программу из репозитория:

```
git clone https://github.com/mmkuznecov/HD_Link_crowd.git
```

Аналогичным образом устанавливаем все необходимые модули через requirements.txt, проверяем подключение, запускаем программу. Также надо будет скачать модель и поместить ее в папку.

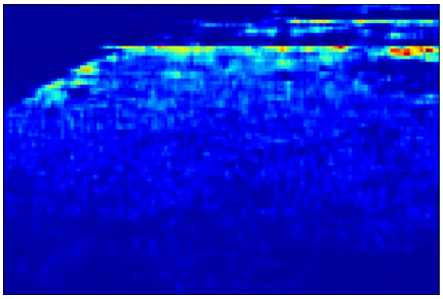
Через определенные промежутки времени, программа сохраняет снимки, полученные с камеры, при этом обрабатывет их и выводит данные о количестве людей.

Пример работы:

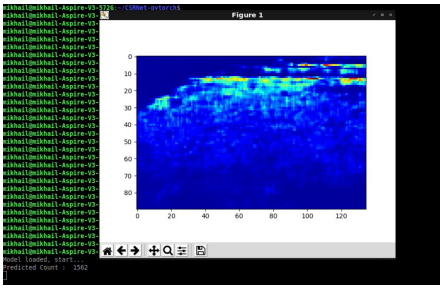
Оригинал изображения:



Тепловая карта плотности скопления людей:



В итоге получается примерно 1562 человека:



Пульт на Android

Все владельцы мобильных устройств фирмы *Apple* ещё морозным январем 2018го обзавелись приятным приложением под *iOS* для пилотирования квадрокоптеров с помощью **WiFi**. И вот, спустя год вышло такое же приложение но уже для другой операционной системы. Актуальную версию вы можете скачать **тут** .

Введение

В данной статье я расскажу вам о том, как можно написать свой или доработать уже имеющийся пульт для Андроид своими руками. Для работы будем использовать модный язык *Kotlin*, а в качестве среды разработки возьмем *Android Studio*. Для тех кто ни разу ей не пользовался рекомендую к ознакомлению следующие *материалы*. Весь код приложения можно найти **тут**. Если вы хотите сразу получить приложение с целью дальнейшей доработки, выполните следующую команду:

```
git clone https://github.com/Tennessium/android
```

Однако чтобы вы смогли полностью понять устройство приложения, я расскажу вам о каждом этапе создания проекта, как если бы вы делали его с нуля.

Обертка

Начнем с самого простого - внешнего вида нашего приложения. На **гитхабе** вы можете найти *HTML*, *CSS* и *JavaScript* файлы, это и есть веб страница с которой будет происходить управление коптером. Чтобы эта страница отображалась у нас в приложении надо:

1. Создать папку **assets** в главной папке приложения **app**
2. Добавить в нее файлы все файлы отсюда

Если вы дошли до этого этапа то у вас уже есть необходимая веб страница, поздравляю! Теперь нам надо её как-то отобразить в приложении. Для этого в классе вашего *activity* в методе **onCreate** необходимо написать следующий код:

```
main_web.loadUrl("file:///android_asset/index.html")
```

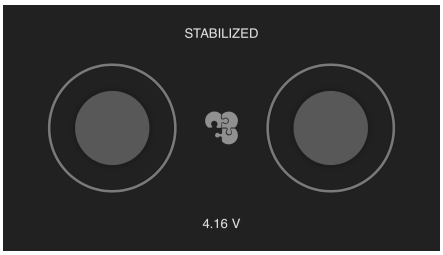
Где *main_web* - id вашего *WebView*, который должен находится в *xml* файле выбранного вами *activity*.

К сожалению, пульт для управления коптером требует всего экрана устройства, а элементы интерфейса системы мешают полноценному использованию программы. Для этого надо в начале метода **onCreate** вызвать следующую функцию:

```
private fun fullScreenCall() {
    window.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.L
    if (Build.VERSION.SDK_INT < 19) {
        val v = this.window.decorView
        v.systemUiVisibility = View.GONE
    } else {
        //for higher api versions.
        val decorView = window.decorView
        val uiOptions = View.SYSTEM_UI_FLAG_HIDE_NAVIGATION or View.SYSTEM_UI_F
        decorView.systemUiVisibility = uiOptions
    }
}
```

Данная функция позволяет избавиться от элементов интерфейса системы. Идем дальше.

Вот так выглядит пульт на этом этапе:



Если вы запустите приложение, то заметите что стики не работают. Это происходит по тому, что на нашей странице отключен *JavaScript*. чтобы его включить надо прописать следующее:

```
    main_web.settings.apply {
    domStorageEnabled = true
    javaScriptEnabled = true
    loadWithOverviewMode = true
    useWideViewPort = true
    setSupportZoom(false)
}
```

Этим куском кода мы разрешаем странице использовать *JavaScript* и заодно готовимся к следующему этапу - **логике**.

Прием данных с веб страницы

Чтобы телефон мог принимать данные с *HTML страницы*, надо создать класс для взаимодействия с веб интерфейсом

```
class WebAppInterface(c: Context) {
    @JavascriptInterface
    public fun postMessage(message: String) {
        val data = JSONObject(message)
        send("255.255.255.255", 35602, pack(
        data.getInt("x").toShort(),
        data.getInt("y").toShort(),
        data.getInt("z").toShort(),
        data.getInt("r").toShort()))
    }
}
```

Данный класс будет получать сообщение с веб страницы отправленное методом *postMessage*, где аргумент *message* - сообщение от страницы.

Теперь надо связать классы **WebAppInterface** и **MainActivity**. Для этого надо всего лишь добавить одну строку в метод **onCreate**:

```
main_web.addJavascriptInterface(WebAppInterface(this), "appInterface")
```

Отправка данных на коптер

Важно! Для любой работы с интернетом на платформе *Android* в файле **AndroidManifest.xml** внутри тега *manifest* необходимо добавить такую строку:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Она дает вашему приложению доступ в интернет и возможность передавать данные по средствам **WiFi**. А как это делать, мы с вами сейчас и узнаем. Идём дальше!

Вы наверное заметили функцию *send* в классе **WebAppInterface**. Именно она отправляет данные на коптер. Давайте объявим ее **вне классов**:

```
fun send(host: String, port: Int, data: ByteArray, senderPort: Int = 0): Boolea
    var ret = false
    var socket: DatagramSocket? = null
    try {
        socket = DatagramSocket(senderPort)
        val address = InetAddress.getByName(host)
        val packet = DatagramPacket(data, data.size, address, port)
        socket.send(packet)
        ret = true
    } catch (e: Exception) {
        e.printStackTrace()
    } finally {
        socket?.close()
    }
    return ret
}
```

Данная функция отправляет данные при помощи *протокола пользовательских датаграмм* на коптер. Программа отправляет **байты**, поэтому неплохо бы было объявить функцию для создания массива **байтов** из четырех переменных:

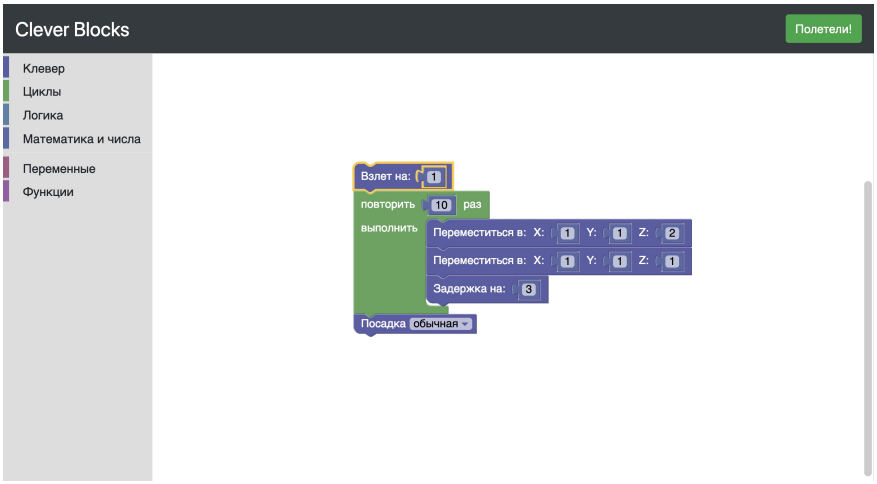
```
fun pack(x: Short, y: Short, z: Short, r: Short): ByteArray {
    val pump_on_buf: ByteBuffer = ByteBuffer.allocate(8)
    pump_on_buf.putShort(r)
    pump_on_buf.putShort(z)
    pump_on_buf.putShort(y)
    pump_on_buf.putShort(x)
    return pump_on_buf.array().reversedArray()
}
```

Итог

Теперь ваше приложение имеет полный функционал аналога под **iOS**. Можете кастомизировать его как пожелаете. По любым вопросам о приложении можете обращаться в Телеграм @Tenessinum.

Блочное программирование Клевера

> В этой статье описана неофициальная реализация блочного программирования. Официальная поддержка появилась в образе Клевера версии **0.21**.



В этой статье я опишу процесс скачивания, установки и запуска блочного конструктора программ для квадрокоптера Клевер.

Скачивание

Есть два варианта скачивания кода проекта на RPi:

Вариант 1

Подключить плату к интернету (вставив в нее ethernet-кабель или перенастроив Wi-Fi) и в командной строке RPi выполнить:

```
cd
git clone https://github.com/garinegor/clever-blocks
```

Вариант 2

Исходный код проекта можно скачать на компьютер с GitHub, а затем скопировать его в домашнюю директорию RPi посредством SFTP или SCP.

Установка

Если Вы хотите, чтобы блочный конструктор запускался автоматически, необходимо создать соответствующий сервис. Для этого следует ввести в командную строку RPi следующие команды:

```
sudo systemctl enable /home/pi/clever-blocks/service/
sudo systemctl start clever-blocks.service
```

Все готово! Теперь можно переходить к использованию.

Использование

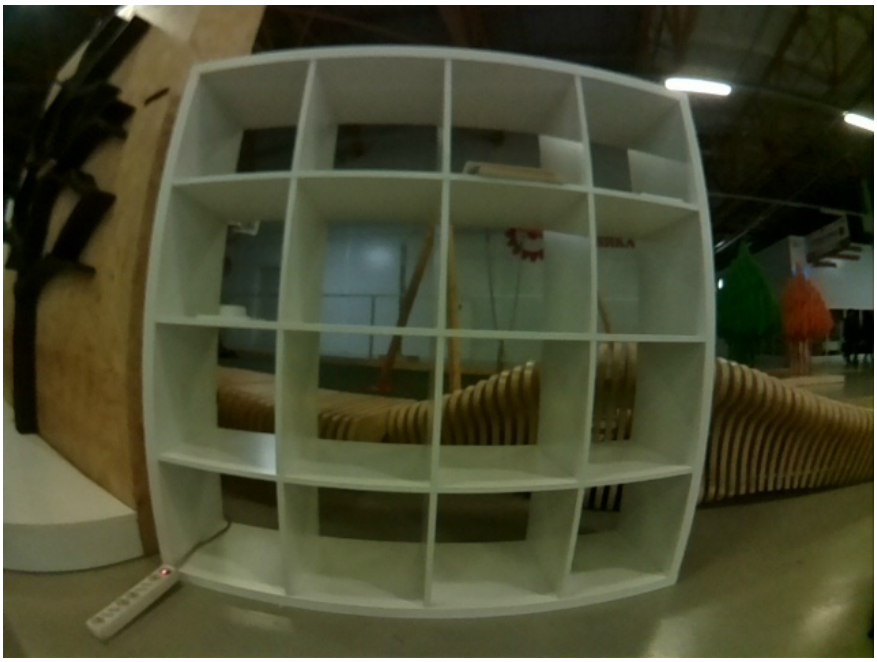
Если Вы не стали добавлять сервис для автоматического старта сервера, для его запуска Вам нужно ввести следующую команду, находясь в папке проекта:

```
python main.py
```

После запуска Вы можете открыть веб-интерфейс для блочного программирования по адресу 192.168.11.1:5000.

Калибровка камеры

Для точной работы систем компьютерного зрения (например, для навигации по ArUco-маркерам) используемая камера должна быть откалибрована.



> Изображение "скруглено" ближе к краям. Какой-либо алгоритм компьютерного зрения будет воспринимать информацию с этой картинки неправильно.

Установка приложения

Для начала, необходимо установить необходимые библиотеки:

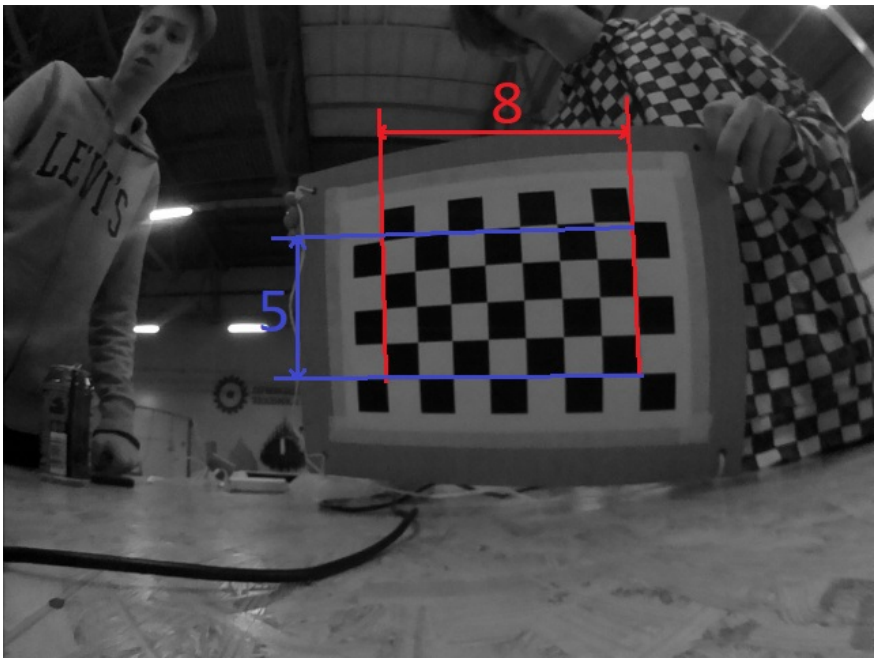
```
pip install numpy
pip install opencv-python
pip install pyyaml
pip install urllib2
pip install flask, flask-wtf
```

Затем скачиваем исходный код из репозитория и проводим установку:

```
git clone https://github.com/tinderad/calibration_web_2.7.git
cd calibration_web_2.7.git
sudo python setup.py build
sudo python setup.py install
```

Подготовка к калибровке

Вам необходимо подготовить калибровочную мишень. Она представляет собой «шахматную доску». Файл можно взять отсюда. Наклейте распечатанную мишень на любую твердую поверхность. Посчитайте количество пересечений в длину и в ширину доски, измерьте размер клетки (в мм), как указано на изображении.



Включите Клевер и подключитесь к его Wifi.

> Перейдите на *192.168.11.1:8080* и проверьте, получает ли компьютер изображения из топика *image_raw*.

Калибровка

Подключитесь к Клеверу по протоколу SSH (например, при помощи PuTTY).

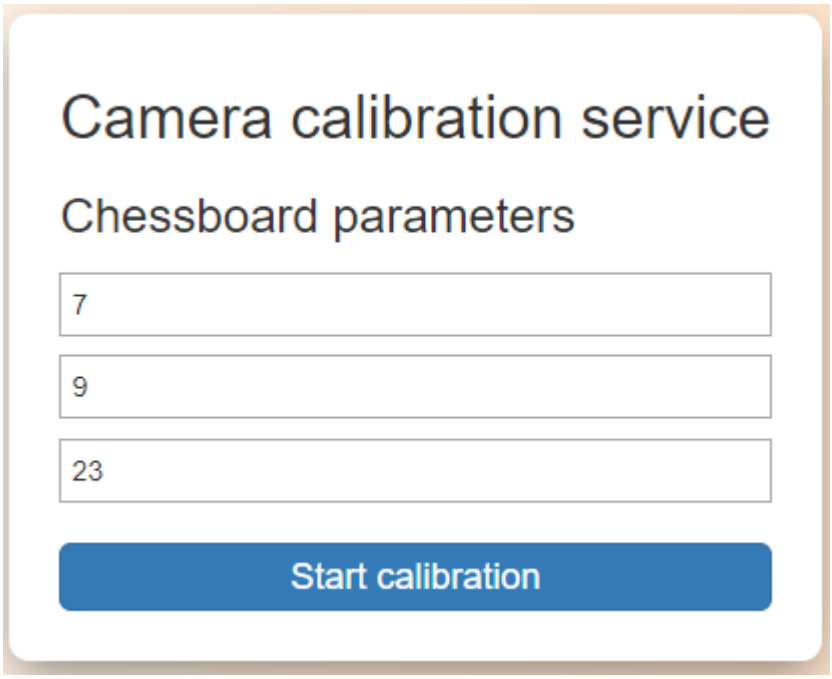
Запустите приложение:

```
>cd calibration_web_2.7/ccc_server
>python app.py
```

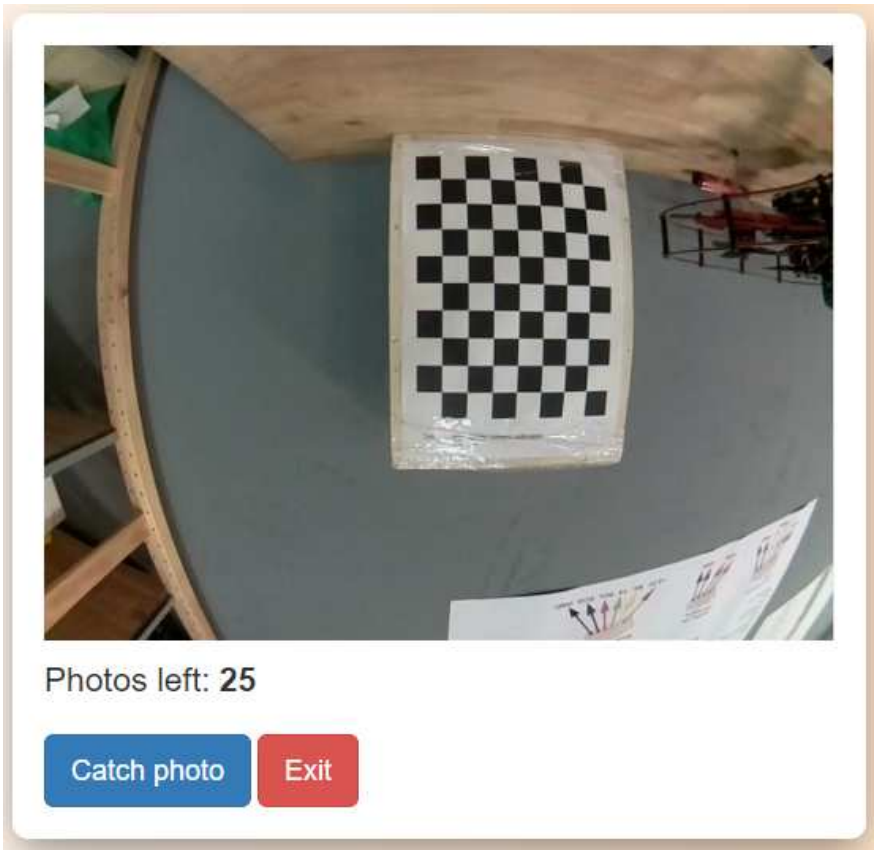
Далее вам необходимо на компьютере открыть в браузере страницу по адресу *192.168.11.1:8081*

> Порт можно настроить в файле *ccc_server/config.py*.

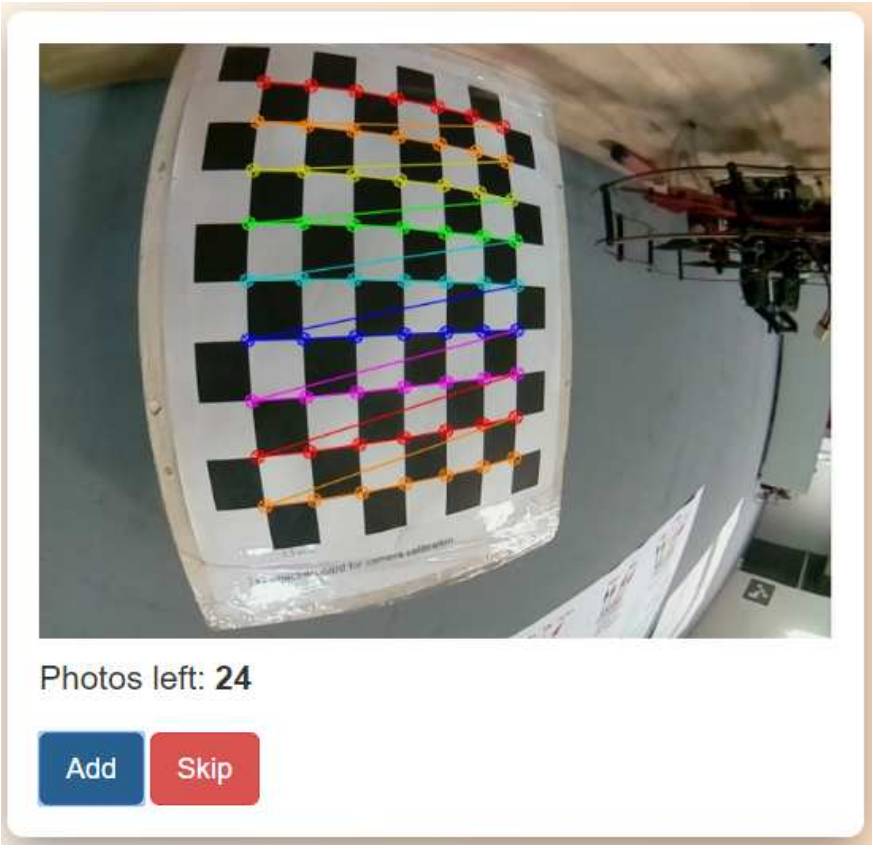
На открытой странице необходимо ввести параметры калибровочной мишени: количество перекрестий в длину и ширину, длину ребра квадрата. Для начала калибровки нажмите кнопку ***Start Calibration***.



На следующей странице при помощи кнопки ***Catch photo*** можно делать фотографии калибровочной мишени.



Если программа нашла на изображении указанную мишень, откроется страница, на которой вам необходимо подтвердить корректность найденных перекрестий.



Если перекрестия были распознаны правильно, нажмите на клавишу ***Add***, и перейдите к получению новых фотографий. В противном же случае, если перекрестия были распознаны некорректно, пропустите данную фотографию при помощи клавиши ***Skip***.

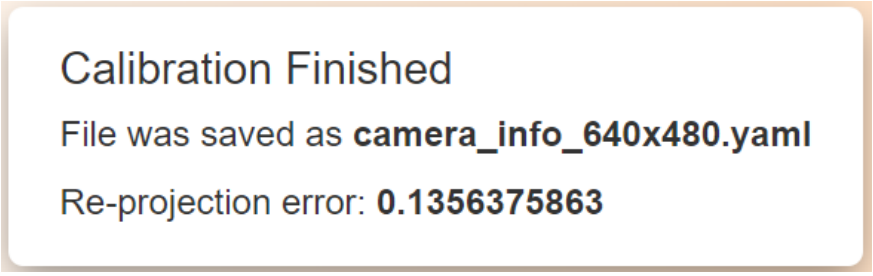
> В большинстве случаев найденные углы будут подсвечиваться разными цветами, но иногда подсветка будет становиться красной. это происходит в том случае, если углы распознаны, но неточно.

