УДК: 004.896

Регистрационный №

Инв. №

НАУЧНО-ТЕХНИЧЕСКИЙ ОТЧЕТ

о работ теме:

«Web-платформа для управления малых беспилотных аппаратов»

Казаков Р. И.

подпись дата

Хабаровск, 2019

РЕФЕРАТ

Научно-технический отчет содержит 52 страницы текстового документа формата А4, включающего 15 рисунков, 2 таблицы, 11 литературных источников.

ИНТЕРНЕТ ВЕЩЕЙ, БЕСПИЛОТНЫЙ АППАРАТ, ДРОН, ПЕРЕДАЧА СООБЩЕНИЙ, ПЛАТФОРМА

Объектом разработки является техническое решение управления малыми дронами с Web-платформы.

Цель работы – разработка приложения, обеспечивающего функции удалённого доступа к дрону, находящегося под управлением любой ОС семейства Linux, на основе протокола MQTT.

В процессе работы произведён обзор существующих решений в области IoT-платформ, осуществлён их анализ. Так же в работе разобран и реализован на прикладном уровне протокол передачи сообщений, подходящий для различных устройств концепции «Интернет вещей».

Приведены результаты разработки приложения на языке Python, программной платформе ThingsBoard v2.3.1. А также создано руководство пользователя, которое содержит информацию о том, как работать с приложением.

Разработанная программа может использоваться пользователями, которым требуется установить управление с беспилотным аппаратом под управлением программы MAVProxy.

Введение 6

1 Выбор одной из существующих Web-платформы для подключения устройств Интернета вещей 8

1.1 Краткий обзор концепции «Интернет вещей» 8

1.2 Критерии для выбора платформы 9

1.3 Краткий обзор протоколов обмена сообщениями для устройств Интернета вещей 10

1.3.1 MQTT 10

1.3.2 STOMP 10

1.3.3 CoAP 11

1.4 Краткий обзор существующих IoT-платформ 11

1.4.1 Mainflux IoT 11

1.4.2 ThingsBoard 12

1.4.3 Thinger.io 13

1.4.4 Kaa IoT 13

1.5 Краткий обзор существующих прошивок для дронов 14

1.5.1 ArduPilot 14

1.5.2 PX4 14

1.5.3 BetaFlight 15

1.5.4 CleanFlight 16

1.6 Обоснование выбора платформы 16

2 Проектирование решения 18

2.1 Краткий обзор схемы разработанного решения 18

2.2 MAVLink 19

2.3 Выбор и обоснование средств программирования 22

2.4 Описание основных функций программ 22

2.4.1 Отправление телеметрии на платформу 22

2.4.2 Выполнение команд по запросу с платформы 28

2.5 Вывод по разделу 34

3 Руководство пользователя 36

3.1 Назначение программ 36

3.2 Системные требования программ 36

3.3 Первичная настройка платформы 36

3.3.1 Особенности установки платформы ThingsBoard 36

3.3.2 Настройка платформы 37

3.4 Настройка программ 39

3.5 Вторичная настройка платформы 45

3.6 Тестирование решения 47

Заключение 50

Список использованных источников 51

Раздаточный материал:

Текстовая часть На 5 отдельных листах

Диск DVD В конверте на обороте обложки

ВВЕДЕНИЕ

Научно-технический прогресс не стоит на месте. Придумываются новые устройства разного рода: для развлечения, науки, профессиональных нужд, автоматизации и др. Рассматривая метод подключения устройств друг к другу по сети интернет, можно выделить концепцию вычислительной сети «Интернет вещей» (англ. internet of things, IoT). Термин появился в 1999 году, как представление потенциальных возможностей устройств радиочастотной идентификации для взаимодействия между собой и с внешним окружением.

Одной из ключевой категории таких «вещей» являются дроны. Дрон – это разговорный термин беспилотного аппарата, который может управляться запрограммированным алгоритмом или человеком удаленно. Если дрон контролируется постоянно, то его называют дистанционно-пилотируемым аппаратом. Их основные преимущества: возможность автоматизировать какой-либо процесс, мобильность, возможность применения в условиях риска для человека.

Для управления беспилотного аппарата используется различное ПО: от разработчика, с открытым исходным кодом или комплексное использование различных программных инструментов. Обязательное правило для таких решений – это безопасность устройства от угона. Для этого необходимо шифровать сетевой трафик, для исключения возможности его подмены в режиме реального времени.

Говоря о проблеме, в мире существует очень мало решений для управления дронами через web-платформу, а в открытом доступе мною не было найдено вовсе.

Цель данной выпускной работы – разработать решение с автоматизированной web-платформы для управления малых беспилотных аппаратов. Для разработки программ и достижения цели необходимо решить следующие задачи:

– Подобрать наиболее подходящую IoT-платформу с открытым исходным кодом по критериям

– Разработать программу для передачи телеметрии с дрона на платформу

– Разработать программу для удаленного управления командами на дроне с платформы

1 ВЫБОР ОДНОЙ ИЗ СУЩЕСТВУЮЩИХ WEB-ПЛАТФОРМЫ ДЛЯ ПОДКЛЮЧЕНИЯ УСТРОЙСТВ ИНТЕРНЕТА ВЕЩЕЙ

В первой главе рассматривается понятие концепции «Интернет вещей», современных протоколов для передачи сообщений между устройствами, основные современные прошивки для беспилотных аппаратов и краткий обзор существующих web-платформ с открытым исходным кодом.

1.1 Краткий обзор концепции «Интернет вещей»

Интернет вещей (англ. internet of things, IoT) – это сеть, связанных через интернет, объектов, способных собирать данные и обмениваться данными, поступающими со встроенных сервисов. Устройства, входящие в интернет вещей, – любые автономные устройства (в частности, дроны), подключённые к интернету, которые могут отслеживаться и/или управляться удалённо [1]. Общая схема представления изображена на рисунке 1.



Рисунок 1 – общая схема представления концепции интернета вещей

Концепция сети интеллектуальных устройств обсуждалась еще в 1982 году, когда модифицированный торговый автомат Coke в Университете Карнеги-Меллона стал первым устройством, подключенным к Интернету. Автомат стал способен сообщить о своем инвентаре и о температуре напитков. Документ Марка Вайзера 1991 года о вездесущих вычислениях, «Компьютер 21-го века», а также академические центры, такие как UbiComp и PerCom, дали современное видение IoT. В 1994 году Реза Раджи описал концепцию в IEEE Spectrum как «перемещение небольших пакетов данных на большой набор узлов, чтобы интегрировать и автоматизировать все, от бытовой техники до целых фабрик». В период с 1993 по 1997 годы несколько компаний предлагали такие решения, как Microsoft at Work или Novell NEST. Область получила всплеск популярности, когда Билл Джой представил связь между устройствами как часть своей структуры «Шесть сетей», представленной на Всемирном экономическом форуме в Давосе в 1999 году [2].

1.2 Критерии для выбора платформы

Беспилотные аппараты должны работать через Web-платформу, поддерживающую устройства интернета вещей, а значит должна соблюдать следующие требования:

– Иметь возможность оповещения через email, для случая экстренных событий

– Поддерживать стабильные и защищенные соединения для передачи сообщений

– Иметь интуитивно-понятный интерфейс для обычного пользователя

– Платформа должна поддерживаться разработчиками и выпускать обновления для нее

– Иметь открытый исходный код

– Поддерживать хотя бы минимальный инструментарий для администрирования платформы

1.3 Краткий обзор протоколов обмена сообщениями для устройств Интернета вещей

Так как одним из главных критериев является надежность соединения, то необходимо иметь представления о существующих популярных типах подключения для устройств интернета вещей.

1.3.1 MQTT

MQTT (англ. message queuing telemetry transport) – это чрезвычайно простой и легкий протокол обмена сообщениями, созданный для устройств с ограниченными возможностями или с аккумуляторной батареей маленькой емкости и сетей с низкой пропускной способностью, высокой задержкой или нестабильным подключением [3].

Обмен сообщениями в протоколе MQTT осуществляется между «клиентом» (client), который может быть «издателем» или «подписчиком» (publisher/subscriber) сообщений, и брокером (broker) сообщений (например, Mosquitto MQTT). Издатель отправляет данные на MQTT брокер, указывая в сообщении определенную тему – «топик» (topic).

Шифрование в сети может быть обработано с помощью SSL, но стоит отметить, что SSL не является самым легким из протоколов и действительно добавляет значительную нагрузку на сеть.

Минимальное управляющее сообщение MQTT может составлять всего два байта данных, но при необходимости, оно может быть размером приблизительно до 256 мегабайт данных. Существует четырнадцать определенных типов сообщений, используемых для подключения и отключения клиента от брокера, для публикации данных, подтверждения получения данных и для контроля соединения между клиентом и сервером.

1.3.2 STOMP

STOMP (англ. Simple или Streaming Text Oriented Message Protocol) – это простой текстовый протокол, разработанный для работы со связующим программным обеспечением, ориентированным на обработку сообщений (англ. message-oriented middleware, MOM). Он обеспечивает совместимый формат соединения, который позволяет клиентам STOMP общаться с любым брокером сообщений, поддерживающим протокол. Протокол в целом похож на HTTP и работает через TCP [4].

1.3.3 COAP

CoAP (англ. Constrained Application Protocol) – это специализированный интернет-протокол приложений для ограниченных устройств, как определено в RFC 7252. Он позволяет тем ограниченным устройствам, которые называются «узлами», обмениваться данными с более широким Интернетом, используя аналогичные протоколы. CoAP предназначен для использования между устройствами в одной ограниченной сети (например, с низким энергопотреблением и сетями с потерями), между устройствами и общими узлами в Интернете, а также между устройствами в различных ограниченных сетях, соединенных Интернетом. CoAP также используется через другие механизмы, например в SMS через сети мобильной связи [5].