Чтобы откалибровать камеру, вам требуется сделать как минимум 25 фото шахматной доски с различных ракурсов. После преодоления данного порога появится кнопка ***Finish***, по нажатию на которую начнется генерация калибровочного файла.

> Это может занять некоторое время.

На открывшейся странице выведется информация о результате калибровки: имя файла и re-projection error.

> re-projection error - отклонение от стандартной математической модели. Чем эта величина меньше, тем точнее проведена калибровка.



Программа обработает все полученные фотографии, и создаст ***.yaml*** файл в нынешней директории. При помощи этого файла можно будет выравнивать искажения на изображениях, полученных с этой камеры.

> Если вы поменяете разрешение получаемого изображения, вам нужно будет снова калибровать камеру.

Предыдущая версия

Также вы можете воспользоваться предыдущей версией программы, не имеющей web-интерфейса.

Запустите скрипт ***calibrate_cam***:

```
>calibrate_cam
```

Задайте параметры мишени:

```
>calibrate_cam
Chessboard width:  # Перекрестий в ширину
Chessboard height:  # Перекрестий в длину
Square size:  # Длина ребра клетки (в мм)
Saving mode (YES - on):  # Режим сохранения
```

> Режим сохранения: если включен, то все полученные фотографии будут сохраняться в нынешней директории.

Скрипт начнет свою работу:

```
Calibration started!
Commands:
help, catch (key: Enter), delete, restart, stop, finish
```

Чтобы откалибровать камеру, вам требуется сделать как минимум 25 фото шахматной доски с различных ракурсов.

Чтобы сделать фото, введите команду ***catch***.

```
>catch
```

Программа будет информировать вас о состоянии калибровки.

```
...
Chessboard not found, now 0 (25 required)
>  # Enter
---
Image added, now 1 (25 required)
```

> Вместо того, чтобы каждый раз вводить команду , Вы можете просто нажимать клавишу ***Enter*** (вводить пустую строку).

После того, как будет набрано достаточное количество изображений, введите команду ***finish***.

```
...
>finish
Calibration successful!
```

Калибровка по существующим изображениям:

Если же у вас уже есть изображения, то вы можете откалибровать камеру по ним при помощи скрипта ***calibrate_from_dir***.

```
>calibrate_from_dir
```

Указываем характеристики мишени, а так же путь до папки с изображениями:

```
>calibrate_cam_ex
Chessboard width:  # Перекрестий в ширину
Chessboard height:  # Перекрестий в длину
Square size:  # Длина ребра клетки (в мм)
Path:  # Путь до папки с изображениями
```

В остальном этот скрипт работает аналогично ***calibrate_cam***.

Программа обработает все полученные фотографии, и создаст файл ***camera_info.yaml*** в нынешней директории. При помощи этого файла можно будет выравнивать искажения на изображениях, полученных с этой камеры.

Исправление искажений

За получение исправленного изображения отвечает функция **clever*cam_calibration._get_undistorted_image(cv2_image, camera_info)***:

- ***cv2_image***: Закодированное в массив cv2 изображение.
- ***camera_info***: Путь до файла калибровки.

Функция возвращает массив cv2, в котором закодировано исправленное изображение.

> Если вы используете fisheye-камеру, поставляемую вместе с Клевером, то для обработки изображений разрешением 320x240 или 640x480 вы можете использовать уже существующие параметры калибровки. Для этого в качестве аргумента передайте параметры ***clever_cam_calibration.CLEVER_FISHEYE_CAM_320*** или ***clever_cam_calibration.CLEVER_FISHEYE_CAM_640*** соответственно.

Примеры работы

Изначальные изображения:



Исправленные изображения:



Пример использования

Обработка потока изображений с камеры.

Данная программа получает изображения с камеры Клевера и выводит их на экран в исправленном виде, используя существующий калибровочный файл.

```
import clever_cam_calibration as ccc
import cv2
import urllib.request
import numpy as np
while True:
    req = urllib.request.urlopen('http://192.168.11.1:8080/snapshot?topic=/mair
    arr = np.asarray(bytearray(req.read()), dtype=np.uint8)
    image = cv2.imdecode(arr, -1)
    undistorted_img = ccc.get_undistorted_image(image, ccc.CLEVER_FISHEYE_CAM_(
    cv2.imshow("undistort", undistorted_img)
    cv2.waitKey(33)
cv2.destroyAllWindows()
```

Использование для ArUco

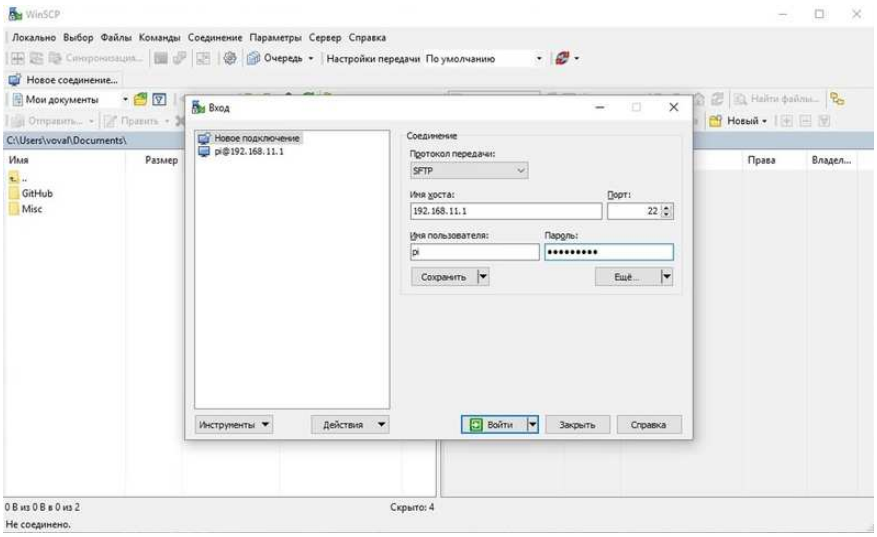
Чтобы применить параметры калибровки к системе ArUco-навигации, требуется перенести калибровочный .yaml файл на Raspberry Pi Клевера и инициализировать его.

> Не забудьте подключиться к WiFI Клевера.

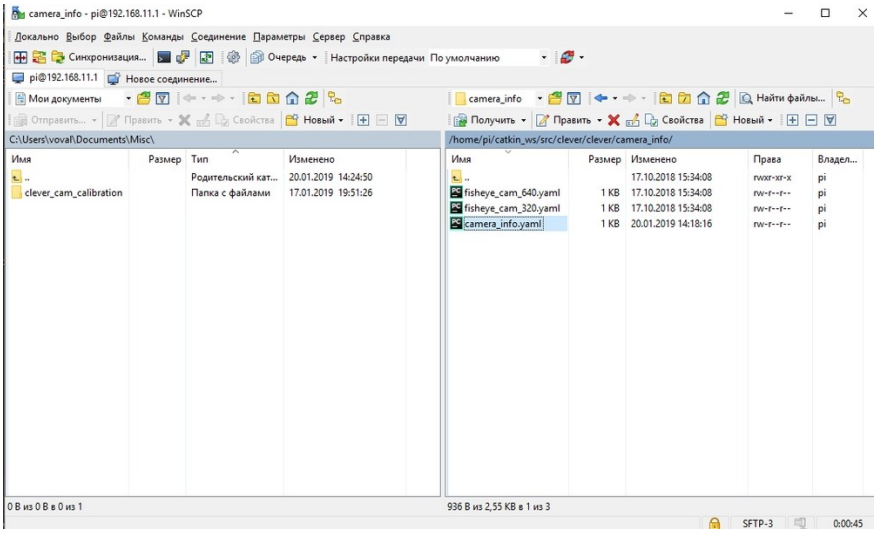
Для передачи файла используется протокол SFTP. В данном примере используется программа WinSCP.

Подключимся к Raspberry Pi по SFTP:

> Пароль:

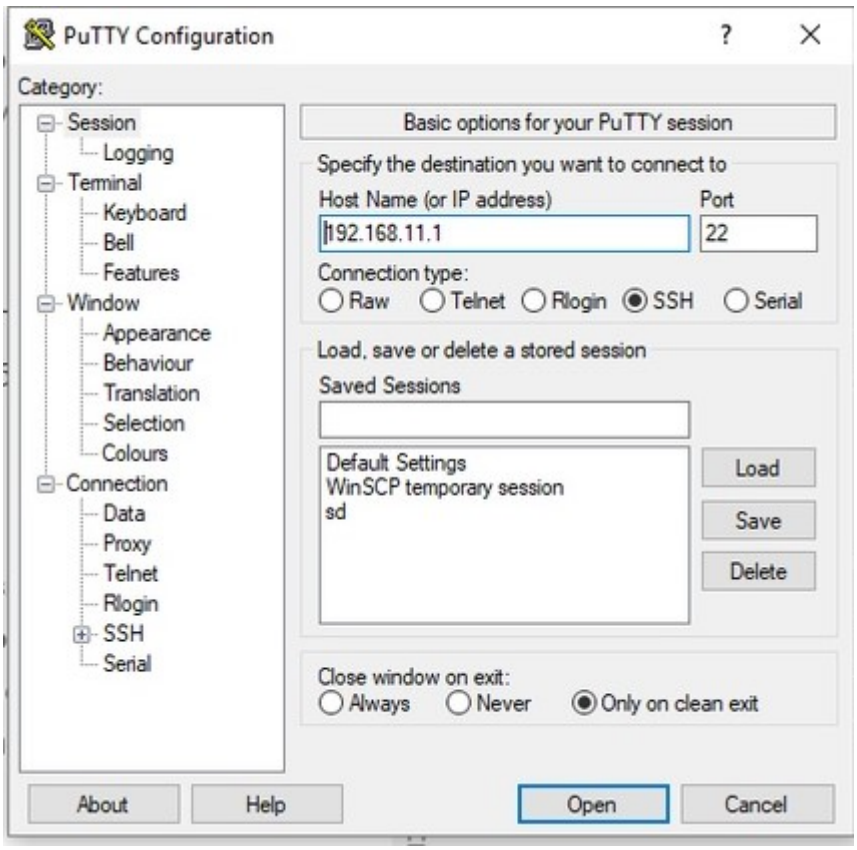


Нажимаем “Войти”. Переходим в ***/home/pi/catkin_ws/src/clover/clover/camera_info/*** и копируем туда калибровочный .yaml файл:

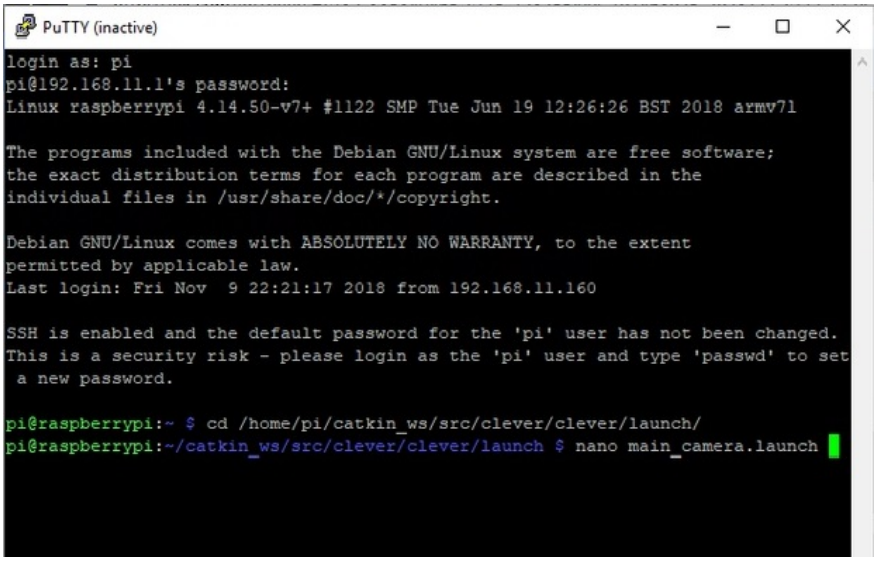


Теперь мы должны выбрать этот файл в конфигурации ArUco. Для этого используется связь по протоколу SSH. В данном примере используется программа PuTTY.

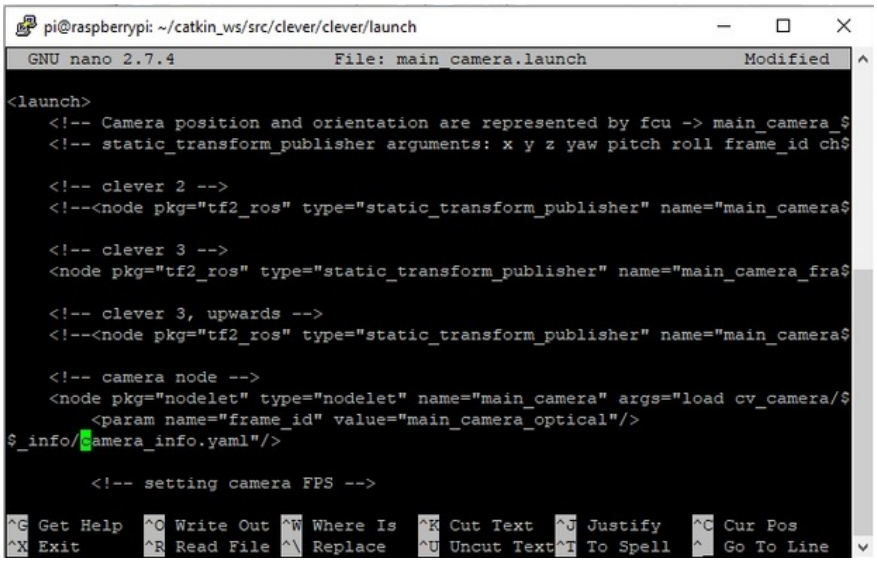
Подключимся к Raspberry Pi по SSH:



Войдем под логином ***pi*** и паролем ***raspberry***, перейдем в директорию ***/home/pi/catkin_ws/src/clover/clover/launch*** и начнем редактировать конфигурацию ***main_camera.launch***:



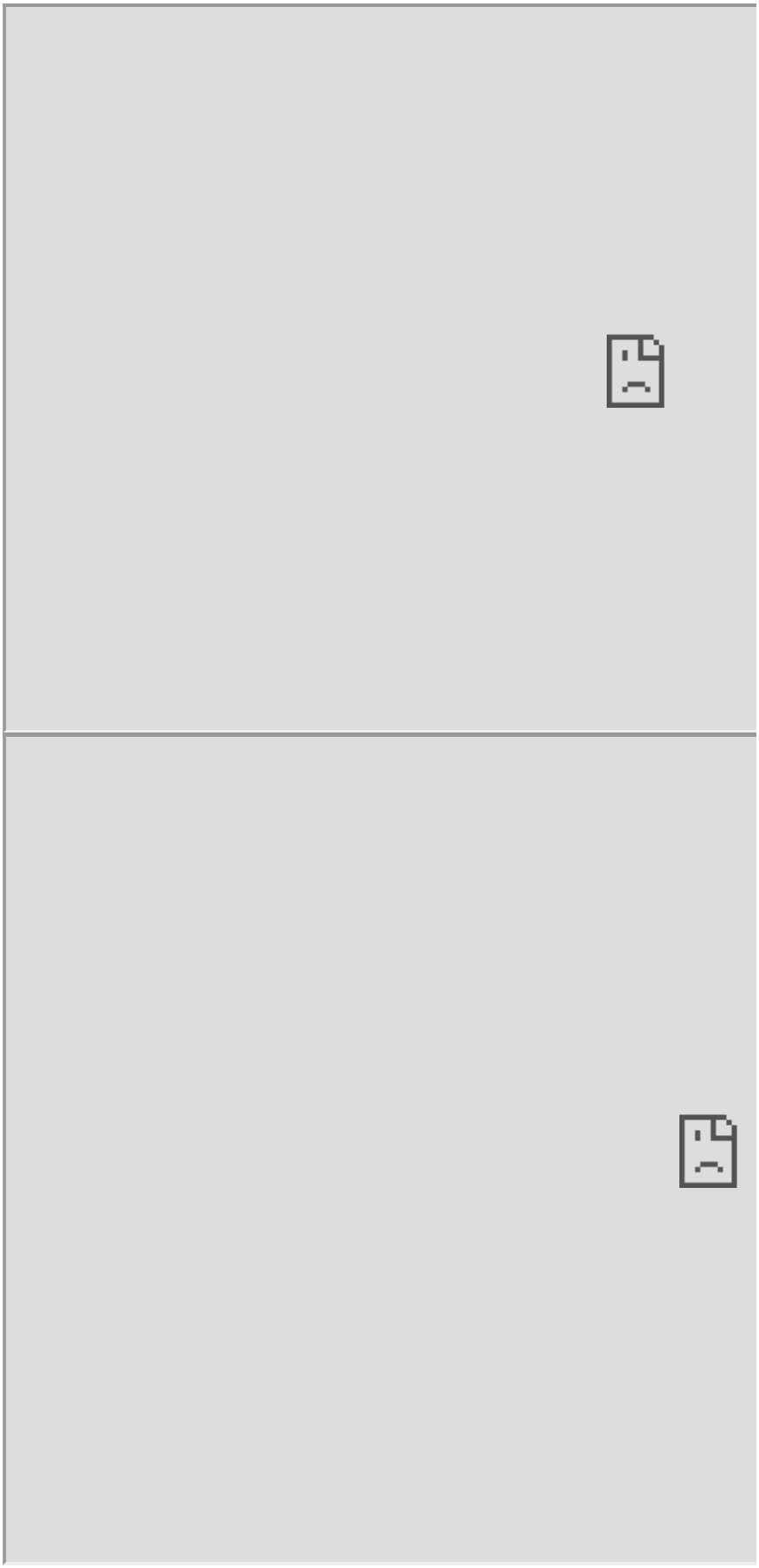
В строке ***camera node*** заменим параметр ***camera_info*** на ***camera_info.yaml***:



> Не забудьте изменить разрешение камеры в *main_camera.launch*.

Управление дроном при помощи позы человека

Демонстрация



Ссылки на литературу

- Руководство по оценке позы человека
- Умные беспилотники учебники
- Posnet GitHub РЕПО
- Posnet Medium артикул
- Tensorflow.js Демонстрация
- Posenet обзор

Распознавание видов агрокультур в массовом сельском производстве

Введение

Современное сельское хозяйство во многих странах превращается в один из ярких примеров быстрого и успешного внедрения новых технологий. Беспилотные летательные аппараты способны выполнять широкий круг задач, среди которых мониторинг сельскохозяйственных угодий сегодня стал уже почти привычным инструментом повышения эффективности сельских хозяйств. Целью моего проекта является написание кода для распознавания видов агрокультур в массовом сельском производстве. В дальнейшем из результатов распознавания можно спроектировать карту посевных площадей.

Мониторинг

В сельском хозяйстве мониторинг необходим для получения информации о состоянии угодий и посевов. Фермеры или специалисты могут по результатам мониторинга понять, нормально ли всходят культуры, есть ли угроза со стороны сорняков и/или насекомых – вредителей, какова степень увлажненности отдельных участков или целых площадей и т.д.

Объяснение кода

Подключаем библиотеки:

```
import rospy
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import numpy as np
```

Создаём некоторые переменные:

```
rospy.init_node('computer_vision_sample')

bridge = CvBridge()

color = 'undefined'
shape = 'undefined'
culture = ""
```

Для реализации алгоритмов компьютерного зрения рекомендуется использовать предустановленную на образ Клевера библиотеку OpenCV. Создаём подписчика на топик с изображением с основной камеры для

обработки с использованием OpenCV:

```
def image_colback_color(data):
    global color, shape

    cv_image = bridge.imgmsg_to_cv2(data, 'bgr8') # OpenCV image
    img_hsv = cv2.cvtColor(cv_image, cv2.COLOR_BGR2HSV) #[118:119,158:159]

    #detected color
    #print(img_hsv[0][0])
```

Каждая культура имеет свой неповторимый оттенок(пшеница- золотистая, гречиха - светло-коричневая).





Прописываем диапазоны цветов для определённых культур:

```
#пшеница
yellow_orange_low = (38, 110, 150)
yellow_orange_high= (52, 110, 150)

#гречиха
brown_low = (23, 50, 50)
brown_high= (37, 50, 50)

yellow_orange_mask = cv2.inRange(img_hsv, yellow_orange_low, yellow_orange_high
brown_mask = cv2.inRange(img_hsv, brown_low, brown_high)

if yellow_orange_mask[119][159] == 255:
    shape = shape_recog(yellow_orange_mask)

elif brown_mask[119][159] == 255:
    shape = shape_recog(brown_mask)

else:
    shape = 'undefined'
    color = 'undefined'

if shape == 'brown':
    culture = "greshiha"
if shape == 'yellow_orange':
    culture = "pshenitsa"

image_sub = rospy.Subscriber('main_camera/image_raw', Image, image_colback_colo
```

Скрипт будет занимать 100% процессора. Для искусственного замедления работы скрипта можно запустить throttling кадров с камеры, например, в 5 Гц (`main_camera.launch`):

```
<node pkg="topic_tools" name="cam_throttle" type="throttle" args="messages mair
```

Топик для подписчика в этом случае необходимо поменять на: `main_camera/image_raw_throttled` .

```
print (culture)
while not rospy.is_shutdown():
    print("color: {}".format(color))
    print("shape: {}".format(shape))
    rospy.sleep(0.2)
```

Данная программа будет определять культуру по её оттенку. Для повышения точности определения можно использовать больше цветовых диапазонов и дрон сможет распознавать большее количество культур.

Вот примеры цветовых диапазонов:

```
red_low1 = (0, 110, 150)
red_high1 = (7, 255, 255)

red_low2 = (172, 110, 150)
red_high2 = (180, 255, 255)

red_orange_low = (8, 110, 150)
red_orange_high = (22, 110, 150)

orange_low = (23, 110, 150)
orange_high = (37, 110, 150)

yellow_orange_low = (38, 110, 150)
yellow_orange_high = (52, 110, 150)

yellow_low = (53, 150, 150)
yellow_high = (67, 255, 255)

yellow_green_low = (68, 150, 150)
yellow_green_high = (82, 255, 255)

green_low = (83, 150, 150)
green_high = (97, 255, 255)

blue_green_low = (98, 150, 150)
blue_green_high = (113, 255, 255)

blue_low = (114, 150, 150)
blue_high = (127, 255, 255)

blue_violet_low = (128, 150, 150)
blue_violet_high = (142, 255, 255)

violet_low = (143, 150, 150)
violet_high = (157, 255, 255)

red_violet_low = (158, 150, 150)
red_violet_hugh = (171, 255, 255)
```

Обратите внимание, что для красного цвета используется два диапазона т. к. красный цвет находится на границах цветового пространства HSV.