1.4 Краткий обзор существующих IoT-платформ

1.4.1 Mainflux IoT

Mainflux - это высокотехнологичная и безопасная IoT-платформа с открытым исходным кодом, обладающая большими возможностями для разработки решений Интернета вещей, IoT-приложений и интеллектуальных продуктов. На рисунке 2 изображена общая схема представления функционала платформы [6].

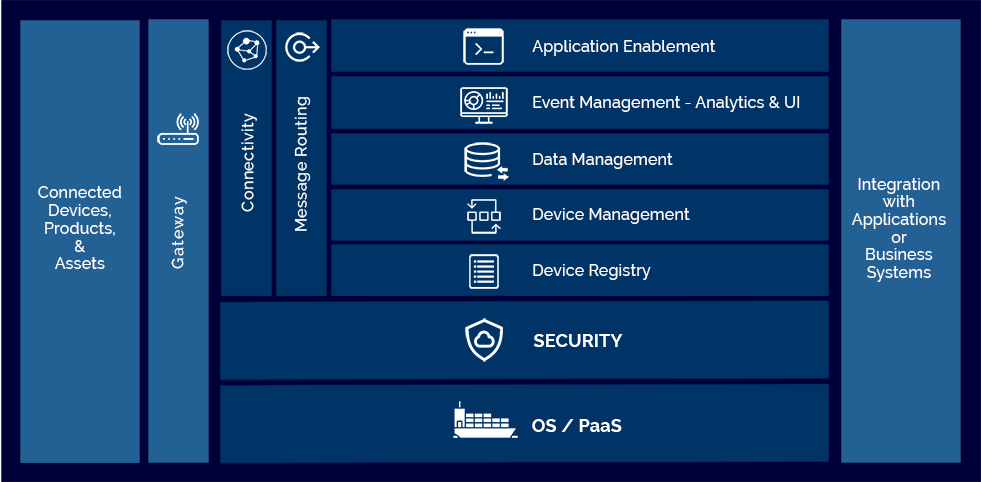


Рисунок 2 – общая схема представления возможностей платформы

В нем включено следующее промежуточное программное обеспечение:

– Панель устройств

– Панель управления данных

– Менеджер событий

– Пользовательский интерфейс

1.4.2 ThingsBoard

ThingsBoard - это IoT-платформа с открытым исходным кодом для сбора, обработки, визуализации данных и управления устройствами. Имеет огромную популярность среди любителей робототехники из-за простоты настройки и удобного интерфейса. Платформа обеспечивает подключение устройств через стандартные протоколы интернета вещей - MQTT, CoAP и HTTP. Поддерживает облачные, так и локальные развертывания. Имеет огромный функционал API. Существует возможность настроить программное взаимодействие между устройствами.

1.4.3 Thinger.io

Thinger.io - это платформа с открытым исходным кодом, разработанная для быстрого и доступного создания какого-нибудь проекта. Разработчики позволяют бесплатно пользоваться платформой с их сайта через регистрацию, но с ограниченным количеством подключенных устройств. Чтобы убрать это ограничение, нужно развернуть платформу на собственном сервере или купить тарифный план у разработчиков. Идеально подходит для личного пользования или компании с небольшим количеством доверенных лиц, потому что не имеет доступной административной панели для управления пользователями [7].

1.4.4 Kaa IoT

Каждая функция платформы Kaa, такая как обмен сообщениями или визуализация, представляет собой набор микросервисов, которые выполняют свои специфические функции. Микросервисы упрощают настройку и обеспечивают эффективное разделение проблем между различными частями платформы [8].

Микросервисы Kaa взаимодействуют через открытые API и могут быть интегрированы со сторонними системами и перестроены [8].

Платформа также идеально подходит для гибридных развертываний, например, когда компания запускает ее в своем собственном центре обработки данных, то может масштабировать его в облаке каждый раз, когда возникает потребность в увеличении производительности [8].

Платформа не является бесплатной и имеет лишь частичный открытый исходный код.

1.5 Краткий обзор существующих прошивок для дронов

1.5.1 ArduPilot

Ardupilot был одним из первопроходцев программного обеспечения беспилотных аппарат с открытым исходным кодом. Проект начался с аппаратного обеспечения Arduino (отсюда и часть «Ardu») и превратился в локомотив индустрии дронов [9].

APM в программе APM Planner также используется для обозначения программного обеспечения ArduPilot.

Сообщество ArduPilot действительно активно, и его форум - отличный инструмент для участия в проекте.

ArduPilot может работать на различных типах транспортных средств, включая: мультироторы, самолеты, вездеходы и даже подводные лодки. Ardupilot стремится сделать эти машины полностью автономными [9].

Существует также разнообразная коллекция плат контроллеров полета, с которыми совместим ArduPilot. Стек полета ArduPilot довольно продвинут, поскольку он использует популярный Mission Planner в качестве наземной станции управления и MAVLink в качестве промежуточного программного обеспечения [9].

1.5.2 PX4

PX4 является частью проекта Dronecode, некоммерческой организации, управляемой Linux Foundation. Dronecode стремится обеспечить развивающуюся индустрию дронов платформой с открытым исходным кодом [9].

Dronecode предоставляет вам платформу «операционной системы», освобождая своих пользователей для разработки многих необходимых инструментов поверх платформы [9].

ArduPilot раньше был проектом программного обеспечения для управления полетом в Dronecode, который внес свой вклад в полу-соперничество ArduPilot против PX4 [9].

PX4 вышли из проекта Pixhawk, который является аппаратным проектом с открытым исходным кодом, поскольку они нуждались в некотором программном обеспечении дрона с открытым исходным кодом для запуска своих плат [9].

Одно из ключевых отличий PX4 от ArduPilot заключается в том, что он работает по лицензии BSD. По сути, это означает, что, если вы добавляете новую функцию в проект PX4 в приватной ветке, вам не нужно возвращать изменения в родительский проект PX4 [9].

Эта функция делает PX4 более привлекательной для компаний, стремящихся защитить свою интеллектуальную собственность. По этой причине, и их включение в проект Dronecode, PX4 получает гораздо больше средств для разработки, чем ArduPilot [9].

PX4 также поддерживается на большом количестве плат управления полетом. Это также не лучший выбор для энтузиастов с FPV(англ. First Person View, вид от первого лица)-квадракоптерами, так как он больше ориентирован на автономные дроны [9].

Учитывая все обстоятельства, PX4 и ArduPilot все еще очень похожи по функциональности [9].

1.5.3 BetaFlight

Программное обеспечение контроллера полета BetaFlight ориентировано на выполнение ручного полета, что делает его отличным выбором для любителей FPV-дронов. На сегодняшний день это, пожалуй, самый популярный проект с открытым исходным кодом для дронов FPV [9].

BefaFlight в основном используется с квадрокоптерами, но также может использоваться на самолетах с неподвижным крылом. Это широко поддерживаемый проект программного обеспечения для управления полетом на популярных платах автопилота FPV. По сути, если на плате используется хотя бы процессор STM32F3, вероятно, BetaFlight будет поддерживаться [9].

BetaFlight является полностью открытым исходным кодом и имеет значительную поддержку сообщества.

BetaFlight также поддерживает такие крупные RC (англ. Radio Control)-продукты, как FrSky, FlySky and Graupner [9].

1.5.4 CleanFlight

Программный проект CleanFlight также чрезвычайно популярен для четырехъядерных FPV-приложений [9].

Он основан на устаревшем коде MultiWii, но обновил прошивку с 8-битного до 32-битного проекта [9].

CleanFlight работает под лицензией GPL, поэтому любые изменения должны быть перенесены обратно в основную ветку [9].

Как и BetaFlight, CleanFlight также поддерживается на самых разных платах автопилота, так как он доступен на 8 различных досках [9].

У них отличная документация и обширные онлайн-ресурсы, которые помогут вам в решении любых проблем [9].

В конечном счете, CleanFlight и BetaFlight очень похожи, и выбор между ними может быть просто вопросом предпочтения [9].

1.6 Обоснование выбора платформы

Для проектирования решения я решил выбрать платформу ThingsBoard. Возможно, на момент написания пояснительной записки, платформа ThingsBoard имеет меньший функционал, чем у Mainflux, но в отличии от остальных разработок, в ней присутствует относительно дружественный интерфейс и хорошую документацию по использованию платформы (на английском языке), а также ее популярность позволяет своевременно получить ответ на какой-либо вопрос от пользователей на web-сайте «https://github.com/thingsboard/thingsboard/issues»

2 ПРОЕКТИРОВАНИЕ РЕШЕНИЯ

В данной главе будет рассмотрена общая схема подключения, программный код и тестирование решения

2.1 Краткий обзор схемы разработанного решения

На рисунке 3 изображена схема подключения дрона. Такой вариант подключения также подходит для подключения нескольких дронов к одной платформе. Иметь к дрону подключенный GSC (англ. Ground Control Station) клиент не обязательно, но рекомендуется для первого тестирования на новом дроне, так как совместимость может быть не полной и программы могут выполнить команду не так, как нужно.

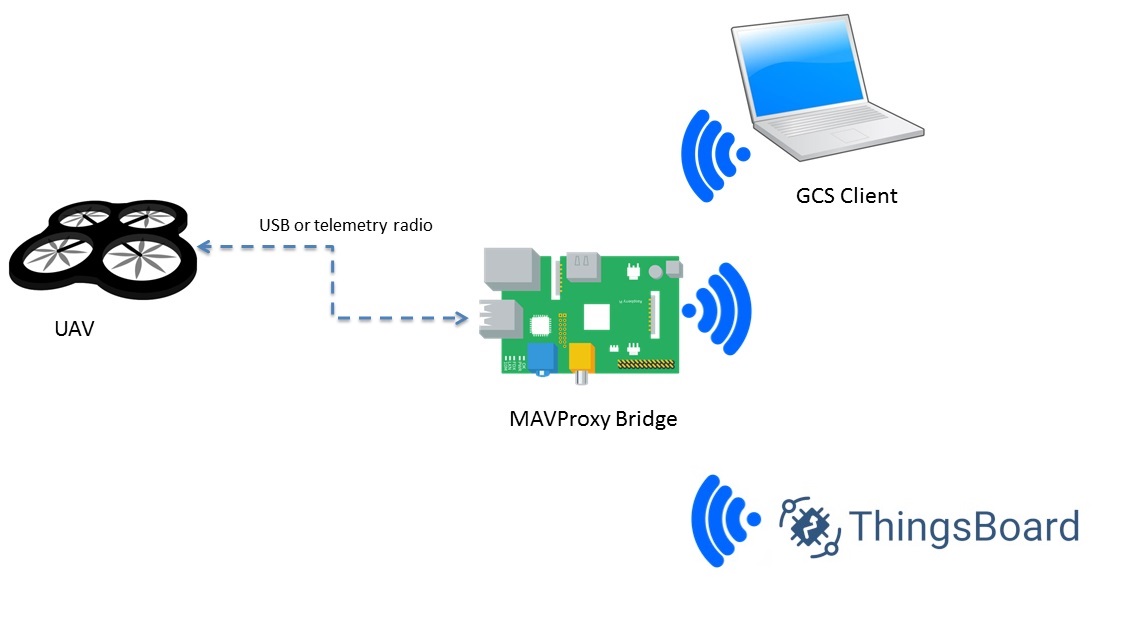


Рисунок 3 – Схема подключения

К дрону подключается переносимый микрокомпьютер (например Raspberry Pi), в котором есть доступ к интернету или к локальной сети (в зависимости от типа соединения между платформой и дроном). MAVProxy устанавливается на микрокомпьютер, который обеспечивает соединение между дроном (по протоколу MAVLink) и программами. Программы, в свою очередь, выполняют заданные команды, взаимодействуя с платформой ThingsBoard.

2.2 MAVLink

MAVLink (Micro Air Vehicle Link) — протокол обмена информацией между GCS-клиентом (англ. Ground control station, наземная станция) а также её компонентами и любыми малыми беспилотными аппаратами (MAV - Micro Air Vehicle): летающими, ездящими, плавающими, и другими [10].

MAVLink распространяется под лицензией LGPL в виде модуля для python и генератора библиотек под различные языки, в том числе С/C++ библиотеки. Есть так же репозитории уже сгенерированных библиотек для MAVLink версии v1 и версии v2 [10].

Базовый пакет MAVLink представлен на рисунке 4.

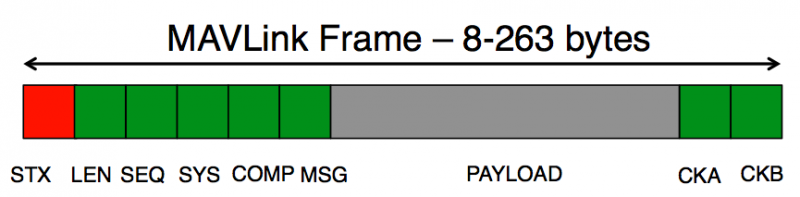


Рисунок 4 – стандартный формат пакета протокола MAVLink

Первый байт пакета STX используется как символ начала пакета, где определяется версия протокола:

– 0xFD для версии v2.0

– 0xFE для версии v1.0

– 0x55 для версии v0.9

Далее в пакете:

– LEN – длина информационной части пакета

– SEQ – счётчик пакета (0-255), который поможет выявить потерю пакетов

– SYS (System ID) – идентификатор отправляющей системы

– COMP (Component ID) – идентификатор отправляющего компонента

– MSG (Message ID) – тип сообщения, от него зависит, какие данные будут лежать в информационной части пакета

– PAYLOAD – информационная часть пакета: сообщение, размером от 0 до 255 байт

– CKA и CKB – нижний и верхний байты, соответственно, содержат контрольную сумму пакета

Библиотека MAVLink позволяет кодировать и раскодировать пакеты согласно протоколу, но она не регламентирует, какими аппаратными и программными средствами данные будет отправлены — это могут быть TCP/UDP сообщения, обмен через последовательный порт или что-то другое, что позволяет использовать двухсторонний обмен. Каждая система или компонент, может одновременно обмениваться данными по разным источникам, но тогда для каждого источника назначается специальный идентификатор, называемый channel (канал). MAVLink содержит буфер на каждый канал [10].

В MAVLink существуют различные встроенные типы сообщений, а так же есть возможность добавлять собственные [10].

Какие данные считать полётными, какие сообщения отправляются периодически, а какие только по запросу - полностью решает полётный контроллер. MAVLink не декларирует, какими сообщениями необходимо пользоваться (кроме HEARTBEAT): при проектировании полётных систем разработчики сами решают какие сообщения программное обеспечение будет обрабатывать, а какие – отправлять [10].

Некоторые общие сообщения, которые должны быть реализованы в большинстве полётных контроллеров и наземных станциях управления (GCS-клиент) вне зависимости от диалекта:

– HEARTBEAT – это самый базовый пакет, в котором система говорит, что она существует, и что оно такое. Это единственный тип пакета, который MAVLink обязывает к использованию [10].

– Сообщение типа ATTITUDE описывает поворотное положение беспилотного аппарата относительно его центра в пространстве – углы тангажа, крена и рыскания [10].

– В пакетах типа VFR\_HUD сгруппированы параметры, которые обычно выводятся на экран в качестве телеметрийных данных OSD (англ. On Screen Display, экранное меню). К этим параметрам MAVLink относит: воздушная скорость, путевая скорость, высота над уровнем моря, скороподъёмность, направление и газ (throttle) [10].

– В сообщениях типа LOCAL\_POSITION и GLOBAL\_POSITION передаются данные о положении аппарата в пространстве, которое определено с помощью локальной или глобальной системы позиционирования соответственно. Пакеты подразумевают уже обработанные, фильтрованные данные. Для сырых показаний GPS сенсора необходимо обрабатывать пакеты GPS\_RAW и GPS\_STATUS [10].

– Полётный контроллер отсылает пакеты с определённой частотой, которую он определяет сам, на основе настроек, скорости передачи данных или типа канала связи. Тем не менее для задания частоты отправки каких-либо данных предусмотрен запрос, путём отправки сообщения MESSAGE\_INTERVAL с указанием идентификатора нужного сообщения и интервалом в микросекундах [10].

2.3 Выбор и обоснование средств программирования

В открытом доступе существует множество SDK (от англ. software development kit) для разработки прикладного программного обеспечения беспилотных аппаратов. Из них самые распространенные:

– DroneKit

– Комплект SDK от DJI

– Dronecode SDK

– Parrot Ground SDK

Среди всех выше перечисленных я выбрал DroneKit. Основные его преимущества:

– Большая и очень понятная документация с примерами

– Простота программирования

– Стабильность в работе, основанная на многолетней поддержке проекта

– Позволяет быстро и удобно установить симулятор дрона для тестирования программ

– Поддержка большинства прошивок с открытым исходным кодом

API, который предлагает DroneKit, используется на языке Python. Python – это высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика, при этом стандартная библиотека имеет огромный функционал.

2.4 Описание основных функций программ

2.4.1 Отправление телеметрии на платформу

За обработку и отправление на сервер телеметрических данных отвечает скрипт «mqttsendtel.py». Через DroneKit SDK программа считывает телеметрию через MAVProxy, который общается с дроном по протоколу MAVLink [11]. Далее будут описаны основные методы в коде программы.

**Синтаксис вызова:**

argparse.ArgumentParser()

**Описание:**

Объект ArgumentParser будет содержать всю информацию, необходимую для разбора командной строки на типы данных Python, в том числе ту, которая будет отображаться пользователю.

**Пример реализации в программе:**

parser = argparse.ArgumentParser(description='Commands vehicle using mqtt thingsboard.')

**Синтаксис вызова:**

parser.add\_argument()

**Описание:**

Заполнение ArgumentParser информацией об аргументах программы выполняется путем вызова метода add\_argument(). Обычно эти вызовы сообщают ArgumentParser, как взять строки в командной строке и переместить их в объекты. Эта информация сохраняется и используется при вызове parse\_args().

**Пример реализации в программе:**

parser.add\_argument('--connect',

help="Vehicle connection target string. If not specified, SITL automatically started and used.")

**Синтаксис вызова:**

parser.parse\_args()

**Описание:**

ArgumentParser анализирует аргументы с помощью метода parse\_args(). Это проверит командную строку, преобразует каждый аргумент в соответствующий тип и затем вызывает соответствующее действие. В большинстве случаев это означает, что простой объект класса Namespace будет создан из атрибутов, проанализированных из командной строки.

**Синтаксис вызова:**

if not connection\_string:

**Описание:**

Если в аргументе, при запуске программы, не было ничего указана, то выполнится запуск симулятора dronekit\_sitl.

**Пример реализации в программе:**

if not connection\_string:

import dronekit\_sitl

sitl = dronekit\_sitl.start\_default()

connection\_string = sitl.connection\_string()

**Синтаксис вызова:**

connect(connection\_string, wait\_ready=True)

**Описание:**

Происходит подключение к беспилотному аппарату по указанным в аргументах данным. Вызов присваивается указателю, который далее будет использоваться как обращение к устройству. Реализован в библиотеке dronekit.

**Пример реализации в программе:**

vehicle = connect(connection\_string, wait\_ready=True)

**Синтаксис вызова:**

time.time()

**Описание:**

Возвращает текущее системное время в тиках с 12:00 1 января 1970 г.

**Пример реализации в программе:**

next\_reading = time.time()

**Синтаксис вызова:**

mqtt.Client()

Конструктор Client() принимает следующие аргументы:

– client\_id – это уникальная строка идентификатора клиента, используемая при подключении к брокеру. Если client\_id имеет нулевую длину или тип None, то он будет сгенерирован случайным образом. В этом случае параметр clean\_session должен иметь значение True.