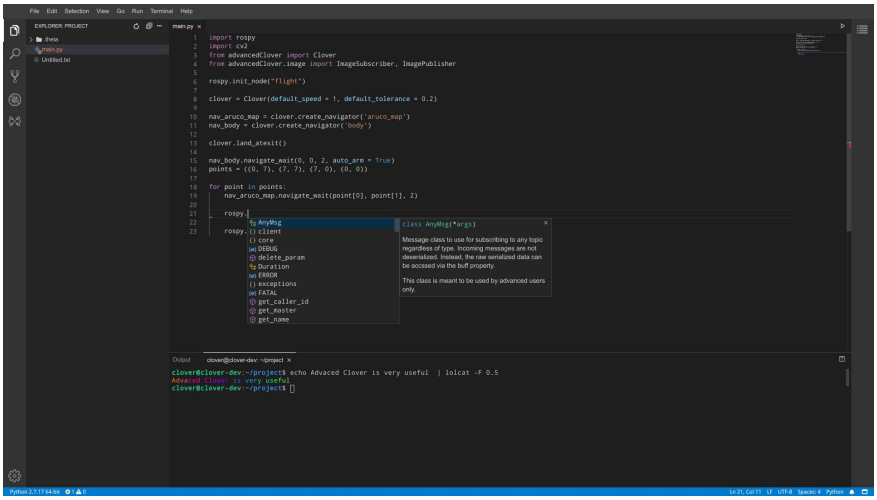
Advanced Clover

CopterHack-2021, team **FTL**. Контакты: @maximmasterr в Telegram.

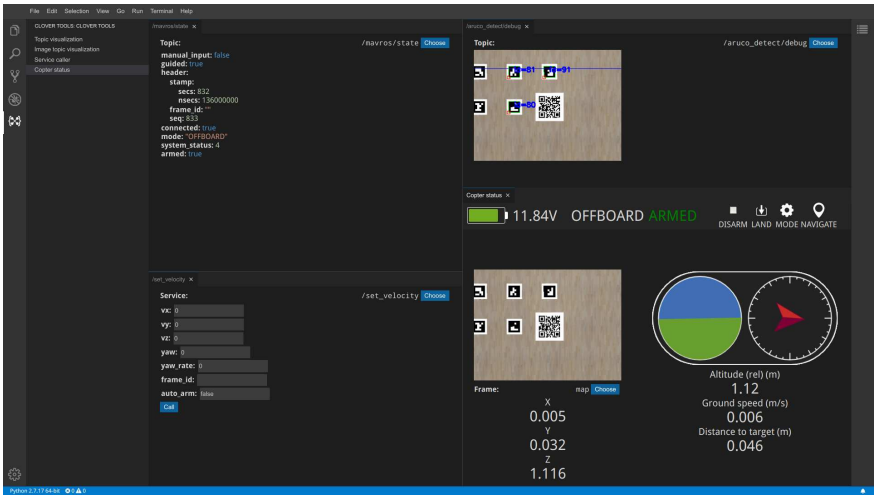
Как-то раз устав выполнять некоторые действия с Клевером мы решили их упростить, и так появился проект advancedClover, которой включает в себя следующие элементы:

- IDE
  - VSCode-like IDE в браузере
  - Интегрированный файловый менеджер
  - Интегрированный терминал
  - Автокомплит
  - Возможность просмотра топиков
  - Возможность просмотра топиков с изображениями
  - Возможность вызова сервисов
  - Возможность просмотра статуса коптера
- Библиотека для Python
  - Упрощение работы с навигацией
  - Упрощение работы с камерой

Пример автокомплита, терминала и файлового менеджера:



Пример инструментов ROS и Copter Status:



Пример полёта по квадрату с использованием advancedClover:

```
import rospy
import cv2
from advancedClover import Clover
from advancedClover.image import ImageSubscriber,
ImagePublisher
rospy.init_node("flight")

clover = Clover(default_speed = 1, default_tolerance = 0.2)

nav_aruco_map = clover.create_navigator('aruco_map')
nav_body = clover.create_navigator('body')

clover.land_atexit()

nav_body.navigate_wait(0, 0, 2, auto_arm = True)
points = ((0, 7), (7, 7), (7, 0), (0, 0))

for point in points:
    nav_aruco_map.navigate_wait(point[0], point[1], 2)
    rospy.sleep(1)
```

Более подробно об установке и использовании advancedClover можно почитать здесь.

Технические подробности

Итак, что у нас внутри:

- Theia прикольная web IDE которая очень похожа на VSCode и во многом даже лучше его.
- Расширения для vscode (работает и в theia), предоставляющее инструменты ROS.
- Питоновская библиотек, на самом деле это просто примеры из документации вынесенные в классы.
- Документация на docsify (gitbook слишком усложнён).
- Сборка на GitHub actions.

На самом деле интерес представляет только расширение, но я расскажу о всех частях.

IDE

Изначально мы использовали code-server, но он слишком долго собирался и был монолитен, поэтому мы перешли на theia которая менее монолитна и собирается намного быстрее, однако её пришлось допилить:

1. Theia по дефолту тянет с собой около 1гб `node_modules` что не очень хорошо, так что мы добавили сборку в один бинарник с помощью pkg, однако необходимо патчик код генерируемый при build перед упаковкой в бинарник с помощью патча.
2. Несмотря на то что по сравнению с code-server сборка занимала раза в три меньше времени, её пришлось ещё немного распараллелить вынеся скачивание плагинов/расширений в отдельный job на GitHub actions.
3. Благодаря стараниям тех людей кто пишет web-стандарты theia нормально работает либо на HTTPS, либо на localhost, по этому мы написали специальную программку на golang которая на лету генерирует сертификат, подписывает его корневым (который установил пользователь), и им по нему отправляет HTTPS.

Расширение

Вот здесь самое интересное, расширение написано на TypeScript, собирается с помощью webpack и использует react, roslibjs и OpenCV.

ROSLIB.js

Для коммуникации с ROS используется библиотека roslib.js, у неё есть некоторые особенности:

1. Так как мы используем строго (ну почти) типизированный TypeScript нам нужен типизированный ROS, типизация по дефолту лежит в DefinitelyTyped, но она там немного кривая, поэтому лучше создать файл roslib.d.ts который будет содержать типизацию для roslibjs и в случае если она не совпадает исправлять её.
2. Roslib использует колбэки а они очень неудобные, поэтому мы промисифицировали функции roslib. Здесь.
3. Потом была написана обёртка для roslib которая ещё немного упростила работу с сервисами и топиками.
4. И ещё мы написали хуки для реакта что бы ещё немного упростить работу.
5. ROS умеет кодировать сообщения в cbot или в json, json не умеет принимать 64-битные числа (они становятся неточными) и занимает больше места, а cbor из-за кривой реализации не может отправлять массивы сообщений но зато меньше по размеру и умеет в 64-bit.
6. Чтобы декодировать MAVLink нужно получать массив из 64 битных чисел, в js они называются BigInt, однако roslib когда видит 64 битное

число обрезает его, поэтому мы написали замену модулю roslib, замена применяется вебпаком.

Рендеринг картинок

Изначально мы использовали web video server для отображения картинок, однако он немного плохо работал и мы переписали рендеринг картинок на OpenCV, и тут кроется несколько подвохов:

1. Загрузка OpenCV происходит асинхронно повлиять на это никак нельзя, самым простым решением оказалось использование экспериментальной штуки в webpack под названием top-level async, мы написали простой модуль который загружает OpenCV, и когда мы хотим использовать OpenCV мы импортируем этот модуль и вызываем в top-level асинхронную функцию которая загружает OpenCV.
2. Так как картинки весят много и слать их закодированными в base64 не очень удобно, мы для получения картинок используем CBOR
3. В ROS картинки могут приходить в разных форматах по этому мы написали функцию которая конвертирует их в RGB. В общем можно посмотреть как это делается здесь.

И ещё чуть чуть веселья

1. Не всё что можно узнать от контроллера публикуется в mavros поэтому надо парсить MAVLink, для этого юзаем кодирование cbor и затем вытягиваем данные на основе данных отсюда.
2. TF2 в roslib не очень удобен, по этому чтобы получить позицию коптера нужно запрашивать фрэйм `map` и смотреть трансформации относительно его (не забывайте их инвертировать).

Библиотека для питона

Является в основном просто красиво оформленным кодом написанным в примерах кода из документации Клевера, однако есть интересный момент с получением картинки, он в отличие от стандартного возвращает последнюю полученную картинку предоставляя API похожий на `VideoCapture`, более подробно можно посмотреть здесь.

Сборка и Документация

Документация написана на markdown, рендерится с помощью docsify (ибо gitbook слишком переусложнён).

Проект собирается на GitHub actions, сборка максимально распаралелена.

Дрон для высаживания семян

CopterHack-2021, команда **MINIONS**.

Вы когда-нибудь задумывались, как будет выглядеть мир без деревьев? Закройте глаза и попробуйте представить себе безлюдную Землю. Деревья являются решающим фактором нашего существования не только потому, что они производят бумагу, пиломатериалы и жевательную резинку, но и потому, что они играют важную роль в углеродном цикле.

Со времени промышленной революции 1760–1840 годов мир пребывает в нескончаемом углеродном хаосе. Деревья и планктон - наши единственные спасители с точки зрения решения этой проблемы, и мы можем контролировать только одно из них - деревья.

Нам нужно спасать деревья, защищая их от разрушительной деятельности человека, такой как вырубка лесов, вырубка лесов в целях урбанизации и т. д. Деревья - легкие для земли. Это важная часть природной экосистемы. Они уравновешивают состав почвы, а также служат барьером для ветра и шторма. Таким образом, они обеспечивают различное использование экосистемы. По этим причинам крайне важно спасать деревья.

Поскольку существует множество опасных и труднодоступных участков для посадки людей, наиболее жизнеспособной альтернативой является использование дронов для посадки в этих регионах.

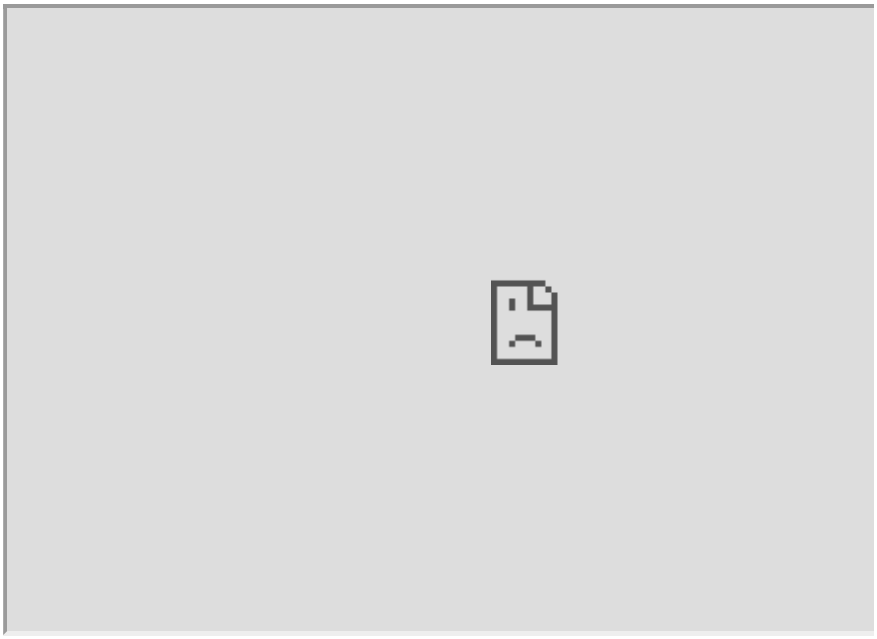
Как следует из названия, дроны, стреляющие семенами, будут стрелять семенами в плодородную почву, чтобы позволить миллионам деревьев вырасти снова после вырубки для промышленного использования. Если скорость посадки превышает скорость вырубки, в конечном итоге мы восстановим деревья, которые когда-то срубили.



Наша цель

Мы сделаем дроны способными удерживать семена на борту и сбрасывать их в область, которую мы проехали в специальном приложении. Мы можем контролировать плотность семян и высоту капли. Мы также думали о защите семян от насекомых, животных и обезвоживания. Мы выбираем технику шара земли, изобретенную Масанобу Фукуока, также известную как Техника Фукуока. Этот земной шар содержит все необходимые элементы для выращивания, семена растений и землю для защиты. Когда мы бросаем его на землю, земной шар будет удерживать семена, пока он не получит необходимое количество воды, и семена не начнут прорастать.

Видео с YouTube:



Нам удалось выполнить небольшие посевные миссии, но мы столкнулись с некоторыми проблемами, связанными с автономным полетом с GPS.

Мы покрыли нашу батарею, чтобы защитить ее от холода, посевные работы нужно начинать зимой, так как семена яблони должны оставаться в холодном месте в течение некоторого времени, чтобы акклиматизироваться.

- Капсулы с семенами.
- Как собрать высевающий механизм на дрон Clover 4.2.
- Как управлять механизмом раздачи.
- Программирование.

Файлы

Ссылка на все файлы, используемые в этом проекте: https://github.com/Sahinysf/TreeSeedQuad.

Капсулы с семенами

Техника Фукуока

На юге Японии японский фермер и философ Масанобу Фукуока изобрел технику посадки семенных шариков. Этот метод считается естественной техникой земледелия, не требующей машин, химикатов и очень небольшого количества прополки. При использовании семенных шариков земля обрабатывается без какой-либо подготовки почвы.



Преимущества семенных шариков

- Делать шарики с семенами проще и проще без использования машин.
- Легче для лесовосстановления и посадки на труднопроходимой местности
- Способствовать защите почвы, окружающей среды и средств к существованию
- Это органический метод, без использования каких-либо химикатов.
- Это недорогой метод по сравнению с традиционными методами облесения / лесовозобновления.
- Требует низких эксплуатационных расходов.

Какие семена можно использовать?

Любое семя, которое растет в вашем районе (у нас это семя яблока).

Размер и вес семенной капсулы:

Размер и вес семенных коробочек очень важны для этого проекта. После некоторых экспериментов мы решили, что лучший размер - диаметр 16-18 мм, а максимальный вес - 10 грамм.

Необходимые материалы для изготовления семенных шариков:

1. 1 ведро глины;
2. 1 ведро органической темной почвы / компоста;
3. 1 ведро воды (количество воды может варьироваться в зависимости от типа почвы);
4. ¼ ведро семян.

Шаги по изготовлению семенных шариков:

1. Соберите одинаковое количество глины и органической почвы. Например, если вы используете одно ведро глины, вам следует смешать его с одним ведром органической почвы.
2. Убедитесь, что глина и органические частицы почвы мелкие.
3. Текстура глины и органической почвы должна быть влажной, но не липкой.
4. Возьмите немного смеси и скатайте ее в шарики. Проверьте мяч, бросив его на плоскую поверхность. Если мяч не ломается легко, значит, у него хорошее сцепление.
5. Шарики с семенами должны быть идеально круглой формы, иначе они застрянут при броске квадрокоптером.
6. Вставьте семена (от 1 до 2 семян на семенной шарик для постоянных деревьев, таких как красное дерево, сандал, апельсин, моринга…) (± 5 семян на семенной шарик для овощей, цветов, трав, клевера…)
7. Сушите семенные шарики в течение одного-двух дней в затененном месте, если высохнуть должным образом, семенные шарики будут защищены от внешних хищников, таких как птицы, грызуны…



Вторая техника - бумажные капсулы с семенами.

На этот метод повлияла корейская газета, в которой были семена, которые можно было посадить на улице после прочтения. Бумажные капсулы с семенами:

Необходимые материалы:

1. любая бумага;
2. вода;
3. блендер;
4. семена.

Шаги по изготовлению шариков из бумаги:

1. Измельчение всей вашей бумаги,
2. Положите бумагу в блендер и добавьте воды, через 2 минуты перемешайте.
3. Выдавить всю воду бумагой,
4. Добавьте семена и придайте круглую форму.
5. Дайте высохнуть на ночь.

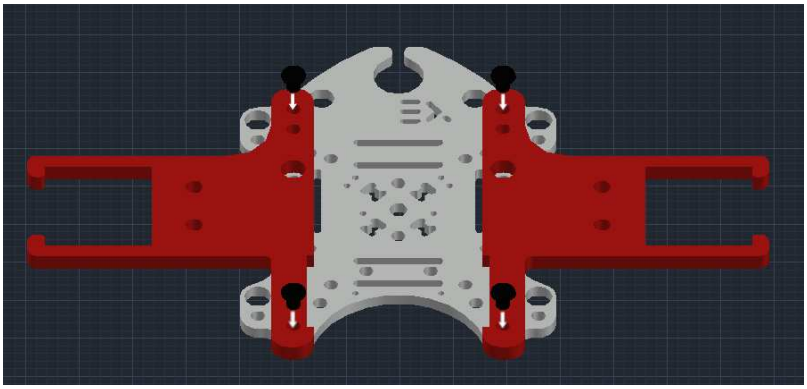


Преимущества бумажных шаров:

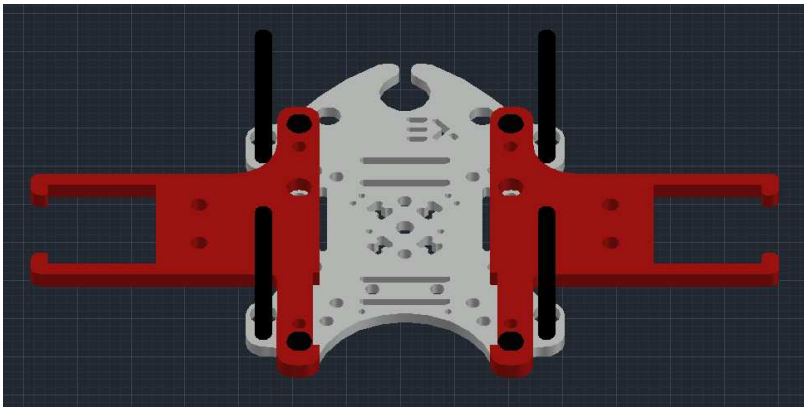
- легко найти материалы;
- экологически чистый.

Как собрать высевной механизм для Clover 4.2

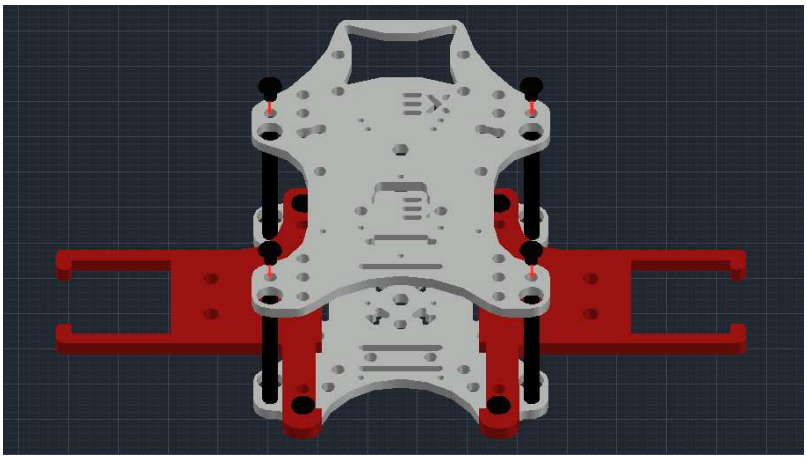
1. Установите нижние держатели бака на крепление верхней палубы и закрепите винтами M3x8.



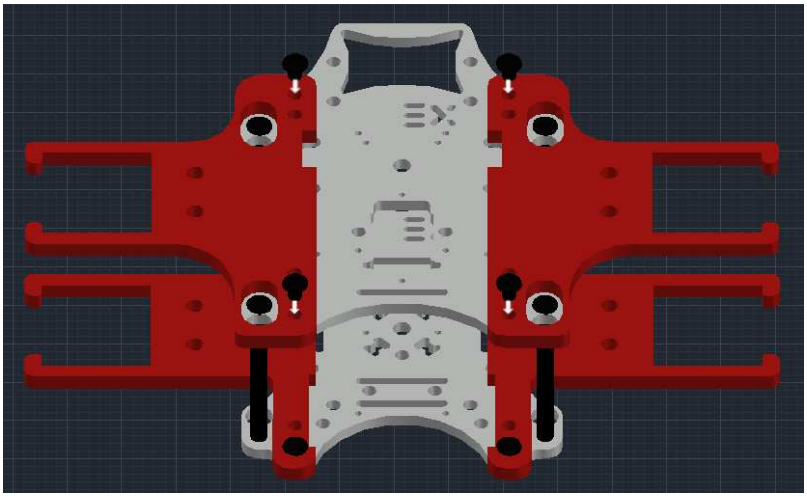
2. Установите нейлоновую стойку (40 мм) с 4 сторон крепления для деки.



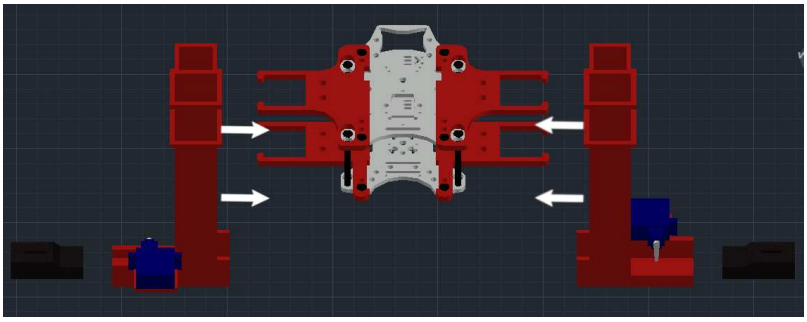
3. Установите поручень и закрепите винтами M3x8.



4. Установите верхние держатели бака на верхнее захватное крепление и закрепите винтами M3x8.

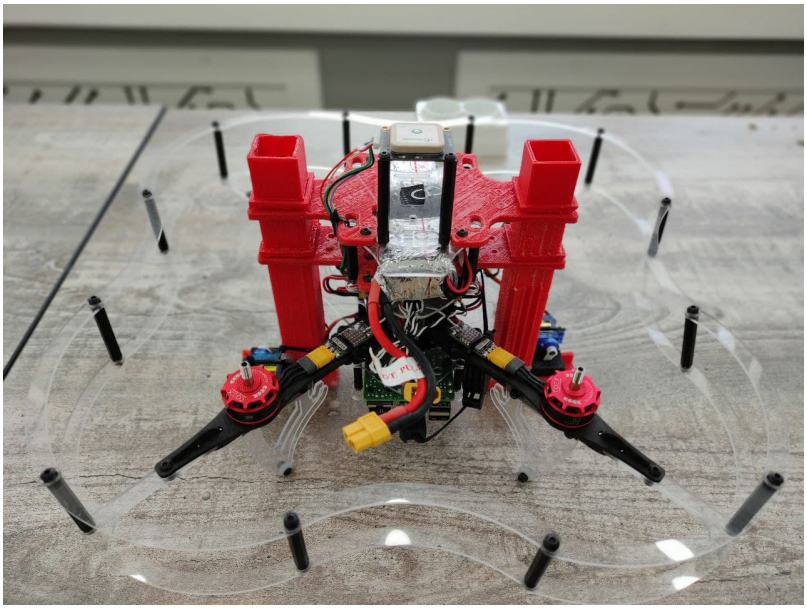


5. Осторожно подсоедините резервуары к держателям резервуаров.