– clean\_session – это логическое значение, определяющее тип клиента. Если True, брокер удалит всю информацию об этом клиенте, когда он отключится. Если значение False, то клиент информация о подписке и сообщениях в очереди будут сохраняться при отключении клиента.

– userdata – пользовательские данные любого типа, которые передаются в качестве параметра для обратных вызовов. Позже он может быть обновлен с помощью функции user\_data\_set().

– protocol – версия протокола MQTT для использования с этим клиентом.

– transport – настроить «websockets» для отправки MQTT через WebSockets. Если оставить значение по умолчанию («tcp»), то будет использоваться стандартный TCP без изменений.

**Пример реализации в программе:**

client = mqtt.Client()

**Синтаксис вызова:**

client.username\_pw\_set()

**Описание:**

Назначить имя пользователя в клиенте. В случае для подключения к платформе ThingsBoard, используется access token, который назначается на платформе.

**Пример реализации в программе:**

client.username\_pw\_set(ACCESS\_TOKEN)

**Синтаксис вызова:**

client.loop\_start()

**Описание:**

Запускает новый поток, который регулярно вызывает метод loop. Он также обрабатывает повторное подключение автоматически при необходимости.

**Синтаксис вызова:**

vehicle.location.global\_relative\_frame.lat

**Описание:**

Возвращает значение широты по GPS. Аналогичный вызов происходит для: долготы, градусов наклона, скорости относительно земли и высоты.

**Пример реализации в программе:**

gps\_lat = vehicle.location.global\_relative\_frame.lat

**Синтаксис вызова:**

client.publish()

**Описание:**

Публикует сообщение на брокер. В случае платформы ThingsBoard, часть сообщения должна содержать 'v1/devices/me/telemetry', чтобы платформа поняла, какого рода сообщение ей отправили.

**Пример реализации в программе:**

client.publish('v1/devices/me/telemetry', json.dumps(raspb\_sensor\_data), 1)

**Синтаксис вызова:**

client.loop\_stop()

**Описание:**

Останавливает поток, который регулярно вызывал метод loop.

**Синтаксис вызова:**

client.disconnect()

**Описание:**

Выключает соединение клиента.

**Синтаксис вызова:**

vehicle.close()

**Описание:**

Закрывает соединение с устройством, тем самым освобождает порт для подключения чем-либо.

**Синтаксис вызова:**

sitl.stop()

**Описание:**

Закрывает симулятор. В программе выполняется только в случае, если симулятор был запущен. Реализован в библиотеке dronekit.

**Пример реализации в программе:**

if sitl:

sitl.stop()

2.4.2 Выполнение команд по запросу с платформы

Работа программы состоит из двух скриптов. За принятие данных с платформы и их обработку отвечает скрипт «mqttrpc1.py». Программа подписывается на сообщения, которые отправляет платформа ThingsBoard, и, в зависимости от характера сообщения, может выполнять автоматически следующие задачи:

– Запуск скрипта «commandscript1.py»

– Вывод тестового сообщения в терминале

– Перезагружать устройство

При запуске по запросу с платформы скрипта «commandscript1.py» происходит два перемещения по координатам и возвращение на базу с 30-секундной задержкой после каждой команды. Если беспилотный аппарат за 30 секунд не успел выполнить команду, то начинается выполнение следующей. После отправление команды по возвращению на базу, скрипт автоматически закрывается. Управлением дрона из скрипта выполняется через библиотеки DroneKit SDK, с включенным MAVProxy, который в свою очередь обращается к дрону через протокол MAVLink. Далее будут описаны основные методы в коде программ, кроме тех, которые уже упоминались.

Сначала будут описаны вызовы из «mqttrpc1.py»

**Синтаксис вызова:**

json.loads()

**Описание:**

Преобразует объект (типа str, bytes или bytearray, содержащий документ JSON) в объект Python, используя таблицу преобразования, как указано в таблице 1.

|  |  |
| --- | --- |
| JSON | Python |
| object | dict |
| array | list |
| string | str |
| number (int) | int |
| number (real) | float |
| true | True |
| false | False |
| null | None |

Таблица 1 – таблица преобразований, используемая в json.loads()

Если преобразованные данные не являются допустимым документом JSON, будет вызван JSONDecodeError.

**Пример реализации в программе:**

data = json.loads(msg.payload)

Зависимость json.loads() с JSONDecodeError необходимо также рассмотреть, хотя напрямую в программе не используется.

**Синтаксис вызова:**

json.JSONDecodeError()

**Описание:**

Подкласс ValueError со следующими дополнительными атрибутами:

– msg – неформатированное сообщение об ошибке

– doc – сообщение об анализе документа JSON

– pos – начальный индекс документа, при котором синтаксический анализ не удался

– lineno – информация об линии, которая соответствует атрибуту pos

– colno – информация об столбце, которая соответствует атрибуту pos

**Синтаксис вызова:**

json.dumps()

**Описание:**

Сериализует объект в форматированную JSON-строку, используя таблицу преобразования, которая указана в таблице 2.

Таблица 2 – таблица преобразований, используемая в json.dumps()

|  |  |
| --- | --- |
| Python | JSON |
| dict | object |
| list, tuple | array |
| str | string |
| int, float, int- & float-производные Enums | number |
| True | true |
| False | false |
| None | null |

**Пример реализации в программе:**

return json.dumps(gpio\_state)

**Синтаксис вызова:**

os.system()

**Описание:**

Исполняет системную команду, возвращает код её завершения (в случае успеха 0). В данной программе запускает скрипт «commandscript1.py» с аргументом --connect, после которого пишется ip, по которому скрипт будет подключаться к MAVProxy.

**Пример реализации в программе:**

os.system("gnome-terminal -- python commandscript1.py --connect 127.0.0.1:14551")

**Синтаксис вызова:**

os.popen()

**Описание:**

Метод открывает канал в или из команды. Возвращаемое значение - это открытый файловый объект, подключенный к каналу, который может быть прочитан или записан в зависимости от того, какой стоит режим: «r» (по умолчанию – режим чтения) или «w» (запись). В данной программе используется для введения команды перезагрузки устройства из терминала.

Имеет следующие параметры:

– command – используемая команда

– mode – этот режим может быть «r» (чтение) или «w» (запись).

– bufsize – Значение буферизации. Если установлено 0, буферизация не будет выполняться. Если значение буферизации равно 1, буферизация строки будет выполняться при доступе к файлу. Если будет указано значение буферизации как число типа int больше 1, то будет выполнено действие буферизации с указанным размером буфера. Если значение отрицательно, то размер буфера измениться на системное значение по умолчанию.

**Пример реализации в программе:**

os.popen("sudo reboot")

**Синтаксис вызова:**

client.loop\_forever()

**Описание:**

Это блокирующая форма сетевого цикла, которая не будет возвращаться до тех пор, пока клиент не вызовет disconnect(). Переподключение при необходимости произойдет автоматически.

Далее будут описаны вызовы из скрипта «commandscript1.py»

**Синтаксис вызова:**

vehicle.mode

**Описание:**

Смена режима беспилотного аппарата.

**Пример реализации в программе:**

vehicle.mode = VehicleMode("GUIDED")

**Синтаксис вызова:**

vehicle.armed

**Описание:**

Изменение режима готовности. Реализован в библиотеке dronekit.

**Пример реализации в программе:**

vehicle.mode = VehicleMode("GUIDED")

**Синтаксис вызова:**

vehicle.simple\_takeoff()

**Описание:**

Взлет на высоту. В параметре указывается высота.

**Пример реализации в программе:**

vehicle.simple\_takeoff(aTargetAltitude)

**Синтаксис вызова:**

vehicle.airspeed

**Описание:**

В параметре указывается скорость, к которой устройство будет стремиться.

**Пример реализации в программе:**

vehicle.airspeed = 3

**Синтаксис вызова:**

LocationGlobalRelative()

**Описание:**

Команда создания точки в пространстве, где в параметрах задается по порядку:

– Широта

– Долгота

– Высота (в метрах)

**Пример реализации в программе:**

point1 = LocationGlobalRelative(48.531311, 135.055093, 20)

**Синтаксис вызова:**

vehicle.simple\_goto()

**Описание:**

Дает команду устройству переместиться в точку, которая указывается в параметре.

**Пример реализации в программе:**

vehicle.simple\_goto(point1)

**Синтаксис вызова:**

time.sleep()

**Описание:**

Приостанавливает выполнение программы на определенное количество секунд. В параметрах указывается количество секунд. Этот аргумент должен быть либо int, либо float.

**Пример реализации в программе:**

time.sleep(30)

**Синтаксис вызова:**

exit()

**Описание:**

Закрывает программу.

2.5 Вывод по разделу

В данном разделе было рассмотрен общий принцип работы решения, выбран инструментарий и разработан небольшой инструмент для подключения платформы ThingsBoard к микрокомпьютеру, который в свое время должен быть подключен к беспилотному аппарату через MAVProxy.

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

3.1 Назначение программ

Существует два способа подключения: по локальной сети и через интернет. Во втором случае необходимо чтобы доступ к платформе по порту 1883 был доступен из интернета, а программам достаточно иметь только доступ к интернету.

Первая программа состоит из единственного скрипта «mqttsendtel.py», которая отвечает за сбор телеметрии и отправку данных на сервер с установленной платформой ThingsBoard. Вторая программа состоит из двух скриптов: «mqttsendtel.py» и «commandscript1.py», которые раздельно друг от друга не работают и должны располагаться в одной папке. Обе программы работают независимо друг от друга, могут располагаться в разных местах или одна из них может не запускаться вовсе. В случае, если не запускается «mqttsendtel.py», то настоятельно рекомендуется подключить к дрону GCS-клиент для отслеживания состояния устройства.

3.2 Системные требования программ

Совместимый компьютер с ARM или x32 архитектурой на Ubuntu 16.04 или выше.

3.3 Первичная настройка платформы

3.3.1 Особенности установки платформы ThingsBoard

Разработчики платформы создали удобную и доступную документацию, с примерами работы скриптов на разных сенсорах и устройствах, но она редко обновляется. Далее будут описаны особенности и проблемы, с которыми я столкнулся во время установки платформы.

Windows версия платформы не запустилась, в текстовом документе с журналом событий платформы сообщалось о невозможности создании базы данных. Схожие ошибки появляются на всех вариантах баз данных, предлагаемых для установки.

Docker версия платформы, установленная на Windows, содержит непонятную ошибку, из-за которого спустя случайное время платформа может потерять абсолютно все данные и запуститься со стартовыми демонстрационными аккаунтами.

Linux версия кроме Ubuntu 16.04, может также устанавливаться на Ubuntu 18.04. По неизвестным причинам Oracle JDK 8 недоступен для установки с официального репозитория. А если скачать из другого источника, то установиться только на Ubuntu 16.04.

Кроме базы данных по умолчанию (HSQLDB), предлагается для использования еще две базы: PostgreSQL и Cassandra, но на этих двух базах платформа выдала ошибку в терминале и не запустилась.

3.3.2 Настройка платформы

На платформе ThingsBoard через аккаунт «Администратор владельца» ( в демонстрационных данных этот аккаунт называется «tenant@thingsboard.org», а его пароль «tenant») необходимо создать устройство, как на рисунке 5. Вводить название и тип устройства можно любые.

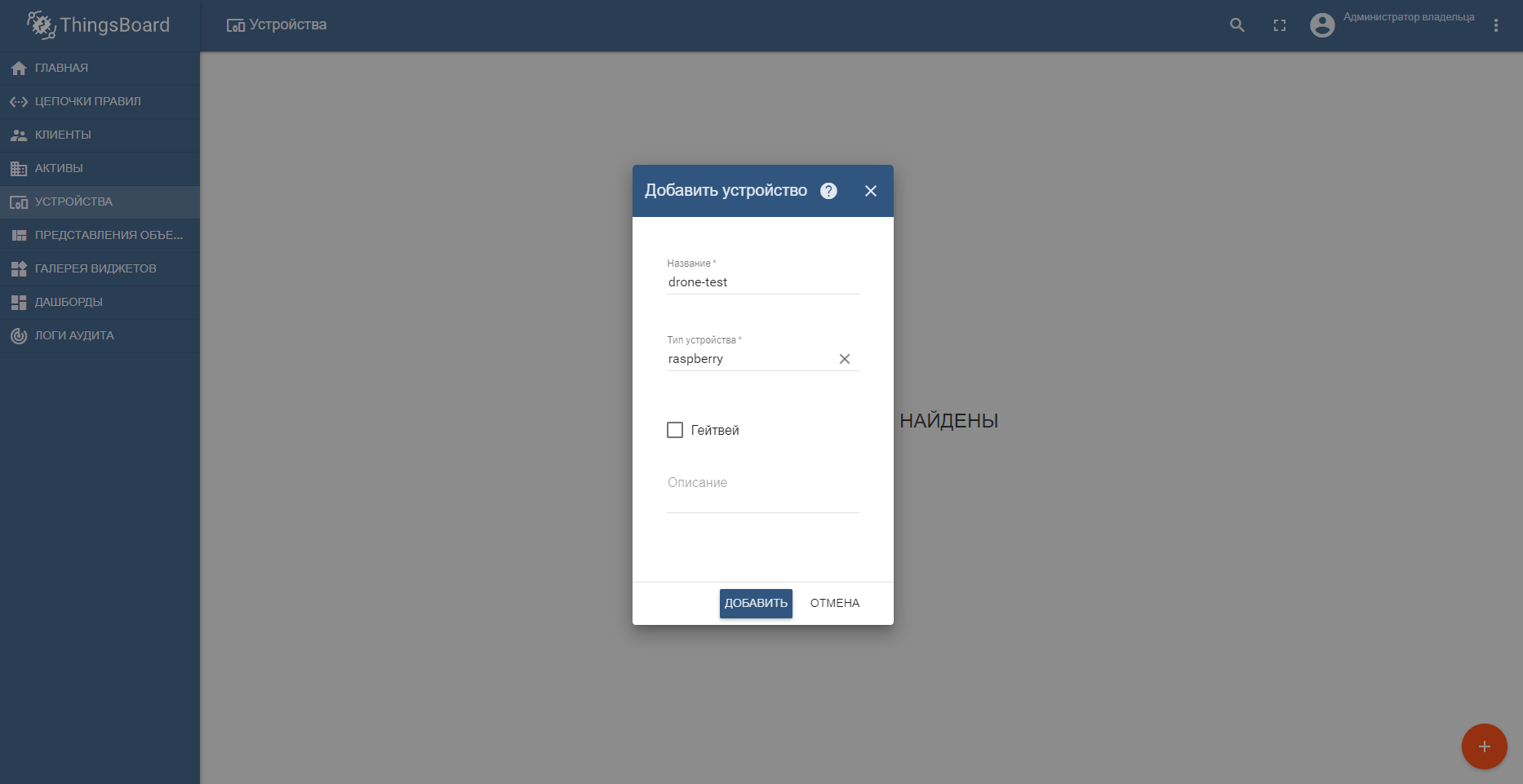


Рисунок 5 – создание устройства на платформе ThingsBoard

Далее необходимо выбрать в категории «тип учетных данных» пункт «Access Token» и ввести любой токен на выбор, как на рисунке 6.

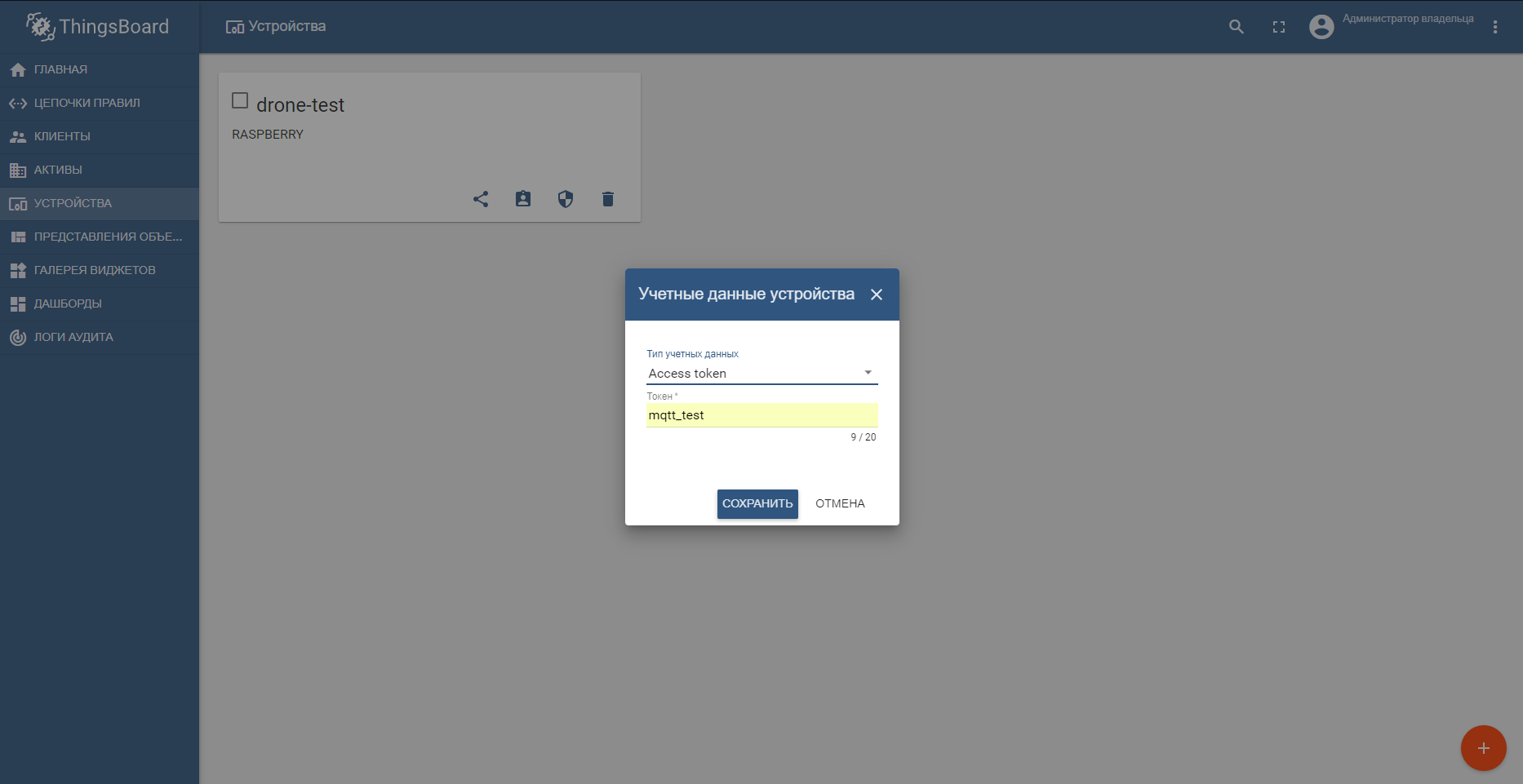


Рисунок 6 – настройка учетный данных

В моем случае далее будет использоваться токен «mqtt\_test»

3.4 Настройка программ

Далее будет рассмотрен пример запуска программ на симуляторе дрона DroneKit-SITL. Если необходимо подключить программы к настоящему беспилотному аппарату, то MAVProxy должен быть подключен к настоящему устройству, а установку и запуск симулятора можно пропустить.

В моем примере запуск программ и симулятора будет производиться на ОС Ubuntu 18.04 Desktop версии

На рисунке 7 изображена общая схема расположения программ в папке, которая будет использоваться в этом руководстве. При смене местоположения программ, команды в терминале необходимо запускать из папки, где они соответственно расположены.