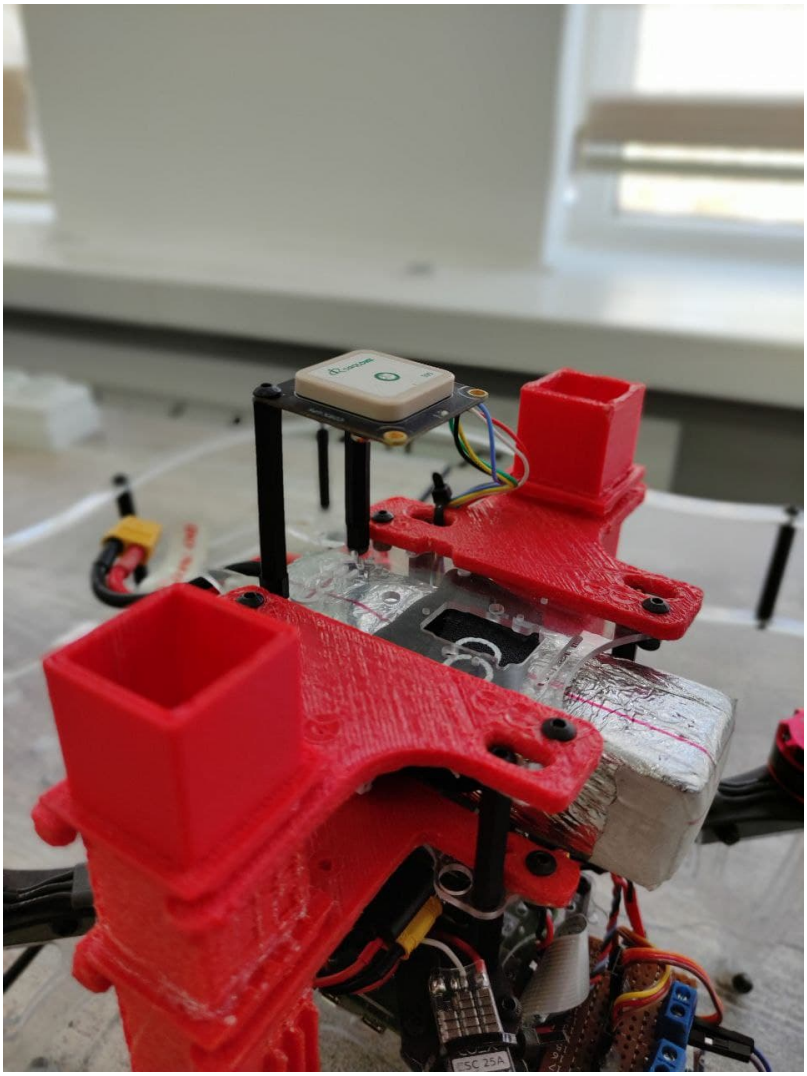
6. Подсоедините серводвигатели SG90 к резервуару с помощью стяжки.

Окончательный вид сеялки дрона:



Модуль GPS

Мы установили модуль GPS наверх, используя 2 нейлоновые стойки (40 мм).



Мы покрыли аккумулятор, чтобы защитить его от холода.

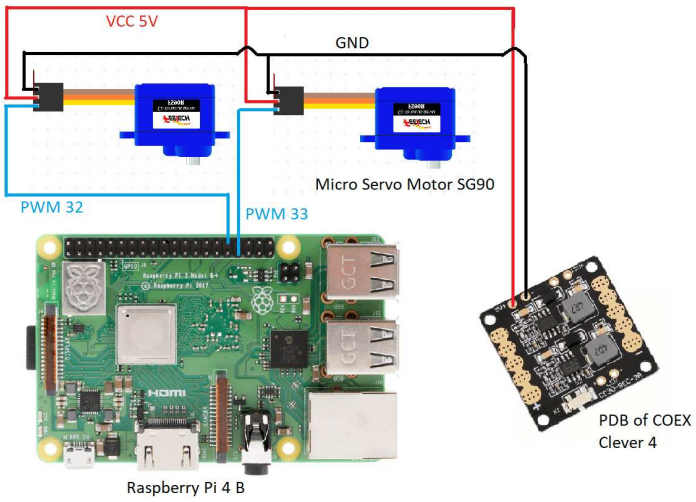


Как управлять механизмом высева

Электронная часть механизма высева семян состоит из:

- Raspberry Pi 4 B из COEX Clover 4.
- 2 микро-серводвигателя SG90.
- PDB (блок распределения питания) COEX Clover 4.

Сигнальные контакты серводвигателя подключены к контактам 32 и 33 аппаратной ШИМ Raspberry Pi, а питание снимается с платы распределения питания (5 В).



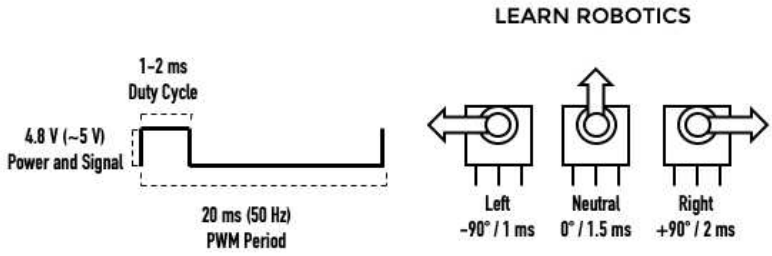
Пояснение кода для управления серводвигателями

Сервомоторы управляются с помощью сигнала ШИМ (широтно-импульсной модуляции) от Raspberry Pi. ШИМ контролирует количество времени, когда сигнал ВЫСОКИЙ или НИЗКИЙ в течение определенного периода времени. Рабочий цикл - процент времени, когда сигнал ВЫСОКИЙ.

В таблице ниже представлен рабочий цикл серводвигателя SG90 для каждого угла серводвигателя. Чтобы использовать рабочий цикл в коде, нам нужно преобразовать время в проценты, разделив время рабочего цикла на общий период ШИМ.

Что мы получаем:

- Угол поворота -90° или рабочий цикл 2 мс => 1/20 * 100% = рабочий цикл 5%.
- Угол поворота 90° или рабочий цикл 2 мс => 2/20 * 100% = рабочий цикл 10%.
- Угол поворота 0° или рабочий цикл 1,5 мс => 1,5 / 20 * 100% = 7,5% рабочий цикл.



Мы сделаем это с помощью библиотеки RPi.GPIO и написания кода Python на Raspberry Pi.

Сначала импортируйте библиотеку RPi.GPIO и функцию сна:

```
import RPi.GPIO as GPIO
from time import sleep
```

Затем установите режим GPIO как BOARD:

```
servo = 33
GPIO.setmode(GPIO.BOARD)
GPIO.setup(servo, GPIO.OUT)
```

Далее создайте переменную для сервопривода ШИМ. Затем отправьте сигнал ШИМ 50 Гц на этот вывод GPIO с помощью функции GPIO.PWM(). Начните сигнал с 0.

```
pwm=GPIO.PWM(servo, 50)
pwm.start(0)
```

Используйте функцию `ChangeDutyCycle()`, чтобы записать проценты рабочего цикла в серводвигатель.

```
pwm.ChangeDutyCycle(5) # left -90 deg position
sleep(1)
pwm.ChangeDutyCycle(7.5) # neutral position
sleep(1)
pwm.ChangeDutyCycle(10) # right +90 deg position
sleep(1)
```

Программирование

Чтобы миссия была достигнута наилучшим образом и в пределах нашей досягаемости, от нас требовалось использовать многопоточность в Python.

Простой код миссии.

```
import threading
import time
import rospy
from clover import srv
from std_srvs.srv import Trigger
import RPi.GPIO as GPIO

rospy.init_node('flight')

get_telemetry = rospy.ServiceProxy('get_telemetry', srv.GetTelemetry)
navigate = rospy.ServiceProxy('navigate', srv.Navigate)
navigate_global = rospy.ServiceProxy('navigate_global', srv.NavigateGlobal)
set_position = rospy.ServiceProxy('set_position', srv.SetPosition)
set_velocity = rospy.ServiceProxy('set_velocity', srv.SetVelocity)
set_attitude = rospy.ServiceProxy('set_attitude', srv.SetAttitude)
set_rates = rospy.ServiceProxy('set_rates', srv.SetRates)
land = rospy.ServiceProxy('land', Trigger)

servo1 = 33        # PWM pins
servo2 = 32

GPIO.setmode(GPIO.BOARD)    #set pin numbering system

GPIO.setup(servo1,GPIO.OUT)
GPIO.setup(servo2,GPIO.OUT)

pwm1 = GPIO.PWM(servo1,50)    #create PWM instance with frequency
pwm2 = GPIO.PWM(servo2,50)

pwm1.start(0)        #start PWM of required Duty Cycle
pwm2.start(0)


def servo_drop(seconds):   #function to drop seed capsules from 2 tanks
    print("Dropping")

    i = 1                          #variable to choose which tank
    for num in range(seconds/2):
        if(i == 1):                #first tank
            pwm1.ChangeDutyCycle(10) # release one seed capsule
            time.sleep(0.5)
            pwm1.ChangeDutyCycle(5) # push then drop the capsule
            time.sleep(0.5)
            i = 2                  #changing the variable for to use the secon

        elif(i == 2):              #first tank
            pwm2.ChangeDutyCycle(10) # release one seed capsule
            time.sleep(0.5)
            pwm2.ChangeDutyCycle(5) # push then drop the capsule
            time.sleep(0.5)
            i = 1                  #changing the variable for to use the first

        print(num)
        time.sleep(2)


if name == "main":
    # Take off and drone 10m above the ground
    navigate(x=0, y=0, z=10, frame_id='body', auto_arm=True)

    # rospy waits for 10 seconds to take off
    rospy.sleep(10)

    # Dropping starts simultaneuously with flying forwards 5 meters
    d = threading.Thread(target=servo_drop, args=(18,))  # 18 is the sum of al
```

```
    d.start()

    navigate(x=5, y=0, z=0, frame_id='body')

    #rospy waits for 8 seconds to fly foward
    rospy.sleep(8)

    # Fly right 1 m
    navigate(x=0, y=1, z=0, frame_id='body')

    #rospy waits for 2 seconds to fly right
    rospy.sleep(2)

    # Fly backward 5 m
    navigate(x=-5, y=0, z=0, frame_id='body')

    #rospy waits for 8 seconds to fly backward
    rospy.sleep(8)

    # Perform landing
    land()

pwm1.stop()
pwm2.stop()
GPIO.cleanup()
```

Литература

- Deforestation explained
- http://www.fao.org/fileadmin/templates/rap/files/NRE/Forestry_Group/Landslide_PolicyBrief.pdf
- Global Forest Change
- https://web.archive.org/web/20090115211020/http://www.rmaf.org.ph/Awardees/Biography/BiographyFukuokaMas.htm
- http://www.guerrillagarpting.org/ggseedbombs.html

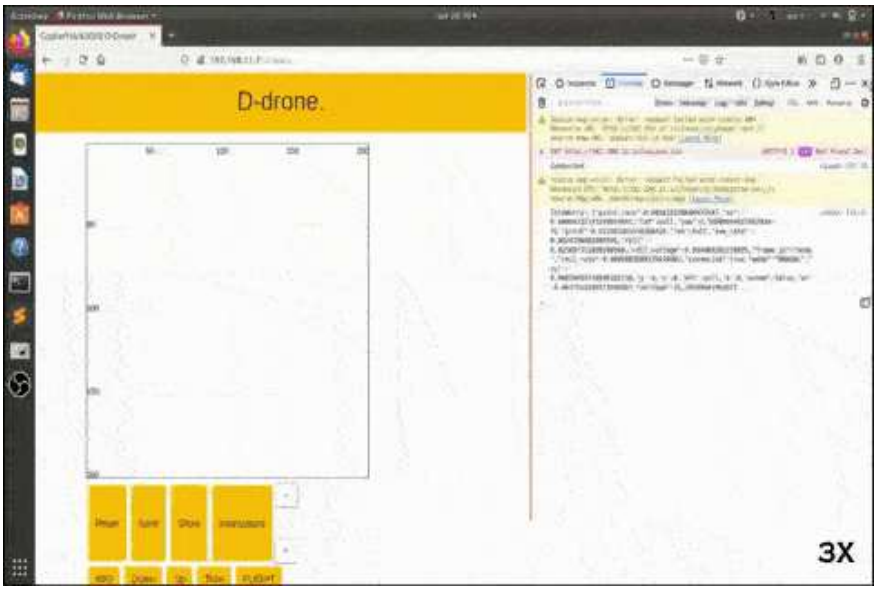
Разработано командой MINIONS

Особая благодарность Международному университету Ала-Тоо за финансирование наборов Clover 4.



Граффити коптер D-drone

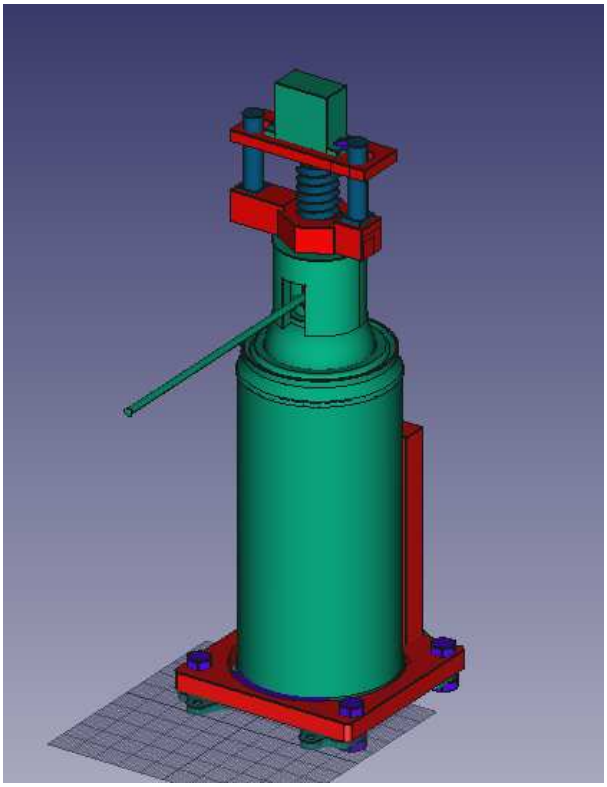
CopterHack-2021, команда **AT Makers**.



Введение

Люди стремятся научить искусственный интеллект всему, что могут делать сами. Рисовать нас учат с детства. И почему бы не научить дрона рисовать? На данный момент коптеры и граффити набирают свою популярность. Поэтому мы решили совместить их.

Модели и сборка





Для выполнения проекта вам нужно иметь в наличии:

- аэрозольная краска
- сервопривод MG90S;
- 3D-принтер;
- удлинитель распылителя;

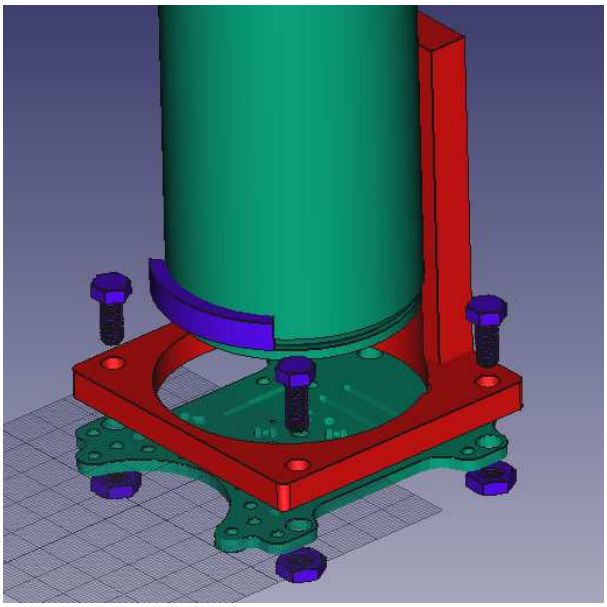
- лента с липучкой (желательно);
- 4 длинных винта M4 или;
- 2-4 коротких самореза M4 или M3.

Скачать и распечатать детали:

- держатель;
- винт;
- держатель стоек с гайкой;
- стойки (2 шт.);
- держатель для серво.

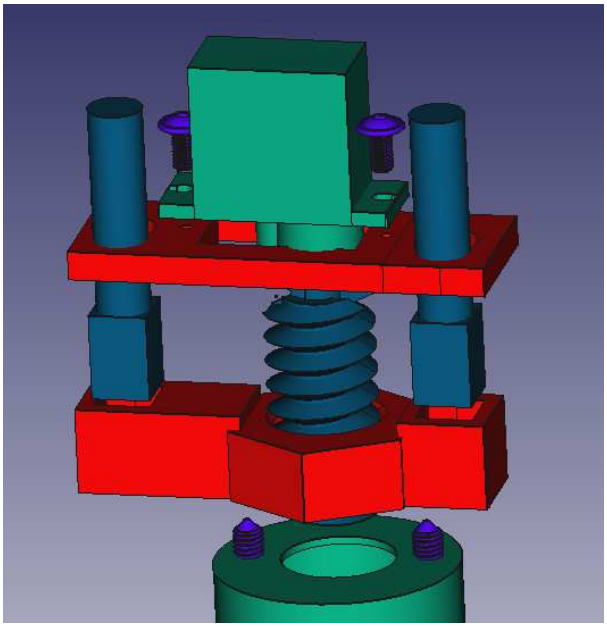
Держатель баллончика. Держатель баллончика прикрепляется к деке 4 винтами и гайками. Чтобы закрепить баллончик к держателю мы использовали ленту с липучкой. С помощью 4 гаек и винтов закрепляем деку с держателем сверху дрона.

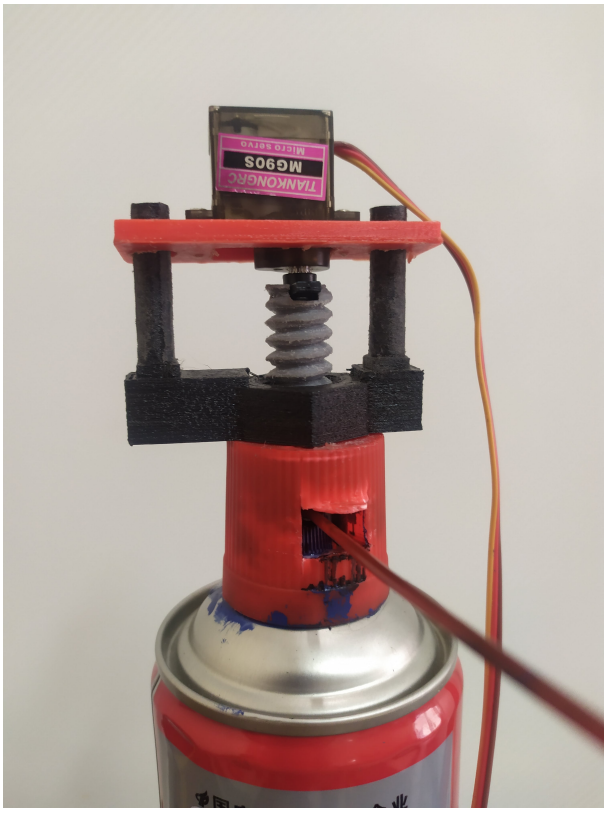
Вес держателя: 90 г.



Если диаметр баллончика меньше диаметра держателя, мы используем деталь в виде дуги, размером разницей между ними. Это помогает нам устойчиво закрепить баллончик.

Схема нажатия. Для нажатия клапана будем использовать винтовую передачу с неподвижной гайкой. К сервоприводу будут прикреплена планка с отверстиями, в которых будут входить стойки, закрепленные к гайке. Это помогает сервоприводу двигаться только по одной оси, вверх вниз. Также мы смоделировали крышку для кнопки баллончика, так как поверхность насадки не ровная.





Перед запуском

Настройка работы сервопривода

Перед запуском коптера нужно скачать servo.py и перенести его на RPi. Можно просто скопировать и вставить используя буфер обмена. Или скопировать используя команду scp. Например, так:

```
scp servo.py pi@192.168.11.1:/home/pi
```

Затем выполнить удаленно на Raspberry Pi следующие команды:

```
sudo pigpiod
python servo.py
```

Настройка веб-интерфейса

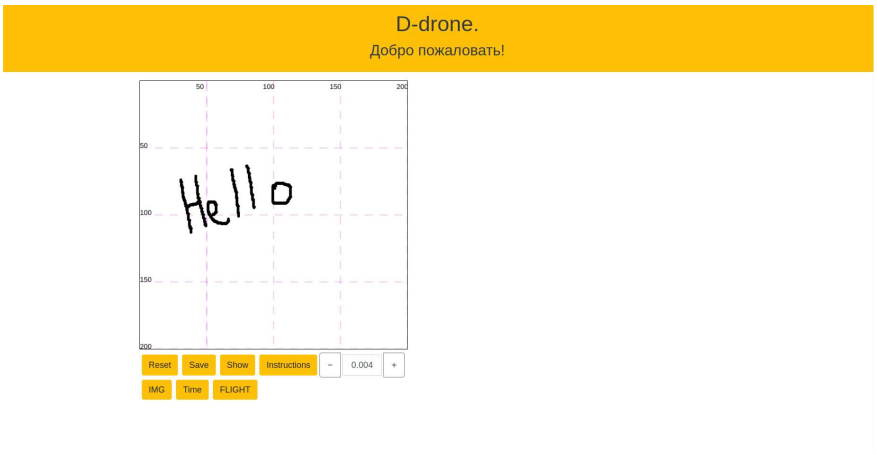
Нужно скачать репозиторий в формате .zip. Копируем на RPi и распаковываем с помощью следующих команд:

```
scp visual_ddrone-master.zip pi@192.168.11.1:/home/pi
cd catkin_ws/src/clover/clover/www
unzip /home/pi/visual_ddrone-master.zip .
mv visual_ddrone-master ddrone
```

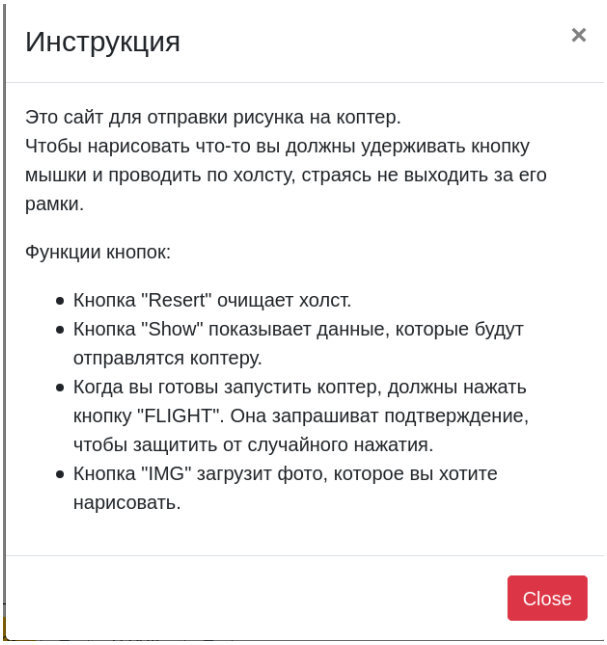
Теперь чтобы открыть веб-интерфейс нужно перейти по ссылке http://192.168.11.1/clover/ddrone.