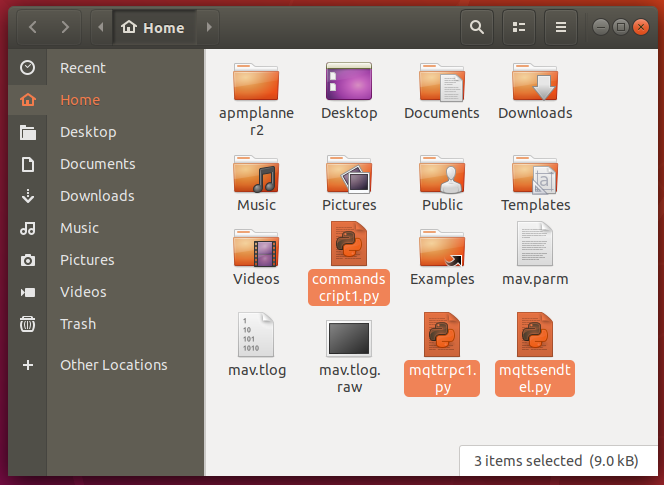


Рисунок 7 – расположение скриптов

Сначала необходимо установить необходимый инструментарий и (в моем случае) симулятор.

Python:

sudo apt install python

Систему управления пакетами Pip, которая используется для установки и управления программными пакетами, написанными на Python:

sudo apt install python-pip

Eclipse Paho MQTT Python библиотеку для клиента:

pip install paho-mqtt

MAVProxy:

pip install mavproxy

DroneKit-Python:

pip install dronekit

И симулятор DroneKit-SITL:

pip install dronekit-sitl

Перезагружаем ОС. Потом необходимо убедиться, что соединение до платформы доступно командой ping. Пример исполнения команды изображен на рисунке 8.

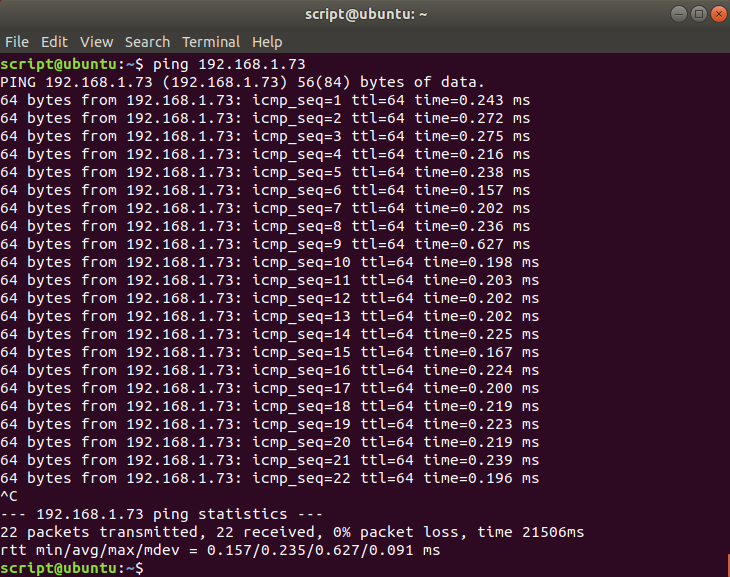


Рисунок 8 – проверка соединения до 192.168.1.73

Запускаем симулятор DroneKit-SITL:

dronekit-sitl copter --home=48.530668,135.054721,0,180

Первые два числа обозначают широту и долготу координаты старта. Процесс запуска симулятора изображен на рисунке 9

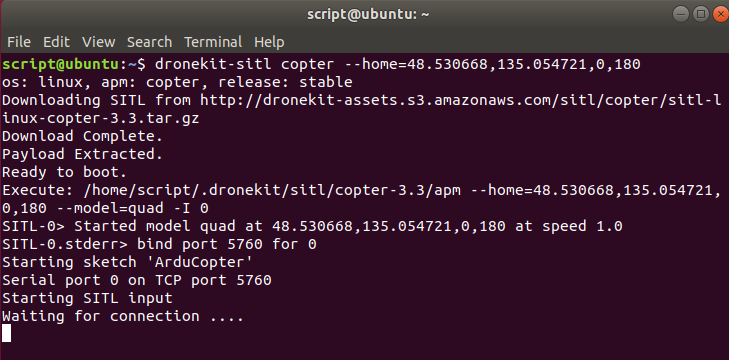


Рисунок 9 – запуск DroneKit-SITL

Далее необходимо в новом терминале запустить MAVProxy:

mavproxy.py --master tcp:127.0.0.1:5760 --out udp:127.0.0.1:14551 --out udp:0.0.0.0:14550 --out udp:127.0.0.1:14549

Параметр master обозначает устройство, а параметры out – возможные методы подключения. Можно вписать ip компьютера, на котором запускается MAVProxy или 0.0.0.0 (порт всегда обязателен), во втором случае ip определиться автоматически. Порты 14551 и 14550 необходимы для запуска скриптов, а 14549 можно поменять или убрать совсем, так как он рассчитан на дополнительный запуск GCS-клмента для полноценного контроллирования дрона. Процесс запуска изображен на рисунке 10, где слева терминал с DroneKit-SITL, а справа с MAVProxy.

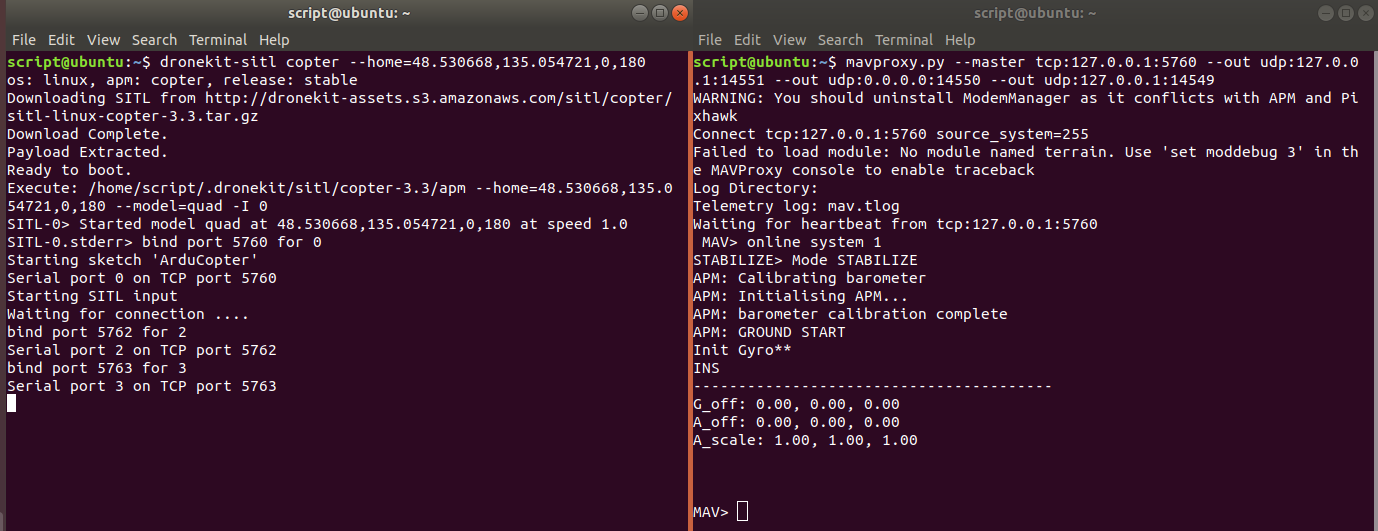


Рисунок 10 – запуск MAVProxy

По одному порту доступно только одно подключение.

Теперь можно производить запуск программ. Скрипты должны запускаться на том же компьютере, где запущен MAVProxy, так как соединение с дроном будет идти через него.

Запуск каждого из скриптов производится в отдельном терминале, так как нельзя закрывать процессы DroneKit-SITL и MAVProxy из уже открытых терминалов.

Перед запуском скриптов («mqttsendtel.py» и «mqttrpc1.py») необходимо изменить их содержимое в каждом из двух.

Необходимо найти строки:

THINGSBOARD\_HOST =

ACCESS\_TOKEN =

THINGSBOARD\_HOST – это переменная, которая будет содержать в себе ip по которому необходимо подключиться. После знака равно в апострофах необходимо изменить значение на ip платформы ThingsBoard.

ACCESS\_TOKEN – это переменная, которая будет в себе содержать токен, по которому будет определяться устройство. После знака равно в апострофах необходимо изменить значение на токен, который был введен при создании устройства на платформе.

Пример записи:

THINGSBOARD\_HOST = '192.168.1.73'

ACCESS\_TOKEN = 'mqtt\_test'

Запуск программы по отправлению телеметрии:

python mqttsendtel.py --connect 0.0.0.0:14550

Пример выполнения программы показан на рисунке 11.

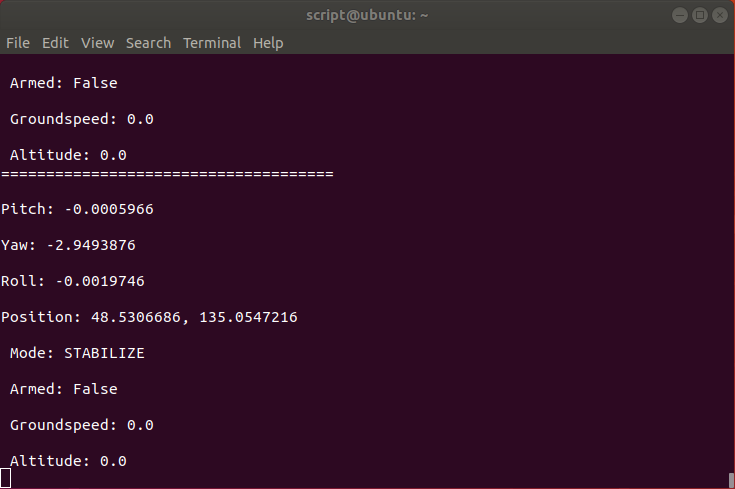


Рисунок 11 – работа скрипта «mqttsendtel.py»

Запуск программы по приему и обработки запросов с платформы:

sudo python mqttrpc1.py

Пример выполнения программы в самом начале после запуска показан на рисунке 12.

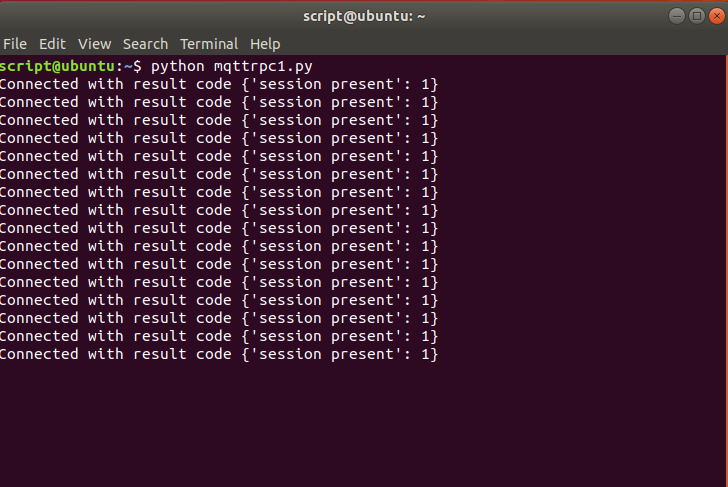


Рисунок 12 – работа скрипта mqttrpc1.py

Стоит обратить внимание, что, если что-то начнет подключение к MAVProxy по портам 14551 или 14550, работа программ остановится и их будет необходимо перезапустить.

3.5 Вторичная настройка платформы

Необходимо создать дашборд с виджетами, которые будут визуализировать телеметрию. Только потом можно будет добавить переключатели для отправления запроса на микрокомпьютер с запущенными скриптами, так как иначе устройство может не появится в списке выбора. В качестве примера далее описано создание карты.

В списках полученной телеметрии необходимо выбрать долготу и широту, как указано на рисунке 13, а потом нажать «Показать на виджете».

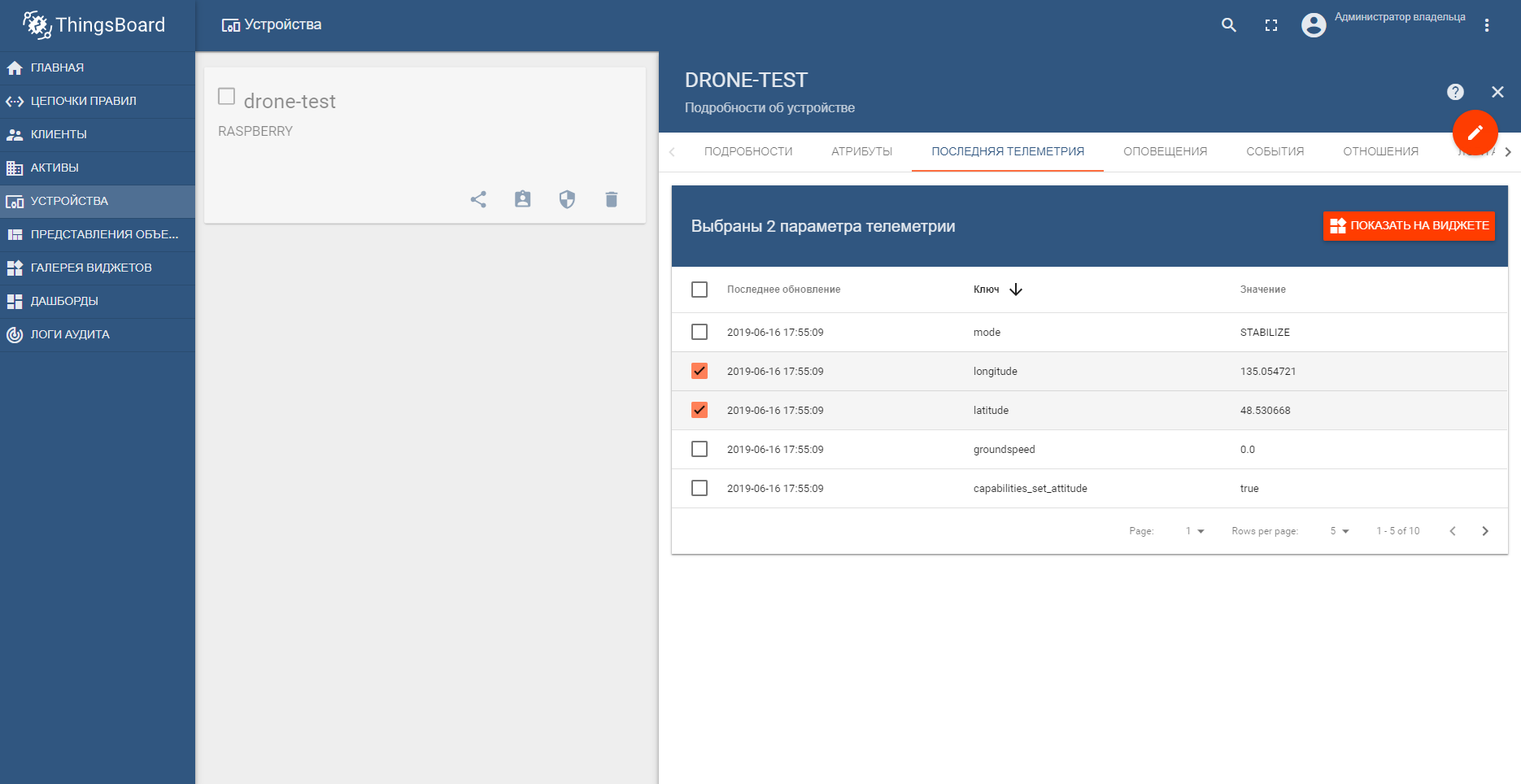


Рисунок 13 – список полученной телеметрии от дрона

Из набора «maps» выбирается любой доступный виджет с картой. Далее предлагается создать новый дашборд. Ввести название, активировать галочку «открыть дашборд» и нажать «Добавить».

Далее в окне созданного дашборда нужно войти в режим редактирования и нажать «добавить новый виджет», а после «создать новый виджет». Из списка наборов в категории GPIO Widget необходимо выбрать управляющий виджет «BASIC GPIO Control», который представлен на рисунке 14.

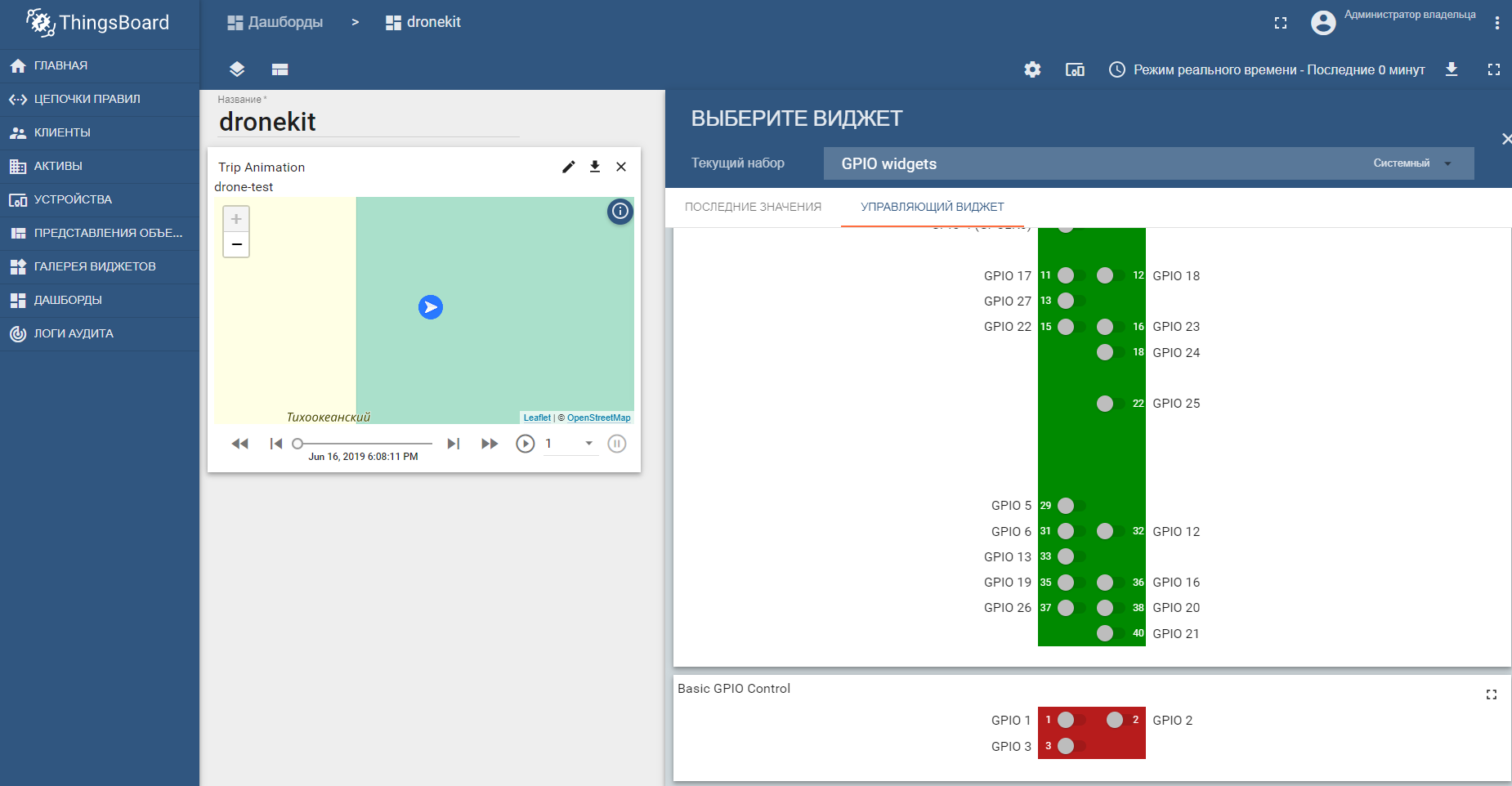


Рисунок 14 – набор управляющих виджетов.