Веб-интерфейс

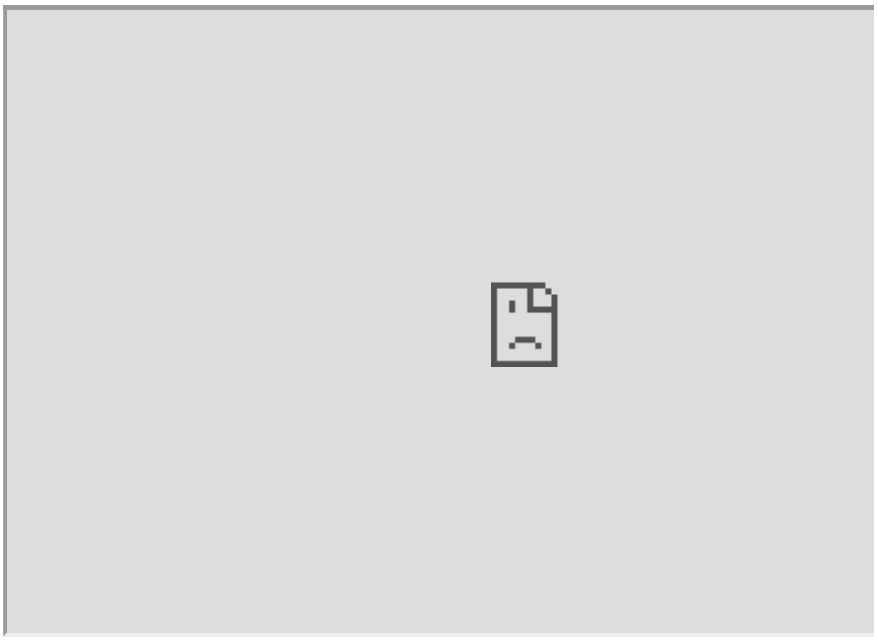
Запуск нашего дрона осуществляется с помощью веб-сайта. Веб-интерфейс позволяет рисовать и кодировать нарисованное в G-code. Данные координат будут переданы для дальнейшей обработки и исполнением коптером.



Мы выбрали веб-интерфейс для управления коптера, потому что он легче и ближе для пользователя.



Полеты



Благодарность

Хотим выразить благодарность Международному университету Ала-Тоо за предоставленную финансовую помощь в осуществлении данного проекта.

ALA-TOO INTERNATIONAL UNIVERSITY

Программируемый летающий автомобиль

Автор: **Колмаков Станислав Витальевич**.

CopterHack-2021, команда Зауральский викинг. Место работы: ГАНОУ КО «ЦРСК», г. Курган.

В мире широко разрабатываются и применяются новые модели мультироторных беспилотных летательных аппаратов (БПЛА). Самый распространенный и доступный способ знакомства с БПЛА – покупка квадрокоптера в магазине бытовой технике. В таком случае потребитель сможет лишь развить навыки пилотирования и производить аэрофотосъемку. Для получения большего количества навыков и возможностью дополнить квадрокоптер лучше заплатить больше. Речь идет о конструкторах для сборки пилотируемого летательного аппарата.

В ногу со временем идет компания Copter Express (COEX), разрабатывая и модернизируя программируемые квадрокоптеры Clover. Проект летающего автомобиля был разработан на основе программируемого квадрокоптера Clover 4 Code с возможностью двигаться по ровной горизонтальной поверхности, используя колесную базу. Данная возможность позволит расширить функционал использования программируемого квадрокоптера.

Задумка проекта началась с конструирования основы под электронные компоненты на гусеничной базе (рис. 1). В качестве основных компонентов выбрал полетный контроллер COEX Pix и микрокомпьютер Raspberry Pi 3 B+. Используя среду программирования Blockly, получилось привести робота в движение.

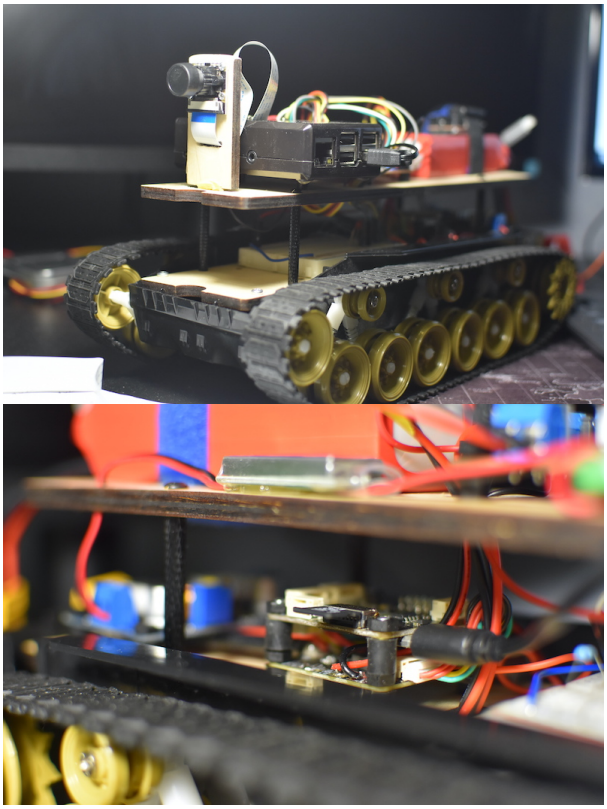


Рис. 1. Гусеничный робот с COEX Pix и Raspberry Pi 3 B+.

Следующим этапом разработки колесной базы для квадрокоптера стало использование конструктора четырехколесной платформы Pirate. Вес платформы с драйвером моторов составил 345 грамм (рис. 2). С учетом увеличения взлетной массы и для увеличения мощности, принято использовать пропеллеры 3-лопастные 6040 (рис. 3).

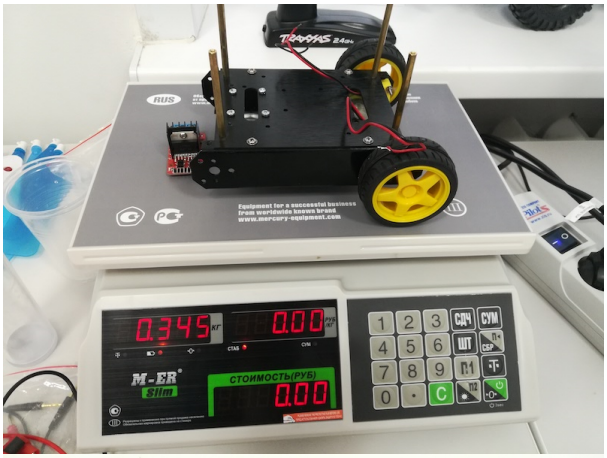




Рис. 2. Вес платформы. Рис. 3. Собранная конструкция.

Для облегчения веса двухколесной платформы решено смоделировать и распечатать на 3D-принтере основу для колесной платформы (рис. 4), используя PETG пластик. Данный пластик обладает более высокими прочностными характеристиками, чем распространенные ABS и PLA пластики.

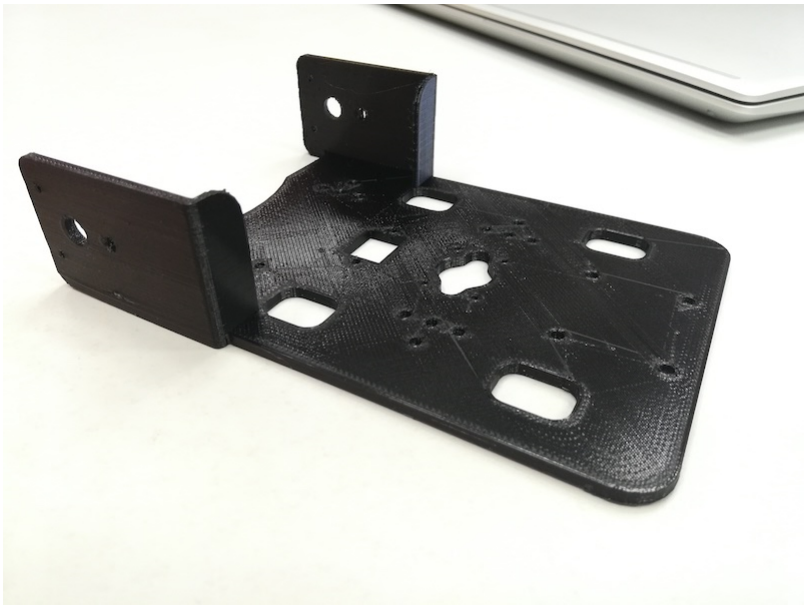
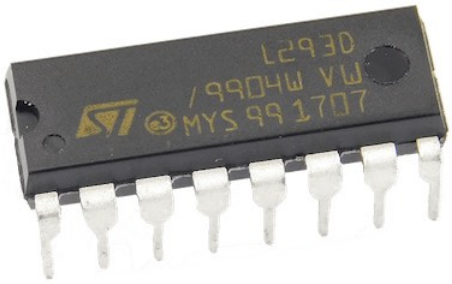


Рис. 4. Основа из пластика.

Ссылка на модель: https://drive.google.com/file/d/1_KPZfldSXNGiHbgnVBgMle-JvKtcDZHm/view?usp=sharing.

Платформа содержит: крепления под мотор-редукторы, отверстия для лазерного дальномера и Pi камеру. Также в основе предусмотрены монтажные отверстия для монтажа к квадрокоптеру и облегчения конструкции.

Следующим шагом для снижения веса, стало решение использовать микросхему драйвера двигателей L293D (рис. 5), вместо модуля драйвера двигателей L298N (рис. 6).



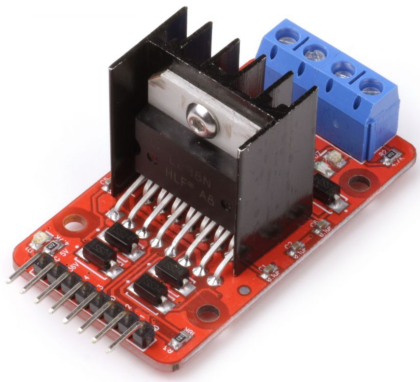


Рис. 5. Драйвер двигателей L293D. Рис. 6. Драйвер двигателей L298N.

С учетом всех облегчений, вес летающей машины составил 1 кг (рис. 7).



Рис. 7. Общий вид программируемого летающего автомобиля.

Испытания квадрокоптера, скрещенного с мобильной платформой, доказали работоспособность проекта с возможностью расширенного функционала программируемого квадрокоптера.

Следующим шагом развития проекта станет разработка алгоритмов распознавания линии, цветов и других образов.

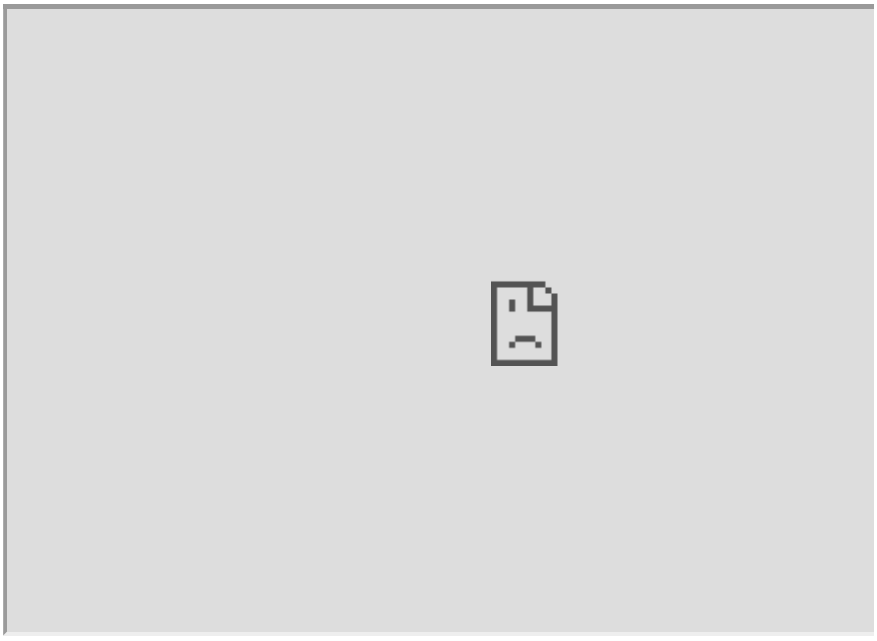
Дрон-Агроном

Проект команды **Quadrotor** с CopterHack-2021.

Суть проекта

В современном мире фермеры сталкиваются с проблемой не качественного или недостаточного орошения полей. Из-за этого по статистике ООН погибает около 25% урожая.

Эту проблему решают использованием авиации, что очень дорого и не практично. Наша команда разработала более дешевое решение проблемы - Дрон-агроном. Наш продукт разработанный на платформе Клевер 3 и базе рамы Tarot Iron Man 650 позволяет при небольшой взлётной массе (в сравнении с конкурентами) сохранить скорость, маневренность и время полета. Также использование насоса высокого давление позволяет обеспечить максимальную площадь распыления (что то похожее на квадрат 3x3 метра). В преимущество перед конкурентами можно также отнести не высокую стоимость и широкий комплект поставки.



Прототип

Первый прототип был полностью построен на базе Клевер 3 Pro (в форм факторе гексакоптера). Но после первых тестов (ссылка на один из них: https://drive.google.com/file/d/1gwpZZ2ZTkYvmTbspskpW79i8WShVEiI0/view?usp=sharing), было принято решение ставить всю электронику на мощную винтомоторную группу и легкую и прочную раму Tarot Iron Man 650.

Дополнительные материалы

Перейдя по этой ссылке (https://drive.google.com/drive/folders/1wCyUtMYOLiqYCBLC8aWpisBAH3ai9WNu?usp=sharing) вы можете получить доступ к нашему проекту. Там находятся примеры кода, 3D модели.

EasyToFly

Информация о команде

CopterHack-2021, название команды: **EasyToFly**.

В команде 5 человек:

- Игорь Сидорин @maerans12 (TeamLead)
- Артём Баталов @bart02 (Full-Stack разработчик)
- Карина Янышевская @fanot (Веб-разработчик)
- Никита Локтев @nikilokser (Специалист отдела Hard)
- Даниил Руфин (Специалист отдела Hard)

E-mail: a@batalov.me
Telegram: @bart02, @maerans

Введение

В работе рассматривается процесс разработки образовательного аппаратно-программного комплекса (далее – АПК, система) для безопасного обучения специалистов по профилю "Летающая робототехника".

Основной **проблемой данной сферы** является недостаточная безопасность открытых систем беспилотных летательных аппаратов (далее – БПЛА), а также сложность в их управлении и программировании для использования в обучении, промышленности и т.д.

Объектом проектной работы являются учебные комплексы для изучения процессов автоматизации. **Предметом** является АПК для безопасного обучения специалистов по профилю "Летающая робототехника".

Цель: Разработка АПК для простого и безопасного обучения специалистов по профилю "Летающая робототехника".

Задачи:

1. Проведение исследования среди потенциальных потребителей об их необходимостях при обучении специалистов.
2. Проведение анализа существующих решений.
3. Разработка технического задания (далее – ТЗ) на АПК.
4. Изучение литературы, необходимой для реализации проекта.
5. Формирование команды исполнителей на основании технического задания.
6. Разработка программной подсистемы АПК «Предотвращение столкновений».
7. Разработка программной подсистемы АПК «Мастер первоначальной настройки БПЛА».
8. Разработка программной подсистемы АПК «Монитор состояния БПЛА».

9. Разработка аппаратной составляющей АПК.



Выбор платформы

Платформа Clover 4 позволяет подключать дополнительные устройства и обеспечивать связь между ними и оборудованием на борту; а также тем, что программная архитектура данного набора основана на открытом полетном стеке PX4 и операционной системе для роботов ROS (далее – ROS), что предоставляет возможность ее использования в том числе и на других беспилотных летательных аппаратах, в которых используется такие же программные компоненты. После изучения рынка образовательных дронов в России, мы пришли к выводу, что данная платформа станет хорошим стартом разработки проекта.

Опрос потенциальных заказчиков. Разработка ТЗ

По результатам опроса потенциальных потребителей системы – педагогов детского технопарка «Кванториум» и других ОУ, было выяснено, что многие учреждения не имеют специально оборудованных зон для полетов, вследствие чего обучающиеся проводят настройку БПЛА и, в редких случаях, полет только под присмотром руководителя, на котором лежит ответственность за аппарат и окружающие предметы, жизнь и здоровье обучающихся.

Также нам стало известно о сложностях при настройке автономного полета, из-за частого использования консоли операционной системы Linux в процессе настройки, а также необходимости установки дополнительных программ.

Помимо этого, одним из запросов была компактная аппаратная составляющая системы, которая представляет собой защищенный корпус БПЛА, с возможностью быстрой установки используемого оборудования.

В качестве рассмотрения возможного места пилотного запуска системы, были проведены переговоры с Кванториумом г. Томска, об использовании АПК в обучающем процессе Аэроквантума.

Процесс разработки

Программная подсистема «Предотвращение столкновений»

Первым приоритетом стала программная подсистема «Предотвращение столкновений». Создав систему автономной защиты от “влетания” БПЛА в предметы, стоящие в учебной аудитории как во время ручного управления, так и тестировании программ автономного полета, а, при необходимости, и облета препятствий, мы сможем организовать более безопасное и продуктивное обучение.

Для разработки данной подсистемы, нами была изучена документация полетного стека PX4. Было выяснено, что полетный контроллер поддерживает эту функцию при передаче физически и геометрически обработанных данных с компьютера на борту. На борту Clover 4 установлен микрокомпьютер Raspberry Pi (далее – микрокомпьютер, компьютер на борту), мощности которого будут использоваться для этих целей.

Далее встал вопрос об использовании датчиков для разработки данной подсистемы.

LIDAR

Первое, что приходит в голову при разработке подобной системы – круговые LIDAR-датчики (далее – лидар). Самым популярным похожим решением на сегодняшний день является RPLIDAR A1.

Датчик основан на принципе лазерной триангуляции и использует высокоскоростное оборудование для получения и обработки изображений, система измеряет данные о расстоянии более 8000 раз в секунду. Лидар вращается по часовой стрелке для выполнения всенаправленного лазерного сканирования окружения на 360 градусов.

Результатом является облако точек, которое можно использовать при дальнейшей работе с пакетами ROS, позволяющими производить действия с этой структурой данных (такие как: перевод в другие структуры данных, построение карты, передача данных на другое оборудование).

Для установки датчика на Clover 4 было разработано и подготовлено для печати на 3D принтере специальное крепление, которое для экономии высоты было совмещено с креплением для аккумулятора.

Далее предстояла разработка программного обеспечения, которое выполняло все преобразования, как типов данных, так и геометрии (соглашения о системах координат полетного контроллера и ROS

отличаются). После этого обработанные данные направляются в полетный контроллер, который отвечает за принятие решения: остановить летящий БПЛА или продолжать полет.

Аппаратное решение из 4-х лазерных дальномеров

Помимо лидара, было принято решение использовать статические дальномеры для удешевления системы. Нами было разработано аппаратное решение на базе Arduino Nano и дальнедистанционных датчиков времени пролета VL53L1X. Изделие совместимо с разработанным нами креплением.

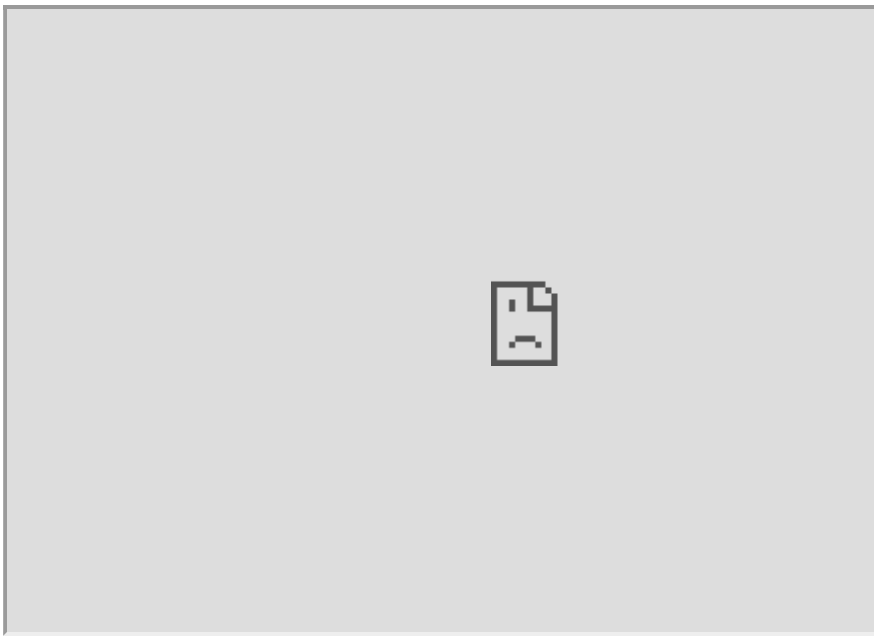
Далее было написано программное обеспечение для работы с оборудованием. Было необходимо передать данные с микроконтроллера, установленного на Arduino Nano на микрокомпьютер, представить показания с датчиков в том же виде, в котором они представлены при получении данных с лидара. И после этого произвести те же действия, что и с лидаром.

Программная подсистема «Мастер первоначальной настройки БПЛА»

Нами была запланирована разработка еще одной программной подсистемы, которая позволит настроить БПЛА, используя единый пользовательский веб-интерфейс, который «проведет» пользователя через весь процесс настройки. Также в данный веб-интерфейс должен быть встроен монитор текущего состояния БПЛА и редактор настроек.

В первую очередь нами был сделан каркас будущего интерфейса.

Также на данный момент реализован интерфейс калибровки датчиков БПЛА со встроенной анимацией для удобной работы с ним. Работу интерфейса калибровки можно увидеть на видео.



Аппаратная составляющая АПК

Также перед нами стояла задача разработать защищенный корпус БПЛА, с возможностью быстрой установки используемого оборудования, который не будет готов к полету и обучающимся не будет необходимости его собирать. На данный момент готова конструкторская документация изделия, а само изделие проходит апробацию.

Использование нашего продукта

Репозиторий проекта доступен по ссылке https://github.com/easy-to-fly/easy-to-fly.

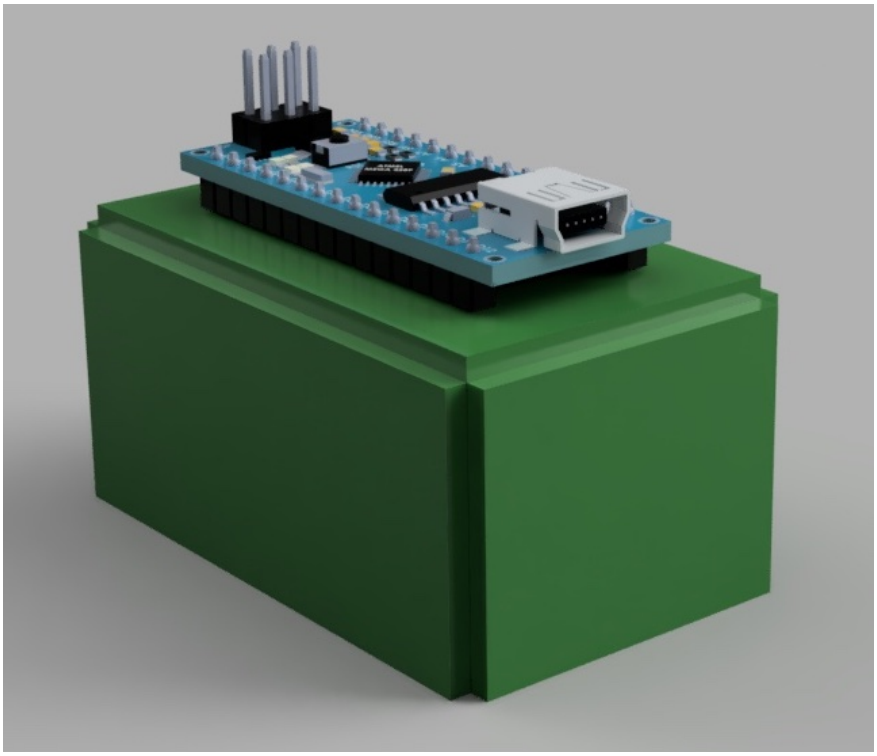
Сборка крепления для датчиков

Вам понадобится:

- **4x** стойки M3x26
- **1x** крепление для батареи
- **1x** платформа для крепления

Соедините всё как показано на модели.

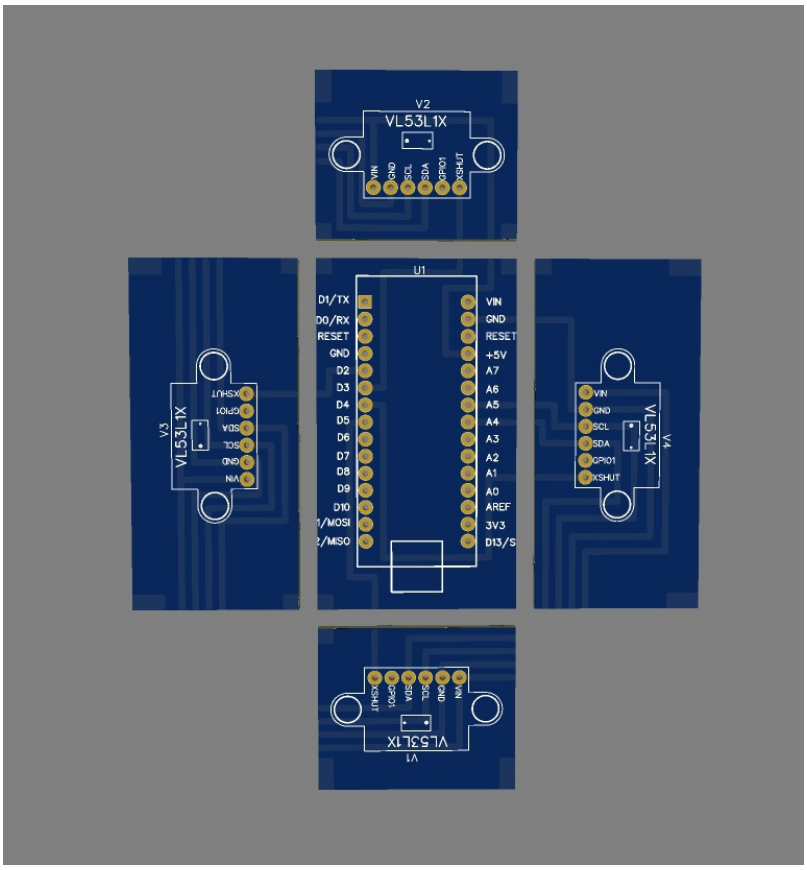
Сборка системы из 4-х лазерных дальномеров



Вам понадобится:

- **4x** дальнедистанционный датчик времени пролета VL53L1X
- **1x** Arduino Nano

- Разведите платы согласно развертке из файлов (вы можете использовать http://gerbv.geda-project.org/ для открытия)
- Припаять пины для установки Arduino Nano и датчиков на каждую из плат.
- Расположить платы как показано на развёртке.



- Составить из плат параллелепипед.
- Спаять все платы между собой по соответствующим контактным группам.
- Установить в пины датчики и Arduino nano.
- Убедиться в отсутствии короткого замыкания.

Установка ПО на Raspberry Pi

В первую очередь необходимо подключить ваш коптер к Интернету, один из способов это сделать - перевести его в режим клиента и подключить в Wi-Fi, имеющему доступ в Интернет.

Склонируйте репозиторий проекта в `catkin_ws` и перейдите в него:

```
cd ~/catkin_ws/src
git clone https://github.com/easy-to-fly/easy-to-fly
cd easy-to-fly
```

Установите необходимые зависимости

```
./install/ros_deps.sh
./install/arduino_deps.sh  # только если собираетесь использовать систему из 4
```

Запуск предотвращения столкновений

Для работы с лидаром необходимо изменить параметр `CP_DIST` PX4. Рекомендуется установить 0.5 м.

Теперь полетный контроллер будет получать сообщения от компьютера о расстоянии вокруг в специальном формате (читать доп. https://mavlink.io/en/messages/common.html#OBSTACLE_DISTANCE).

Теперь задача заключается в передаче сообщения такого типа на полетный контроллер.

В библиотеке MAVROS есть специальный плагин для этих целей: https://github.com/mavlink/mavros/tree/master/mavros_extras#obstacle_distance.

Таким образом, достаточно ее включить в `<rosparam param="plugin_whitelist">` в файле `mavros.launch`.

Далее, запускаем один из `.launch` файлов.

Заключение

В ходе выполнения проекта были проанализированы существующие разработки в области образовательной «Летающей робототехники», описаны их достоинства и недостатки. Выбрана платформа для старта проекта.

Реализована и протестирована основная часть проекта – программная подсистема «Предотвращение столкновений». Веб-интерфейс, согласно плану, находится в разработке до конца марта 2021 года.

Для реализации аппаратной составляющей был выполнен сборочный чертеж в системе автоматизированного проектирования Fusion 360. Полученные чертежи позволят производить БПЛА для реализации проекта. В будущем на основе конструкторской документации будут изготовлены дополнительные БПЛА.

В следующие два года планируется совершенствование проекта и выход на рынок. Некоторые из основных задач на 2021-2023 год: более подробное исследование рынка для выделения основных задач дальнейшей разработки системы, апробация системы в целом в Кванториуме, разработка системы навигации БПЛА по карте с использованием оборудования, которое будет установлено на аппаратной составляющей.

Хардатон Квиддич

CopterHack-2021, команда **Хардатон**.

Название проекта: Инженерное командное соревнование - хардатон «Квиддич на квадрокоптерах».



Команда проекта



- Бокта Оксана Александровна - лидер проекта, руководитель Центра по работе с одарёнными детьми МАОУ «Лицей № 176» г.Новосибирска.
- Шунаев Никита Александрович – проектный менеджер, куратор IT-направления.

- Алеков Иван Анатольевич - эксперт по программированию, преподаватель спецкурсов МАОУ «Лицей № 176».
- Жданов Олег Игоревич - эксперт по 3D-моделированию, преподаватель спецкурсов МАОУ «Лицей № 176».

Цель проекта

Развитие инженерных hard-компетенций школьников 7-11 классов по направлению «Беспилотные авиационные системы» с использованием платформы «COEX Клевер 3, 4».

Задачи проекта

1. Привлечение образовательных организаций, наставников, школьников, партнёров в направление «Беспилотные авиационные системы», к занятиям инженерным творчеством с использованием платформы «COEX Клевер 3,4».
2. Развитие hard-компетенций школьников (программирование, моделирование, пилотирование, в том числе автономно).
3. Развитие soft-компетенций (умение распределять роли, работать в команде, нестандартно мыслить, презентовать проект и др.).
4. Подготовка к инженерным соревнованиям, конкурсам и научно-практическим конференциям инженерно-технологической направленности (Олимпиада НТИ, WorldSkills и др.).
5. Создание условий для раннего выявления и сопровождения талантов НТИ.
6. Формирование среды, способствующей вовлечению в сообщества технологических энтузиастов, зарождению и развитию кружков НТИ по направлению «Беспилотные авиационные системы».

Актуальность проекта

В России с зарождением Национальной технологической инициативы растёт число школьников, наставников, энтузиастов, вовлечённых в техническое творчество. Современным школьникам доступны соревнования, чемпионаты, конкурсы инженерно-технологической направленности. Набирают обороты по количеству участников Олимпиада НТИ, чемпионат WorldSkills и другие. Профиль ОНТИ «Летающая робототехника», компетенция WorldSkills «Эксплуатация беспилотных авиационных систем» основаны на умениях осуществлять сборку, дефектовку, программирование, управление квадрокоптером с использованием платформы «COEX Клевер 3,4». Но сегодня очень мало соревнований, подготавливающих школьников к участию в таких сложных инженерных олимпиадах и конкурсах, вовлекающих в техническое творчество с использованием программируемых квадрокоптеров. Мы предлагаем концепцию инженерного командного соревнования - хардатон «Квиддич на квадрокоптерах», в котором школьник

с разным уровнем подготовки (начинающий, продолжающий, продвинутый), работая в команде, сможет проявить инженерное мышление, умение программировать, моделировать и управлять квадрокоптером. Вовлечению будет способствовать игровая модель, основанная на известном фильме «Гарри Поттер» и разные форматы проведения (очный и дистанционный).

Концепция проекта

Хардатон «Квиддич на квадрокоптерах» представляет собой инженерное командное соревнование под руководством наставника (3 человека в команде, роли распределяются по компетенциям - моделист, программист, пилот), в котором необходимо максимально проявить свои умения в испытаниях (как личностных, так и командных).

Соревнование проходит с использованием следующего оборудования.

Для команды: конструктор программируемого квадрокоптера «COEX Клевер 3, 4» (либо другой с возможностью установления захвата и камеры), ресурсный набор для ремонта (по необходимости), оборудование для изготовления захвата (можно использовать оборудование площадки проведения), инструменты, запасные аккумуляторы не менее 3 штук с возможностью подзарядки, ноутбук, флешка с файлами.

Для площадки проведения: оборудование для изготовления захвата - материал и обработку команда выбирает самостоятельно (3D-принтер, станок с ЧПУ, ручная обработка, композитные материалы и др.). Площадка проведения финала - спортивный зал (другое помещение), распределённый на зоны (входная зона, зона моделирования, зона программирования, полётная зона, коворкинг-зона с панелью для защиты презентаций и просмотра фильмов о Гарри Поттере, защитная сетка, разметка, 6 колец (размер колец 70-90), 10 подставок для мячей, 2 маленьких квадрокоптера-сничи, открытые лаборатории, ремонтная зона (паяльники, провода, запасные части), метки, Raspberry Pi, FPV, камера, 10 теннисных мячей, ноутбуки, удлинители, интернет.

Пример оформления и зонирования площадки:


Квиддич
на
квадрокоптерах
инженерное командное
соревнование
Хардатон









Дизайн помещений, названия и содержание испытаний схожи со сценами из знаменитого фильма «Гарри Поттер». Задания по компетенциям имеют дифференциацию по уровню подготовки участников (начальный, продолжающий, продвинутый). Ключевым испытанием является квиддич на квадрокоптерах. Победу одерживает команда, набравшая большее количество очков по всем испытаниям. К судейству могут привлекаться эксперты в компетенциях, не заинтересованные наставники команд, представители компаний-партнёров из реального сектора экономики, представители инженерных ВУЗов и др. Все участники, не прошедшие в финал хардатона, получают сертификаты участников; участники финала, не

занявшие места, получают диплом участника финала; команды, занявшие места в квиддиче, получают дипломы победителей или призёров и ценные подарки.

Этапы соревнования

1. Отборочный этап (командам необходимо разработать идею, изготовить захват и защиту на квадрокоптер).
2. Основной личный этап хардатона для команд начинающего, продолжающего и продвинутого уровня по моделированию, программированию и пилотированию.
3. Основной командный этап (квиддич на квадрокоптерах).
4. Заключительный этап (подведение итогов, награждение победителей).

Задания хардатона

Разработаны разноуровневые задания для очного и дистанционного формата проведения мероприятия.

- Ссылка на задания отборочного этапа.
- Ссылка на задания по программированию.
- Ссылка на задания по моделированию.
- Ссылка на задания квиддича на квадрокоптерах.
- Для оценивания выполнения заданий разработаны критерии.

Апробация проекта

Инженерное командное соревнование по беспилотным авиационным системам «Квиддич на квадрокоптерах» было апробировано на площадке МАОУ «Лицей № 176» города Новосибирска:

1. Дистанционный отборочный этап со 2 ноября по 20 ноября 2020 года.
2. Очный подготовительный этап - организаторами мероприятия проведено установочное совещание с наставниками команд, экспертами организована подготовка команд с предоставлением оборудования лабораторий электроники и прототипирования, новых производственных технологий со станками с ЧПУ, лаборатории DronLab.
3. Финал прошел с 4 по 6 декабря 2020 года.

В очном мероприятии приняли участие 27 команд из города Новосибирска, 81 участник из образовательных организаций и организаций дополнительного образования детей города Новосибирска, 14 наставников, 5 волонтёров, 3 эксперта, 5 помощников эксперта, 3 организатора. Общий охват мероприятия – 111 человек.





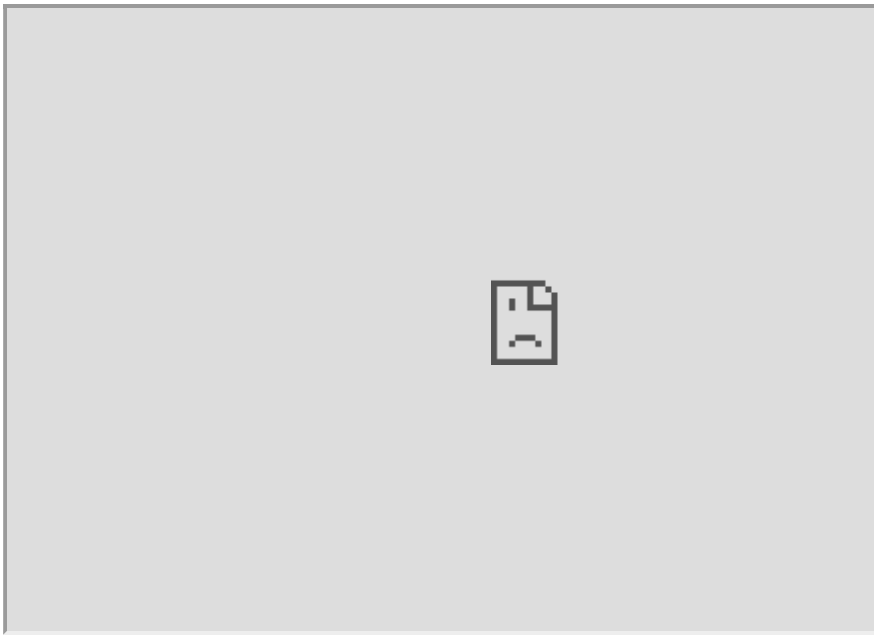






Предложенные участниками хардатона инженерные решения по разработке захвата и защиты на квадрокоптер:

Видео о команде и с полётной зоны.



Освещение мероприятия в СМИ

- https://nsknews.info/materials/kviddich-na-kvadrokopterakh-ustroili-v-novosibirskoy-shkole/ (городской новостной сайт).
- http://www.nios.ru/news/24302 (сайт департамента мэрии города Новосибирска).
- http://xn--176-qddohl3g.xn--p1ai/index.php/dostizheniya-litseistov/1627-kviddich-na-kvadrokopterakh (сайт МАОУ “Лицей №176”)
- https://www.facebook.com/633602557253181/posts/739165210030248/.

Перспективы развития проекта

- Проведение мероприятий продолжающего и продвинутого уровня с целью подготовки школьников к олимпиадам, чемпионатам и инженерным конкурсам по направлению «Беспилотные авиационные системы» с использованием платформы «COEX Клевер 3,4» как в очном, так и в дистанционном формате.
- Привлечение партнёров, экспертного сообщества к доработке заданий, организации и совместному проведению инженерного командного соревнования «Квиддич на квадрокоптерах» на Всероссийском уровне.
- Трансляция опыта проведения мероприятия на различных уровнях с целью вовлечения школьников в занятия инженерным творчеством, технологические кружки, развитие hard и soft-компетенций.

Октокоптер со специфичным расположением пропеллеров

CopterHack-2021, команда **PaD30DЖ**.



Введение

Наверное каждый, когда видит впервые летающий дрон, задается рядом вопросов типа:"А сколько он может летать так?","А как далеко он может улететь?","А сколько он может поднять?". Наша команда, проявила интерес к последнему из них, а именно к вопросу грузоподъёмности квадрокоптера компании COEX - Clover 4. Имея на руках сопутствующее оборудование и тягу к полетам, мы экспериментально выяснили, что полезная нагрузка данной модели коптера является масса от 800 до 1000 грамм. Но почему так мало и что с этим делать? Ведь в недалеком будующем уже должна развиться повсеместная доставка товаров на дронах и данное направление не должно ограничиваться 1 килограммом полезной нагрузки. Пребывая в поисках решений этой проблемы, мы решили реализовать самое элементарное что может прийти в голову - просто присоединить к одному дрону второй! Изучая список рекомендованных организаторами тем на конкурс CopterHack2021, мы нашли такую тему как "Два дрона в твердой связке", что являлось 100-процентным попаданием в интересующую нас область летательной робототехники.



Разработка

В результате поиска решения, удовлетворяющего всем нашим требованиям, мы остановились на нескольких вариантах. Было решено для начала сделать октокоптер со специфичным расположением пропеллеров, собственно как если бы два дрона соединили вместе на стандартном расстоянии двух парных пропеллеров квадрокоптера. Промежуточную часть было решено делать из стандартных соединительных деталей дрона (центральная дека, 4 луча, 2 крестовины), которые идут в базовой комплектации Clover 4.



В качетве полетного контролера был выбран Pixhawk v.4, т.к. он, в отличии от стандартных на Клеверах Pixracer и CoexPix имеет больше сигнальных портов для движков.



В качестве радиоприемника и радиопередатчика использовались стандартное оборудование FlySky.

Чтобы не отходить от темы "двух дронов в твердой связке", было принято решение использовать для одного октокптера две стандартные платы распределения питания. Вопрос питания временно остается открытым. Сейчас к двум отдельным PDB подсоединяются два отдельных аккумулятора, провода к контроллеру идут только от одной из плат распределения питания. Можно было бы использовать специализированную PDB и один обычный акквмулятор на 5200 mA/h, но тогда бы наш проект потерял свою "изюминку" и основную идею для развития.

Прошивка

https://drive.google.com/drive/folders/1Zc878JuDw_4FKSvvzibzGLVOnaxFhiqL?usp=sharing - ссылка на прошивку октокоптера с расширением PX4.

https://github.com/matveylapin/PX4-Autopilot - ссылка на прошивку октокоптера со всеми исходниками.

Основное отличие нашей прошивки, от прошивки Clover 4, не считая того, что первый окто-, а второй квадро-коптер, является наличие нескольких новых особых файлов, они и будут описаны ниже.

Первый файл - это файл геометрии.В нем прописываются координаты моторов в локальной системе координат дрона. Это требуется для составления обратной кинематической модели.

```
# Firmware/src/lib/mixer/geometries/MultirotorMixer/octa_h.toml

[info]
key = "8h"
description = "the coolest geometry on the planet"

[rotor_default]
axis      = [0.0, 0.0, -1.0]
Ct        = 1.0
Cm        = 0.05

[[rotors]]
name      = "front_right"
position  = [0.5, 1, 0.0]
direction = "CCW"

[[rotors]]
name      = "rear_left"
position  = [-0.5, -1, 0.0]
direction = "CCW"

[[rotors]]
name      = "mid_front_right"
position  = [0.5, 0.5, 0.0]
direction = "CW"

[[rotors]]
name      = "rear_right"
position  = [-0.5, 1, 0.0]
direction = "CW"

[[rotors]]
name      = "front_left"
position  = [0.5, -1, 0.0]
direction = "CW"

[[rotors]]
name      = "mid_rear_left"
position  = [-0.5, -0.5, 0.0]
direction = "CW"

[[rotors]]
name      = "mid_front_left"
position  = [0.5, -0.5, 0.0]
direction = "CCW"

[[rotors]]
name      = "mid_rear_right"
position  = [-0.5, 0.5, 0.0]
direction = "CCW"
```

Вторым файлом является миксер. В нем можно задать ограничения как всех моторов сразу, так и для отдельных групп.

```
# Firmware/ROMFS/px4fmu_common/mixers/octo_h.main.mix
# Octo H

R: 8h 10000 10000 10000 0
```

Третий и последний файл является основным файлом эирфпейма. В нем записываются дефолтные конфиги самой прошивки PX4. Также прописывается сам миксер и выход

```
#!/bin/sh
# Firware/ROMFS/px4fmu_common/init.d/airframes/8111_octo_h

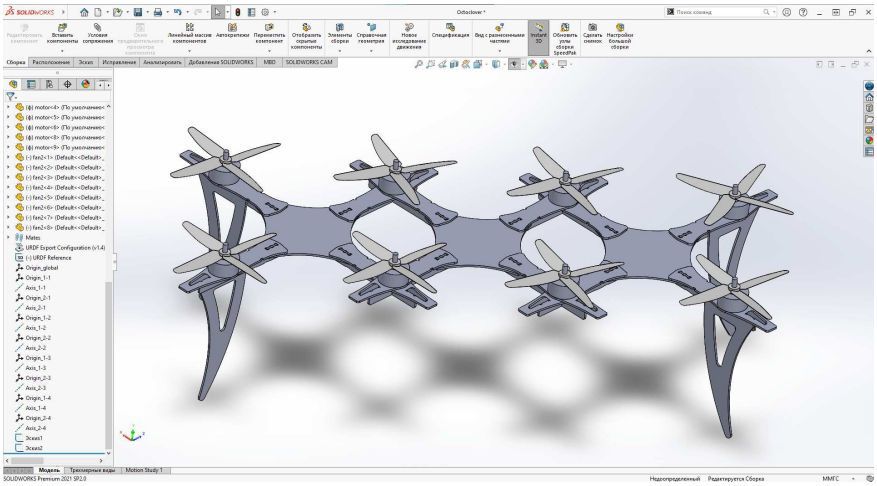
sh /ect/init.d/rc.mc_defaults

set MIXER octo_h

set PWM_OUT 12345678
```

3D-модель

Изначально, при принятии решения взяться за проект, нашим программистом был предложен вариант протестить все в симуляторе Gazebo. Ввиду недостатка времени над работой связанной с проектом, реализация предложения оттянута на неопределенный срок. Но, не смотря на этот факт, для более лучшей визуализации проекта, была сделана 3D-модель октокоптера. Все детали для сборки можно скачать по ссылке - https://drive.google.com/drive/folders/1TuxiSWmiPWIslp03GDKJL4yQ729Cn70I?usp=sharing.



Ожидание

Изначально у команды имелось намерение реализовать все в очень короткие сроки, сделав акцент на презентации и дальнейшем развитии.

Реальность

Ввиду многих отвлекающих факторов, таких как: учеба, работа, проживание в условиях пандемии, невозможность собраться всем сокомандникам вместе на одной площадке, выполнение проекта долго оттягивалось и конкретика по этому вопросу каждый раз переносилась на следующую неделю. За

достаточно малый промежуток времени, а именно за три недели до финала конкурса, мы взялись за дело. Вплоть до дня проведения финала CopterHack2021 команда будет работать над рабочим прототипом, чтобы 20-21 марта 2021 года показать реализованный проект. Данная статья также подлежит редактированию, по мере нашего прогресса.

С наилучшими пожеланиями и воодушевленным настроем, команда PaD30DЖ!