В нем необходимо выбрать созданное на платформе устройство и настроить три переключателя (уже предлагается по умолчанию). По желанию можно поменять имена кнопок или изменить параметр RPC request timeout на значение больше 500, если программы были запущены на низкопроизводительном микрокомпьютере, так как если платформа не дождется ответа, то она может повторно отправить сообщение с данными, что может неправильно сказаться на работе программы.

По аналогии создаются виджеты для оставшейся телеметрии.

3.6 Тестирование решения

Платформа ThingsBoard была запущена на стационарном ПК со следующими характеристиками:

– Процессор AMD Ryzen 5 2600, 6 ядер, 16 МБ кэш-памяти на частоте 3900 МГц

– Оперативная память 16гб DDR4, на частоте 2660 МГц

– Графический процессор NVIDIA GTX 1080, 8192 МБ GDDR5X памяти

* Твердотельный накопитель 120 ГБ, интерфейс SATA 6Gb/s

Программы и симулятор запускались на ноутбуке со следующими характеристиками:

– Процессор Intel Core i5-8250U, 4 ядра, 6 МБ кэш-памяти на частоте 1600-3400 МГц

– Оперативная память 8 ГБ DDR4, на частоте 2400 МГц

– Графический процессор NVIDIA GeForce MX150, 2 ГБ GDDR5 памяти

– Твердотельный накопитель 256 ГБ, интерфейс PCIe 3.0 x4

На виджете управления первый переключатель отвечает за запуск скрипта «commandscript1.py» на микрокомпьютере, который отдаст команды дрону переместиться по двум точкам с задержкой в 30 секунд для каждой команды и выводит сообщение «command 1» в терминале с запущенным «mqttrpc1.py». Выполнение команды изображено на рисунке 15.

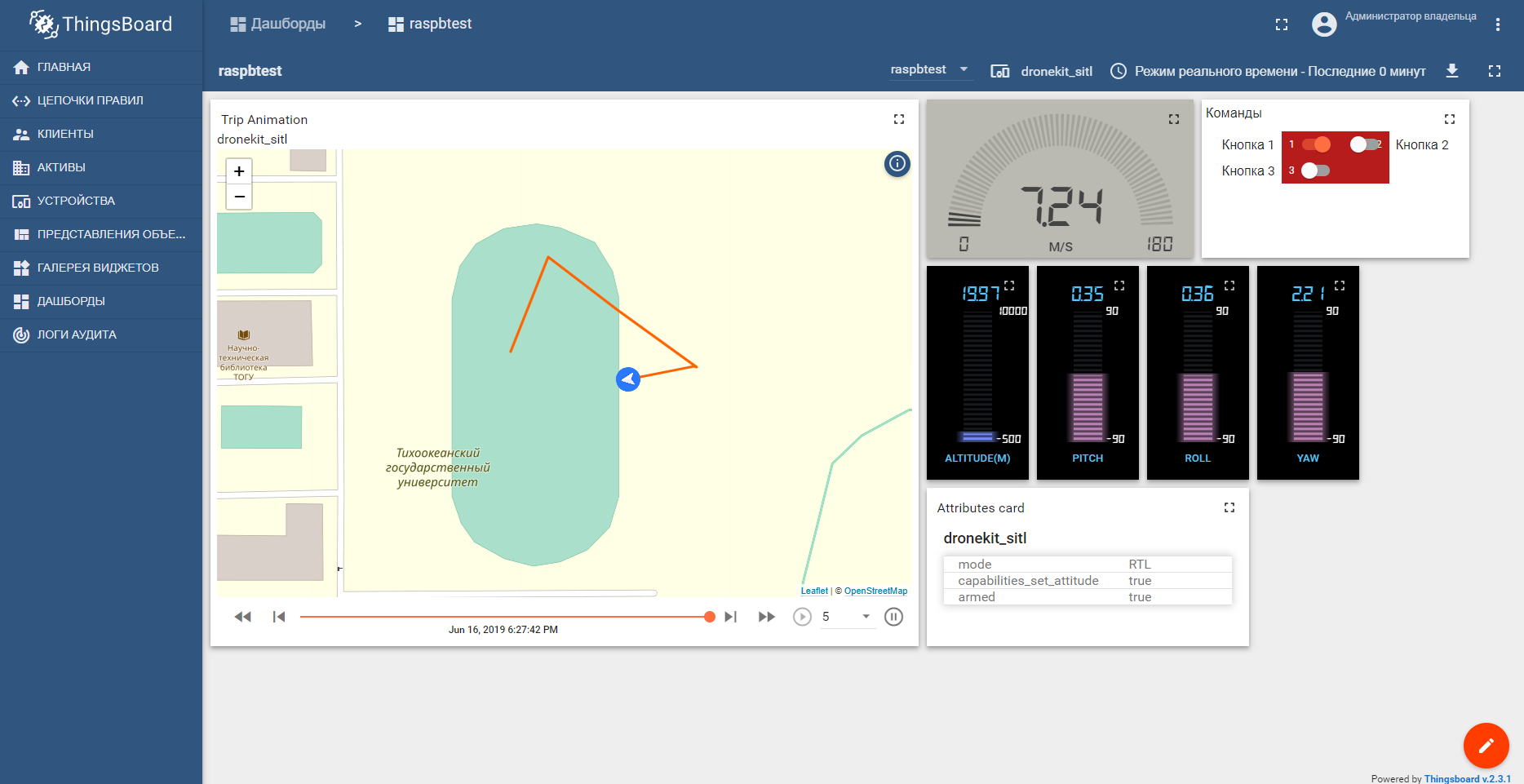


Рисунок 15 – пример выполнения запроса с платформы

Второй переключатель выводит сообщение «command 2». Его можно использовать для проверки соединения между платформой и микрокомпьютером. Третий переключатель перезагрузит микрокомпьютер. В этом случае рекомендуется добавить скрипт в автозапуск операционной системы для автоматического восстановления соединения. Выполнение переключателей работает только при переключении вправо.

ЗАКЛЮЧЕНИЕ

Целью данной дипломной работы была разработка решения для управления малыми беспилотными аппаратами с web-платформы по протоколу MQTT.

Автором данной работы был проведен краткий анализ концепции «Интернет вещей». Был разобран список существующих популярный IoT-платформ для подключения различных устройств. Была разработана структура управления дроном с web-платформы ThingsBoard. Исходный код платформы и используемого инструментария является открытым, что позволяет разработчикам создавать свои решения для использования различных беспилотных аппаратов. Для данной работы в дальнейшем возможно следующее:

– Повышение стабильности в работе скриптов

– Расширение функционала по управлению беспилотными аппаратами с платформы

– Создание заготовок скриптов для различных задач

– Автоматическая установка необходимых библиотек и настройка программ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интернет вещей [Электронный ресурс] / многоязычная энциклопедия// Википедия – проект свободной многоязычной энциклопедии. – Режим доступа: https://ru.wikipedia.org/wiki/Интернет\_вещей, свободный (дата обращения 15.05.18)
2. Internet of things [Электронный ресурс] / многоязычная энциклопедия// Википедия – проект свободной многоязычной энциклопедии. – Режим доступа: https://en.wikipedia.org/wiki/Internet\_of\_things, свободный (дата обращения 15.05.18)
3. Frequently Asked Questions [Электронный ресурс] / документация разработчиков// Официальный сайт протокола MQTT. – Режим доступа: http://mqtt.org, свободный (дата обращения 15.05.18)
4. The Simple Text Oriented Messaging Protocol [Электронный ресурс] / документация разработчиков// Официальный сайт протокола STOMP. – Режим доступа: https://stomp.github.io, свободный (дата обращения 15.05.18)
5. Constrained Application Protocol [Электронный ресурс] / многоязычная энциклопедия// Википедия – проект свободной многоязычной энциклопедии. – Режим доступа: https://en.wikipedia.org/wiki/Constrained\_Application\_Protocol, свободный (дата обращения 15.05.18)
6. Open source IoT Platform Edge computing and Consulting services [Электронный ресурс] / документация разработчиков// Официальный сайт IoT-платформы MAINFLUX. – Режим доступа: https://www.mainflux.com, свободный (дата обращения 15.05.18)
7. Connect and Manage your Internet of Things products within minutes [Электронный ресурс] / документация разработчиков// Официальный сайт IoT-платформы thinger.io. – Режим доступа: https://thinger.io, свободный (дата обращения 15.05.18)
8. Everything an IoT platform should stand for [Электронный ресурс] / документация разработчиков// Официальный сайт IoT-платформы Kaa. – Режим доступа: https://www.kaaproject.org, свободный (дата обращения 15.05.18)
9. Open Source Drone Software Projects [Электронный ресурс] / энциклопедия// DRONE DOJO. – Режим доступа: https://dojofordrones.com/open-source-drone/, свободный (дата обращения 15.05.18)
10. MAVLink [Электронный ресурс] / энциклопедия// RCSearch. – Режим доступа: http://rcsearch.ru/wiki/MAVLink, свободный (дата обращения 15.05.18)
11. DroneKit-SITL[Электронный ресурс] / документация разработчиков//Инструкция для DroneKit-SITL. – Режим доступа: https://github.com/dronekit/dronekit-sitl, свободный (дата обращения 15.05.18)