Теория

Урок №1 «Знакомство. Принципы проектирования и строение мультикоптеров»

Урок №2 «Основы электричества»

Урок №3 «Теория пайки»

Урок №4 «Аэродинамика полета. Пропеллер»

Урок №5 «Бесколлекторные двигатели и регуляторы их хода»

Урок №6 «Основы электромагнетизма. Типы двигателей»

Урок №7 «Принцип работы, типы и устройство аккумуляторов»

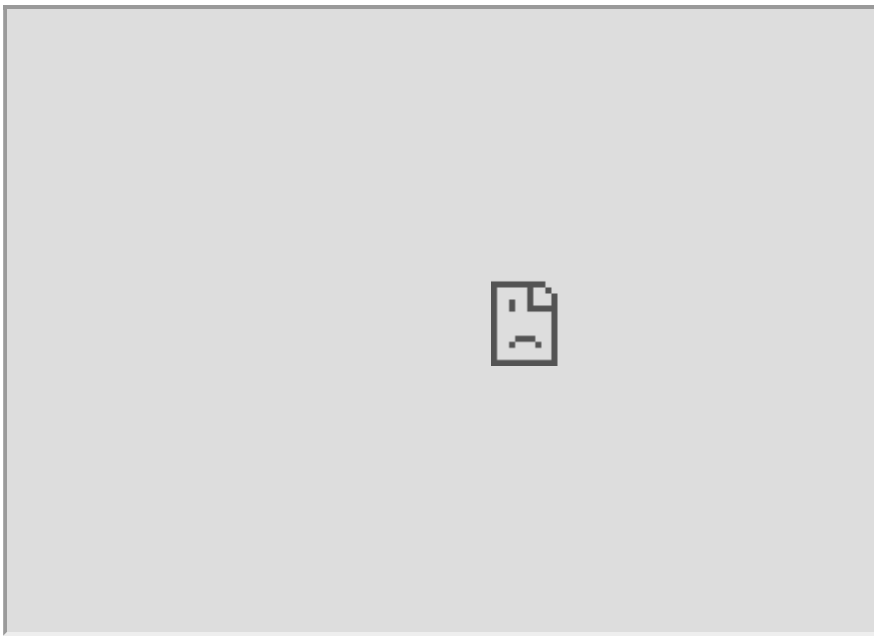
Урок №8 «Управление полётом мультикоптера. Принцип функционирования полётного контроллера. ПИД регуляторы»

Урок №9 «Основы радиосвязи. Принцип работы радиоаппаратуры управления»

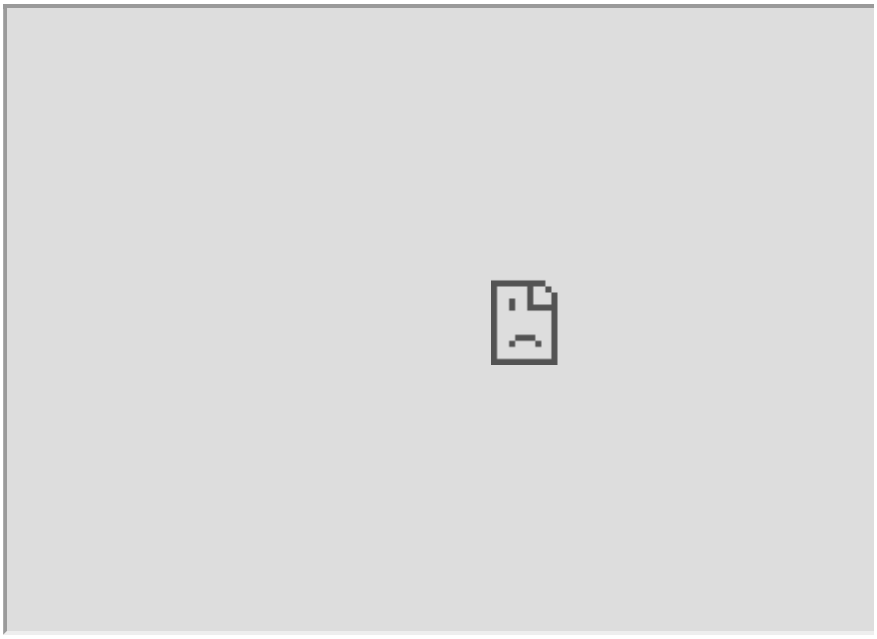
Урок №10 «Аналоговая и цифровая видеотрансляция. Применяемые камеры, радиопередатчики и приёмники»

Видеоуроки

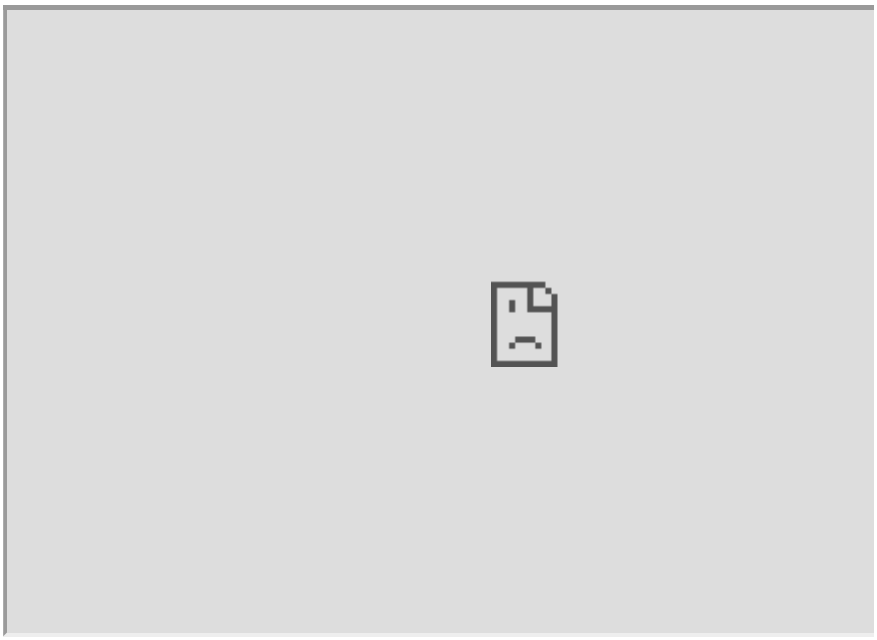
Немного о видах коптеров



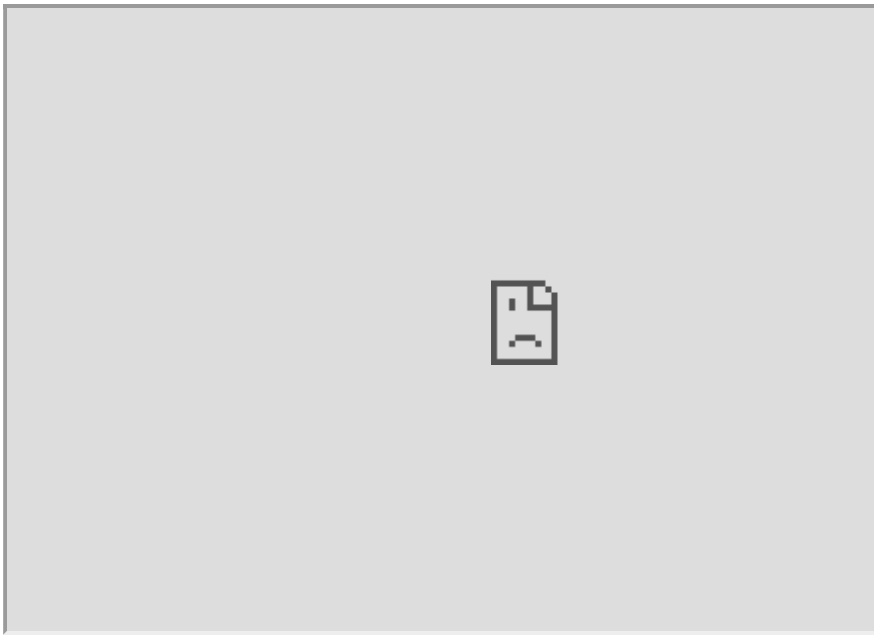
Часть 1



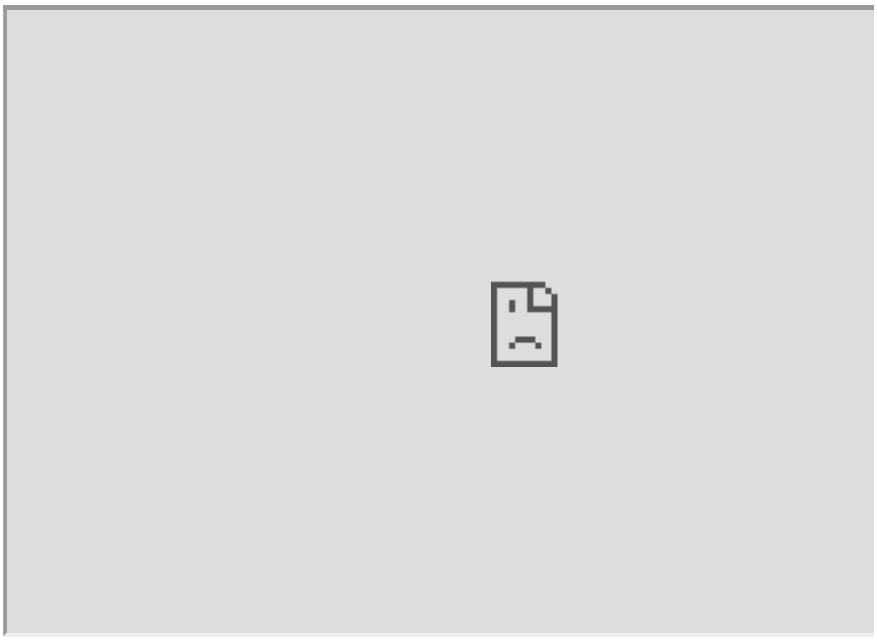
Часть 2



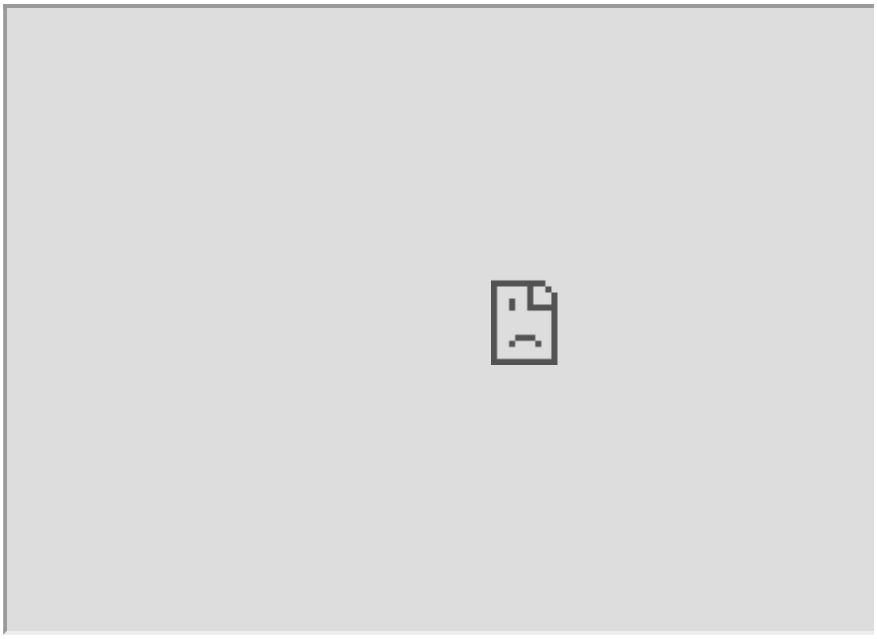
Часть 3



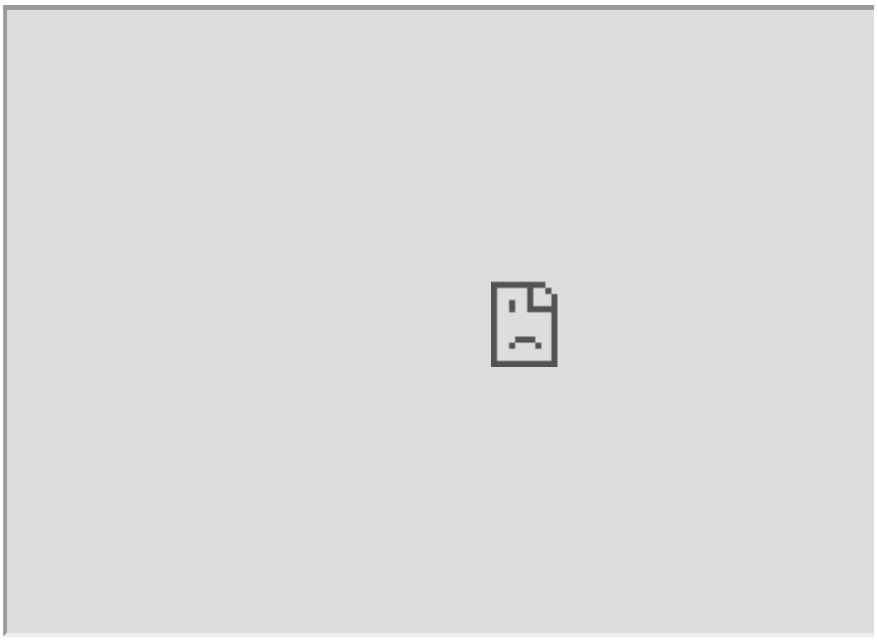
Часть 4



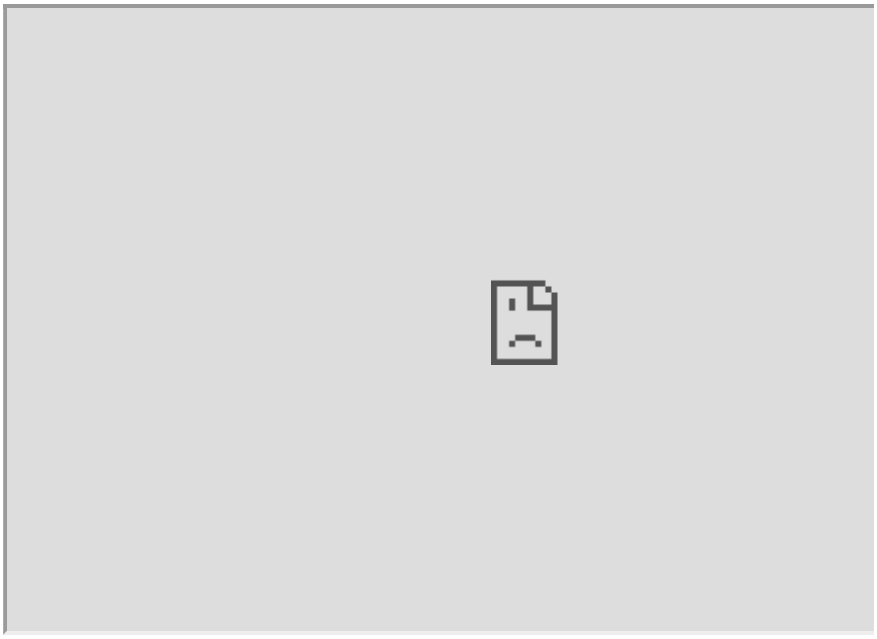
Часть 5



Часть 6



Автономные полеты



Методические материалы

Здесь представлены методические материалы для ознакомления с основами БПЛА, которые помогут грамотно организовать процесс как самостоятельного изучения, так и в группах.

Учебно-методическое пособие

Урок 1. «Знакомство. Принципы проектирования и строение мультикоптеров»

Урок 2. «Основы электричества»

Урок 3. «Теория пайки»

Урок 4. «Аэродинамика полета. Пропеллер»

Урок 5. «Основы электромагнетизма. Типы двигателей»

Урок 6. «Бесколлекторные двигатели и регуляторы их хода»

Урок 7. «Принцип работы, типы и устройство аккумуляторов»

Урок 8. «Управление полётом мультикоптера. Принцип функционирования полетного контроллера. ПИД регуляторы»

Урок 9. «Основы радиосвязи. Принцип работы радиоаппаратуры управления»

Урок 10. «Аналоговая и цифровая видеотрансляция. Применяемые камеры, радиопередатчики и приёмники»

Урок 11. «Техника безопасности при сборке и настройке коптеров, при подготовке к вылету. Техника безопасности при работе с аккумуляторами»

Урок 12. «Теория ручного визуального пилотирования»

Урок 13. «Техника безопасности при летной эксплуатации коптера»

Урок 14. «Обучение лётному мастерству»

Урок 15. «Основы радиоэлектроники, схемотехники и макетирования электрических схем»

Урок 16. «Основы работы с аналоговым и цифровым сигналом»

Урок 17. «Основы работы с лабораторным оборудованием»

Урок 18. «Теория FPV полетов»

Урок 19. «История автономных полетов. Развитие автопилотов в авиации»

Урок 20. «Основы программирование на языке Python»

Урок 21. «Знакомство с компьютером Raspberry Pi»

Урок 22. «Управление автономным дроном: теория»

Контрольные и проверочные материалы

Проверочные задания включают в себя:

- Контрольные вопросы к каждой теме
- Тесты текущего контроля знаний по темам
- Тест итогового контроля знаний

Проверочные задания охватывают все темы по программе обучения при помощи Конструктора БАС, разрабатываемого в рамках проекта Национальной технологической инициативы «Создание модульного конструктора беспилотной авиационной системы и учебно-методического комплекса на его основе» по направлению дорожной карты НТИ «Модернизация образовательной системы для целей подготовки кадров для рынка Аэронет».

Контрольные вопросы по темам

Знакомство. Принципы проектирования и строение мультикоптеров

1. В какое время появился первый квадрокоптер, и в чём был его недостаток?
2. Чем отличаются БПЛА самолетного типа от обычных самолетов?
3. В каких сферах можно использовать БПЛА самолетного типа?
4. В каких сферах можно использовать коптеры?
5. Какие конфигурации квадрокоптеров бывают?
6. Перечислите название осей коптера.
7. По какому принципу вращаются винты коптера?
8. За что отвечает полётный контроллер?
9. Для чего нужен ESC?
10. Какой вид электродвигателей применяется в коптерах? В чём их преимущество?
11. Какими тремя параметрами обладают воздушные винты?
12. Может ли квадрокоптер летать в вакууме?

Основы электричества

1. Что такое электродвижущая сила?
2. Как найти сопротивление в проводнике используя закон Ома.
3. Чем отличается проводник от диэлектрика?
4. Где применяется первый закон Кирхгофа?
5. Из-за чего в проводнике происходит выделение тепла при прохождении тока?

Теория пайки

1. Какое вещество не допускает окисление?
2. Перечислите основные этапы пайки.
3. Что такое лужение?
4. В каких случаях пайку использовать нельзя?
5. Какой флюс лучше использовать при пайке микросхем.

Аэродинамика полета. Пропеллер

1. За счёт чего образуется сила тяги в пропеллере?
2. Как узнать шаг пропеллера по названию его марки?
3. Что такое пропеллерная константа?
4. Для чего в конструкции коптера одновременно используются пропеллеры, вращающиеся по и против часовой стрелки?
5. Что является исходными данными для подбора винта в коптере?
6. Какие характеристики пропеллера нужны для быстроходного и тихоходного коптера?
7. Определите по таблице к мотору X2204S 2300kv, с каким пропеллером будет развиваться максимальная скорость.

Основы электромагнетизма. Типы двигателей

1. Как, следуя закону Ампера, ведут себя проводники с электрическими токами?
2. По закону Кулона как взаимодействуют относительно друг друга два точечных заряда в вакууме.
3. В чём основное различие коллекторных и бесколлекторных электродвигателей?
4. По каким характеристикам бесколлекторные электродвигатели подходят для использования их на квадрокоптерах?

Бесколлекторные двигатели и регуляторы их хода

1. Зачем нужны датчики в бесколлекторных электродвигателях?
2. На что влияет количество фаз в бесколлекторном электродвигателе?
3. Перечислите основные характеристики контроллеров.
4. Какие ошибки при подключении контроллеров возможно допустить?
5. К каким последствиям могут привести эти ошибки?
6. Перечислите возможные настройки контроллера.

Принцип работы, типы и устройство аккумуляторов

1. Какие устройства называют аккумуляторами?
2. За счёт каких процессов в аккумуляторе накапливается энергия?
3. Что происходит в аккумуляторе во время его заряде и разряде?
4. Опишите два способа соединения аккумуляторов.
5. Какие аккумуляторы применяются при сборке коптеров?

6. Перечислите основные характеристики аккумуляторов.

Управление полётом мультикоптера. Принцип функционирования полетного контроллера. ПИД регуляторы

1. По какому принципу работает полётный контроллер?
2. Перечислите основные задачи полётного контроллера.
3. Сформулируйте принцип работы ПИД-регулятора.

Основы радиосвязи. Принцип работы радиоаппаратуры управления

1. Как происходит передача радиосигнала от передатчика к приёмнику?
2. Чем отличается AM и FM модуляция передачи сигнала?
3. Почему передатчики радиоуправления делают многоканальными?
4. Какая модуляция используется в пультах управления коптерами?
5. По какому принципу работает приёмник радиосигнала?

Аналоговая и цифровая видеотрансляция. Применяемые камеры, радиопередатчики и приёмники

1. Опишите принцип работы аналоговой камеры
2. Опишите принцип работы цифровой камеры.
3. В чём основное отличие аналоговой и цифровой камеры?
4. На какой дистанции можно производить видеосъёмку.
5. Что позволит увеличить дистанцию приёма видеосигнала.
6. Что ещё может повлиять на дальность полёта?

Техника безопасности

1. Назовите меры предосторожности при использовании LiPo аккумуляторов.
2. Чего не следует делать при работе с паяльником?
3. Какие действия нужно выполнить перед взлетом?
4. Что запрещено делать во время полета?

Теория ручного визуального пилотирования

1. Что такое “Arm” и как его выполнить?
2. Что такое “Disarm” и как его выполнить?
3. Что включает в себя предполетная подготовка?

Техника безопасности при летной эксплуатации коптера

1. Какие пункты включает в себя “чеклист”?
2. Назовите правила техники безопасности при полете.

3. Что делать в случае падения и повреждения коптера в полете?

Обучение лётному мастерству

1. Как заармить коптер?
2. Как будет двигаться коптер, если левый стик поднять вверх на 50%, а правый Переместить назад?
3. Какие действия нужно выполнить стиками, чтобы развернуть коптер на 180 градусов?
4. Какие стили полета бывают?
5. Как задизармить коптер?

Основы радиоэлектроники, схемотехники и макетирования электрических схем

1. Каким образом было обнаружено существование электричества и как обосновать это с физической точки зрения?
2. Что такое сопротивление и в чем оно измеряется?
3. Как звучит закон ома для участка цепи?
4. Объясните, в чем отличие аналоговых схем от цифровых?
5. Назовите самые часто встречающиеся компоненты в радиоэлектронных схемах.
6. В чем разница между микрокомпьютером и микроконтроллером?
7. Зачем нужны макетные платы?

Основы работы с аналоговым и цифровым сигналом

1. Какие типы сигналов бывают и чем они отличаются?
2. Объясните, зачем нужны АЦП?

Основы работы с лабораторным оборудованием

1. Что можно измерить при помощи мультиметра?
2. Можно ли при помощи обычного мультиметра измерить напряжение в розетке?
3. Что такое фаза?

Теория FPV полетов

1. Какое оборудование используется при FPV полетах?
2. Какими стиками чаще всего происходит управление при FPV полетах?
3. Какие действия стоит проделать стиками, чтобы полететь вправо?

История автономных полетов. Развитие автопилотов в авиации

1. Приведите пример первых автономных систем и принципы их работы.
2. Как работает автопилот?

3. Какие приборы задействовали изобретатели при создании первых автономно управляемых торпед?
4. Какими углами определяется положение летательного аппарата в пространстве?
5. Почему нельзя было использовать радиосигналы для управления стенфордской тележкой?
6. Как ориентировался первый полностью автономный наземный автопилот?

Основы программирование на языке Python

1. К какому типу языков программирования относится Python?
2. Зачем нужны библиотеки?
3. Объясните, что означает термин “переносимость” программ.
4. Приведите пример кода с использованием оператора ветвления.
5. Какие формы записи могут принимать логические “ложь” и “истина”?
6. Приведите пример кода с использованием цикла for.
7. Приведите пример кода с использованием цикла while.
8. В каком случае используются операторы break и continue?

Знакомство с компьютером Raspberry Pi

1. Что такое микрокомпьютер? Приведите примеры известных вам микрокомпьютеров.
2. Какие устройства можно подключить к Raspberry Pi 3?
3. Каким образом можно соединить полетный контроллер и Raspberry?
4. Какое напряжение требуется для корректной работы Raspberry Pi 3?
5. Откуда происходит загрузка операционной системы при включении Raspberry Pi 3?
6. Что такое SSH клиент и для чего используется?
7. Перечислите основные команды при работе с командной строкой ОС.
8. В какой момент запускаются демоны?
9. Как система обрабатывает комментарии в коде, оставленные программистом?
10. Каким образом можно получить права суперпользователя?
11. Зачем нужен протокол MAVLink?

Управление автономным дроном: теория

1. Почему нельзя летать в помещении, используя GPS координаты?
2. Можно ли автономно летать используя только локальные координаты коптера?
3. Какие устройства нужно установить на коптер для ориентации по специальным меткам?
4. Что включает в себя предполетная подготовка к автономному полету?

Тесты текущего контроля знаний по темам

Знакомство. Принципы проектирования и строение мультикоптеров

1. Кто создал первое беспилотное судно?

1. Альберт Эйнштейн
2. Никола Тесла
3. Исаак Ньютон
4. Чарльз Кеттеринг

2. Как называется коптер с 6 моторами?

1. Пентакоптер
2. Октокоптер
3. Трикоптер
4. Гексакоптер

3. Что такое “тангаж”?

1. Наклон коптера вперед-назад
2. Наклон коптера вправо-влево
3. Вращение коптера вокруг своей оси
4. Набор скорости

4. Где расположены датчики, отвечающие за определение положения коптера в пространстве?

1. В регуляторе оборотов
2. В плате распределения питания
3. В полетном контроллере
4. В пульте радиоуправления

5. Какие типы аккумуляторов бывают?

1. Литий-ионные
2. Литий-полимерные
3. Свинцово-кислотные
4. Никель-металл-гидридные

Основы электричества

1. Как обозначается сопротивление в законе Ома?

1. I
2. R
3. U
4. S

2. Как обнаружить короткое замыкание в цепи?

1. “Прозвонить” мультиметром
2. Измерить напряжение во включенном состоянии
3. Измерить сопротивление в цепи
4. Измерить напряжение в выключенном состоянии

3. При каком типе соединения аккумуляторов напряжение складывается?

1. Последовательное
2. Параллельное
3. Смешанное
4. Замкнутое

4. Электрический ток это -

1. Движение заряженных частиц (электронов).
2. Движение заряженных частиц (протонов).
3. Движение заряженных частиц (бозонов).
4. Движение заряженных частиц (нейтронов).

5. Сумма токов, подходящих к узловой точке электрической цепи, равна

1. Разности токов приходящих к узлу и уходящих от него
2. Полусумме токов, уходящих от этого узла
3. Сумме токов, уходящих от этого узла
4. Произведению токов, уходящих от этого узла

6. Что отражает закон Джоуля-Ленца

1. Направление силы тока и силовых магнитных линий
2. Переход электрической энергии в тепловую
3. Связь электродвижущей силы источника (или электрического напряжения. с силой тока, протекающего в проводнике, и сопротивлением проводника)
4. Соотношение между токами и напряжениями в разветвленных электрических цепях

Теория пайки

1. Чего нельзя делать во время пайки?

1. Соприкасаться жалами двух работающих паяльников
2. Трогать жало паяльника
3. Очищать жало паяльника при помощи металлической губки
4. Паять на температуре свыше 400 градусов

2. Что нужно сделать с проводами перед тем, как спаять их между собой?

1. Изолировать
2. Зачистить
3. Залудить
4. Скрутить

3. За какую часть следует держать паяльник?

1. Фартук
2. Ручка
3. Корпус
4. Жало

4. На каком этапе используется флюс?

1. Лужение
2. Процесс спаивания двух поверхностей
3. Зачистка
4. Скручивание многожильных проводов

5. Какой флюс следует использовать с осторожностью при пайке микросхем?

1. Нейтральный
2. Активированные
3. Пассивный
4. Активный

Аэродинамика полета. Пропеллер

1. К чему ведет увеличение диаметра пропеллера?

1. Уменьшению расхода заряда аккумулятора
2. Увеличению подъемной силы
3. Ускорению набора скорости вращения
4. Замедлению набора скорости вращения

2. Пропеллер с каким количеством лопастей создает наибольшую подъемную силу

1. 2
2. 3
3. 4
4. Подъемная сила не зависит от количества лопастей

3. Что будет если пропеллеры установить в перевернутом виде?

1. Коптер перевернется
2. Коптер будет лететь вниз
3. Коптер взлетит, но с меньшей скоростью
4. Коптер начнет вращаться вокруг своей оси

4. При каких дефектах на воздушном винте нельзя совершать полеты?

1. Трещина на лопасти
2. Лопасть сколота на 20%
3. Лопасть имеет зазубрины
4. Лопасть искривлена

5. В соответствии с какими параметрами моторов БПЛА подбираются пропеллеры?

1. Количество обмоток
2. Мощность двигателя
3. Токопотребление
4. Частота вращения

Основы электромагнетизм1. Типы двигателей

1. Какие моторы чаще всего используются в коптерах?

1. Коллекторные
2. Асинхронные
3. Бесколлекторные
4. Синхронные

2. Отметьте преимущества коллекторных двигателей:

1. Высокий КПД
2. Низкий вес двигателя
3. Продолжительный срок службы
4. Низкая стоимость

3. Отметьте преимущества бесколлекторных двигателей

1. Высокий КПД
2. Низкая стоимость
3. Высокая максимальная скорость
4. Высокая износостойкость

4. Как можно изменить направление вращения бесколлекторного двигателя на коптере?

1. Поменять “+” и “-”
2. Перепрошить регулятор оборотов
3. Поменять между собой 2 фазных провода
4. Это невозможно

5. Как можно изменить направление вращения коллекторного двигателя на коптере?

1. Подать на оба провода ток “+”
2. Поменять “+” и “-”
3. Подать на оба провода ток “-”
4. Это невозможно

Бесколлекторные двигатели и регуляторы их хода

1. Что необходимо использовать для работы бесколлекторного двигателя?

1. Систему охлаждения
2. Стабилизатор напряжения
3. Регулятор оборотов
4. Виброразвязку

2. Как подается ток на обмотки трехфазного бесколлекторного двигателя?

1. Попарно подается ток + и - на обмотки
2. Попарно подается ток - и - на обмотки
3. Попарно подается ток + и + на обмотки

4. Ток подается на все обмотки сразу

3. Какой кратности должно быть число обмоток в бесколлекторном моторе?

1. 2
2. 3
3. 5
4. 7

Принцип работы, типы и устройство аккумуляторов

1. Какая характеристика аккумуляторов влияет на скорость вращения моторов?

1. Емкость
2. Максимальный разрядный ток
3. Напряжение
4. Токоотдача

2. На что влияет емкость аккумулятора

1. На время работы
2. На максимальное выдаваемое напряжение
3. На время заряда заряда аккумулятора
4. На величину тока, которым можно заряжать аккумулятор

3. Каким напряжением можно запитать зарядное устройство Li-Po аккумуляторов для коптеров?

1. 5В
2. 12В
3. 100В
4. 220В

4. Что произойдет в случае прокола Li-Po аккумулятора

1. Вытекание кислоты
2. Возгорание
3. Вздутие аккумулятора
4. Ничего не произойдет

5. Как обозначается трехбаночный аккумулятор?

1. 3C
2. 3S
3. 3V
4. 3G

Управление полетом мультикоптер1. Принцип функционирования полетного контроллера. ПИД регуляторы

1. Что является “мозгом” коптера?

1. Регулятор оборотов (ESC).
2. Плата распределения питания
3. Полетный контроллер
4. Радиоприемник

2. Какие функции не выполняет полетный контроллер?

1. Рассчитывает свое положение в пространстве, по показаниям датчиков
2. Прием сигналов с пульта
3. Вносит корректировку с помощью коэффициентов ПИД
4. Распределяет питание на моторы

3. Что обозначает P в формуле ПИД-регулятора

1. Мощность двигателя
2. Дифференциальная составляющая
3. Погрешность датчиков
4. Пропорциональная составляющая

4. Как обозначаются ШИМ-импульсы?

1. TX
2. PPM
3. PWM
4. RX

5. Как обозначается угол крена?

1. throttle
2. roll
3. force
4. spin

Основы радиосвязи. Принцип работы радиоаппаратуры управления

1. На какой частоте работает аппаратура радиоуправления коптера

1. 0-1 ГГц
2. 1-2 ГГц
3. 2-3 ГГц
4. 3-4 ГГц

2. Какое минимальное количество каналов управления нужно для квадрокоптера?

1. 2
2. 4
3. 6
4. 8

3. Как обозначается фазово-импульсная модуляция?

1. TX
2. PPM
3. PWM
4. RX

4. Какого типа бывают каналы управления?

1. Импульсные
2. Дифференциальные
3. Дискретные
4. Пропорциональные

5. Куда передаются сигналы с радиоприемника в квадрокоптера?

1. На регуляторы оборотов
2. На моторы
3. На полетный контроллер
4. На плату распределения питания

Аналоговая и цифровая видеотрансляция. Применяемые камеры, радиопередатчики и приемники

1. Укажите преимущества аналоговых видеокамер перед цифровыми.

1. Помехозащищенность
2. Высокая взаимосовместимость
3. Просмотр видео в режиме реального времени
4. Высокая надежность

2. Что не относится к возможностям цифровых камер?

1. Возможность работы в паре с датчиком движения
2. Просмотр видео в режиме реального времени
3. Запись видео с точностью до долей секунд
4. Использование встроенного динамика и микрофона

3. Выберите верные утверждения.

1. Дальность передачи видеосигнала не зависит от количества помех в зоне полета
2. Разные системы передачи сигнала имеют различную способность огибать препятствия
3. Дальность полета не зависит от погоды
4. Дальность полета, в большинстве случаев, ограничивается лишь емкостью батареи, но для реализации всего потенциала современных технологий необходима наземная станция

4. Что не относится к схеме работы цифровой камеры?

1. Блок сжатия
2. АЦП
3. Блок оцифровки
4. ПЗС - матрица

5. Что относится к схеме работы аналоговой камеры?

1. Линза
2. Цветофильтр
3. Блок оцифровки
4. Блок сжатия

Техника безопасности при сборке и настройке коптеров, при подготовке к вылету. Техника безопасности при работе с аккумуляторами

1. В какой момент нужно устанавливать пропеллеры на коптер?

1. Перед установкой моторов
2. При сборке защиты коптера
3. При настройке коптера
4. Перед взлетом

2. Что запрещается делать с Li-Po аккумуляторами?

1. Устанавливать на холоде
2. Подключать и отключать держась за разъемы
3. Наносить механические повреждения
4. Нарушать целостность изоляции

3. Выберите неверное утверждение.

1. Паяльник следует хранить в подставке
2. Паять можно только при естественном освещении
3. Нельзя паять включенные в сеть электроприборы
4. Во время пайки следует использовать пинцет и “третью руку”

4. Вы заармили коптер. Пропеллеры коптера вращаются, но он не взлетает. Что следует проверить?

1. Заряд аккумуляторов
2. Правильность установки воздушных винтов
3. Затянутость гаек на моторах
4. Уровень сигнала с пульта радиоуправления

5. Произошла аварийная ситуация и коптер упал. Что следует сделать в первую очередь?

1. Попытаться взлететь снова
2. Убрать коптер с полетной зоны
3. Disarm
4. Проверить целостность защиты

Теория ручного визуального пилотирования

1. Как называется процедура разблокировки моторов коптера?

1. Disarm
2. Kill Switch

3. Arm
4. FPV

2. Что должно произойти в первую очередь при FPV пилотировании?

1. Включение FPV шлема
2. Включение пульта управления
3. Включение питания коптера
4. Включение моторов

3. Что не включает в себя предполетная подготовка

1. Укладка проводов таким образом, чтобы они не попадали под пропеллеры
2. “Прозвонка” платы распределения питания
3. Проверка целостности рамы коптера
4. Правильная установка пропеллеров

4. В какой момент включается пульт радиоуправления?

1. Перед полетом после подключения аккумуляторов
2. Во время предполетной подготовки
3. Перед полетом до подключения аккумуляторов
4. Правильный ответ отсутствует

5. Как называется процедура блокировки (выключения моторов?)

1. Disarm
2. Kill Switch
3. Arm
4. FPV

Техника безопасности при летной эксплуатации коптера

1. Зачем нужен чеклист?

1. Чтобы записать показания заряда аккумуляторов
2. Чтобы отметить время полета
3. Чтобы отметить дальность полета
4. Чтобы верно провести предполетную подготовку

2. На каком минимальном расстоянии от коптера должен находиться пилот во время полета?

1. 0-1 м
2. 1-2 м
3. 2 - 3 м
4. Более 3 м

3. Где находятся зрители во время полета?

1. Слева от пилота, если пилот правша
2. Спереди от пилота на расстоянии 3-5 метров
3. За спиной пилота

4. Справа от пилота, если пилот правша

4. Чего нельзя допускать во время полета?

1. Резких движений стиками
2. Полной разрядки аккумуляторов
3. Полетов выше своего роста
4. Полетов далее 3 метров от себя

5. Укажите правильную последовательность действий при аварийной посадке.

1. Прекратить полёт. Посадить коптер на землю. Выключить пульт. Disarm (стик YAW влево вниз на 3 секунды.. Отключить аккумулятор на коптере.
2. Прекратить полёт. Посадить коптер на землю. Посадить коптер на землю. Отключить аккумулятор на коптере. Disarm (стик YAW влево вниз на секунды). Выключить пульт.
3. Прекратить полёт. Посадить коптер на землю. Disarm (стик YAW влево вниз на 3 секунды). Отключить аккумулятор на коптере. Выключить пульт.
4. Прекратить полёт. Посадить коптер на землю. Disarm (стик YAW влево вниз на 3 секунды). Выключить пульт. Посадить коптер на землю.

Обучение лётному мастерству

1. Как заармить Clover?

1. Яв вправо вниз
2. Яв влево вниз
3. Крен вправо вниз
4. Яв влево вниз

2. Как полететь вправо или влево?

1. Переместить стик в нужную сторону по яву
2. Переместить стик в нужную сторону по крену
3. Переместить стик в нужную сторону по газу
4. Переместить стик в нужную сторону по тангажу

3. Как полететь вперед или назад?

1. Переместить стик в нужную сторону по яву
2. Переместить стик в нужную сторону по крену
3. Переместить стик в нужную сторону по газу
4. Переместить стик в нужную сторону по тангажу

4. Как развернуть коптер вокруг оси, проходящей перпендикулярно плоскости коптера через его центр?

1. Переместить стик в нужную сторону по яву
2. Переместить стик в нужную сторону по крену
3. Переместить стик в нужную сторону по газу
4. Переместить стик в нужную сторону по тангажу

5. Как задизармить Clover?

1. Яв вправо вниз
2. Яв влево вниз
3. Крен вправо вниз
4. Яв влево вниз

Основы радиоэлектроники, схемотехники и макетирования электрических схем

1. В каких единицах измеряется сила тока?

1. [Вольт]
2. [Кулон]
3. [Ампер]
4. [Ом]

2. Какого типа электронных схем не существует?

1. Гибридные
2. Пропорциональные
3. Цифровые
4. Аналоговые

3. Укажите электронный компонент, позволяющий ограничить ток.

1. Светодиод
2. Резистор
3. Конденсатор
4. Трансформатор

4. Укажите электронный компонент, служащий для накопления заряда и энергии электрического поля.

1. Светодиод
2. Резистор
3. Конденсатор
4. Трансформатор

5. Какого типа печатных плат не существует?

1. Замкнутые (ЗПП)
2. Односторонние (ОПП)
3. Двусторонние (ДПП)
4. Многослойные (МПП)

Основы работы с аналоговым и цифровым сигналом

1. Что не может измерить мультиметр?

1. Сопротивление
2. Напряжение
3. Силу тока
4. Правильный вариант ответа отсутствует

2. Какое значение измеряемой величины следует устанавливать на мультиметре?

1. Максимальное
2. Немного меньше предполагаемого значения
3. Немного больше предполагаемого значения
4. Минимальное

3. Зачем нужен режим “прозвонки”

1. Чтобы обнаружить разрывы в цепи
2. Чтобы обнаружить короткое замыкание
3. Чтобы измерить напряжение
4. Чтобы измерить силу тока

4. Что не измеряет осциллограф?

1. Угол сдвига фаз
2. Угловая скорость
3. Частота
4. Напряжение фазы по отношению к земле

5. Какое сопротивление покажет омметр, если соприкоснуть щупы между собой?

1. 0 Ом
2. 1 Ом
3. -1 Ом

Теория FPV полетов

1. Какой стик является основным для позиционирования при FPV полетах?

1. Roll
2. Pitch
3. Yaw
4. Throttle

2. Каким стиком удерживается высота?

1. Roll
2. Pitch
3. Yaw
4. Throttle

3. Что такое FPV пилотирование?

1. Полеты с ориентацией “от первого лица”
2. Полеты с грузом
3. Полеты в помещении
4. Полеты на большой высоте

История автономных полетов. Развитие автопилотов в авиации

1. Автопилот - это

1. БПЛА, который может лететь в заданную точку
2. Устройство или программно-аппаратный комплекс, который может вести вверенное ему транспортное средство по заданной траектории
3. Программа, заставляющая БПЛА лететь в заданную точку
4. Правильный вариант ответа отсутствует

2. Кто изобрел первый автономно управляемый аппарат?

1. Георгий Ботезат
2. Братья Райт
3. Леонардо да Винчи
4. Альфред Уайтхед

3. Какой прибор помогает определить ориентацию летательного аппарата?

1. Осциллограф
2. Гироскоп
3. Барометр
4. Гигрометр

4. Что такое “Фау-2”

1. Коптер
2. Самолет
3. Ракета
4. Спутник

5. Как называется первый полностью автономный наземный автопилот?

1. Фау-2
2. Тележка Леонардо да Винчи
3. Стэнфордская тележка
4. Торпеда Александровского

Основы программирование на языке Python

1. К каким типам языков относится Python?

1. Компилируемый
2. Низкоуровневый
3. Объектно-ориентированный
4. Высокоуровневый

2. Укажите правильную конструкцию.

1.
```
if test1:
    state1
elif test2:
    state2
else:
    state3
```

2.
```
if test1:
    state1
else:
    state2
elif test2:
    state3
```

3.
```
a = int(input(a)
else a < -5:
    print('Low')
elif -5 <= a <= 5:
    print('Mid')
if:
    print('High')
```

4.
```
a = int(input(a)
if a < -5:
    print('Low')
else -5 <= a <= 5:
    print('Mid')
elif:
    print('High')
```

3. Укажите верные обозначения логической истины.

1. 1
2. 0
3. True
4. False

4. Укажите верное обозначение логического оператора “и”.

1. elif
2. or
3. if
4. and

5. Что не является оператором?

1. for
2. continue
3. break
4. while

Тест по теме: «Знакомство с компьютером Raspberry Pi

1. Что такое Raspberry Pi 3?

1. Операционная система

2. Микрокомпьютер
3. Микроконтроллер
4. Процессор

2. Укажите количество ядер микрокомпьютера Raspberry Pi3 model B?

1. 1
2. 2
3. 4
4. 6

3. С помощью какой команды можно перейти в предыдущую директорию?

1. cd ~
2. cd /
3. cd ..
4. cd -

4. Укажите команду, используемую для перехода в директорию.

1. mkdir
2. nano
3. ls
4. cd

5. Как получить права суперпользователя?

1. sudo
2. nano
3. rm
4. ls -l

Управление автономным дроном: теория

1. Какой полетный режим используется для автономных полетов?

1. STABILIZED
2. OFFBOARD
3. ACRO
4. ALTHOLD

2. Почему нельзя ориентироваться только на показания датчиков при автономном полете?

1. Датчики не синхронизированы между собой
2. Коптеру недостаточно информации, получаемой с датчиков, чтобы определить свое положение в пространстве
3. Быстро накапливается ошибка
4. Для полета обязательно нужны глобальные координаты, получаемые с GPS спутника

3. Как называются метки, по которым ориентируется Clover?

1. QR

2. ArUco
3. ID
4. Map

4. Что измеряют сонары?

1. Температуру
2. Расстояние
3. Освещенность
4. Излучение

5. Что такое ROS?

1. Фреймворк
2. Редактор
3. Операционная система для роботов
4. Компилятор

Тест итогового контроля знаний

1. Как называется коптер с 8 моторами?

1. Пентакоптер
2. Октокоптер
3. Трикоптер
4. Гексакоптер

2. Как обозначается напряжение в законе Ома?

1. I
2. R
3. U
4. S

3. При каком типе соединения аккумуляторов напряжение не складывается?

1. Последовательное
2. Параллельное
3. Смешанное
4. Замкнутое

4. Какие типы флюсов следует использовать при пайке микросхем?

1. Нейтральный
2. Активированные
3. Пассивный
4. Активный

5. В соответствии с какими параметрами моторов БПЛА подбираются пропеллеры?

1. Количество обмоток
2. Мощность двигателя
3. Токопотребление

4. Частота вращения

6. Какие моторы редко используются в коптерах?

1. Коллекторные
2. Асинхронные
3. Бесколлекторные
4. Синхронные

7. Отметьте преимущества бесколлекторных двигателей

1. Высокий КПД
2. Низкая стоимость
3. Высокая максимальная скорость
4. Высокая износостойкость

8. Какой кратности должно быть число обмоток в бесколлекторном моторе?

1. 2
2. 3
3. 5
4. 7

9. Как обозначается трехбаночный аккумулятор?

1. 3C
2. 3S
3. 3V
4. 3G

10. Что является “мозгом” коптера?

1. Регулятор оборотов (ESC)
2. Плата распределения питания
3. Полетный контроллер
4. Радиоприемник

11. Какое минимальное количество каналов управления нужно для квадрокоптера?

1. 2
2. 4
3. 6
4. 8

12. Как обозначаются ШИМ-импульсы?

1. TX
2. PPM
3. PWM
4. RX

13. Что не относится к возможностям цифровых камер?

1. Возможность работы в паре с датчиком движения

2. Просмотр видео в режиме реального времени
3. Запись видео с точностью до долей секунд
4. Использование встроенного динамика и микрофона

14. В какой момент нужно устанавливать пропеллеры на коптер?

1. Перед установкой моторов
2. При сборке защиты коптера
3. При настройке коптера
4. Перед взлетом

15. Как называется процедура разблокировки моторов коптера?

1. Disarm
2. Kill Switch
3. Arm
4. FPV

16. Где находятся зрители во время полета?

1. Слева от пилота, если пилот правша
2. Спереди от пилота на расстоянии 3-5 метров
3. За спиной пилота
4. Справа от пилота, если пилот правша

17. В каких единицах измеряется напряжение?

1. [Вольт]
2. [Кулон]
3. [Ампер]
4. [Ом]

18. Что не является оператором?

1. for
2. continue
3. break
4. while

19. Какой полетный режим используется для автономных полетов?

1. STABILIZED
2. OFFBOARD
3. ACRO
4. ALTHOLD