МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ** **ПОЛИТЕХНИЧЕСКИЙ** **УНИВЕРСИТЕТ»** **(МОСКОВСКИЙ** **ПОЛИТЕХ)**

Факультет информационных технологий

Кафедра «Прикладная информатика»

Форма обучения: очная

**ОТЧЕТ**

**о** **выполнении** **лабораторной** **работы** **№2**

**по** **дисциплине**

**«Платформы интернета вещей»**

Студент

(личная подпись)

(И.И. Вобленко)

(И.О. Фамилия)

**Москва** **2022**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc100839465)

[1 Структура MQTT сообщений 6](#_Toc100839466)

[2 Архитектура издателя-подписчика MQTT. Основные типы сообщений 9](#_Toc100839467)

[3 Анализ программных продуктов разных производителей, предоставляющих сервисы MQTT-брокеров 11](#_Toc100839468)

[4 Mosquitto server 13](#_Toc100839469)

[5 Работа с MQTT сообщениями 15](#_Toc100839470)

[ЗАКЛЮЧЕНИЕ 22](#_Toc100839471)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc100839472)

# ВВЕДЕНИЕ

В данной лабораторной работе рассмотрено понятие «Платформы интернета вещей».

Актуальность данной работы заключается в том, что системы интернета вещей используются практически повсеместно от банального автоматизированного склада до умных городов. Чтобы поддерживать эти системы в рабочем состоянии и иметь возможность управлять ими крайне необходимо уметь пользоваться различными платформами интернета вещей, а также понимать их достоинства и недостатки [1].

Целью работы является изучение и освоение платформ интернета вещей.

Объектом исследования является платформа интернат вещей ThingsBoard.

Предметом исследования являются технологии платформ интернета вещей для сбора, обработки и визуализации данных, а также управления устройствами.

Платформа интернета вещей (далее IoT) – это комплекс программ, которые применяются для подключения различных интернет вещей к облачной инфраструктуре хранения информации и предоставления удаленного доступа к ним.

На данный момент наиболее востребованными программными платформами для интернета вещей считаются:

* облачная платформа Azure от компании Microsoft;
* коммерческое публичное облако Web Services от компании Amazon;
* Cloud Platform от компании Google;
* платформа для промышленных инноваций ThingWorx;
* суперкомпьютер Watson от компании IBM и др.

Программные IoT платформы отличаются по следующим техническим характеристикам:

* Масштабируемость – максимально возможное число интернет вещей, подключенных к платформе.
* Простота применения – гибкость программного интерфейса приложения и простота работы с исходником.
* Способы развертывания – частное или публичное облачное хранилище.
* Безопасность – обеспечение защиты информации посредством шифрования, мониторинга доступа и т.д.
* База данных – организационная структура хранения информации, которая передается с устройств.

В процессе использования IoT платформ применяется множество протоколов. Самыми востребованными вариантами считаются Constrained Application Protocol (CoAP), Message Queue Telemetry Transport (MQTT), DirectDraw Surface (DDS), eXtensible Messaging and Presence Protocol (XMPP) и HyperText Transfer Protocol Secure (HTTP/HTTPS) [2].

Задачи, решаемые в процессе выполнения лабораторной работы:

1. Изучение платформы ThingsBoard;
2. Развертывание платформы ThingsBoard;
3. Изучение и создание виртуальных устройств;
4. Изучение и использование механизмов правил;
5. Изучение и использование обработки сигналов тревоги;
6. Изучение и использование системы email оповещений;
7. Изучение методов визуализации данных.

# Платформа ThingsBoard

ThingsBoard — это платформа Интернета вещей с открытым исходным кодом для сбора, обработки, визуализации данных и управления устройствами.

Он обеспечивает подключение устройств по стандартным отраслевым протоколам Интернета вещей - MQTT, CoAP и HTTP, а также поддерживает как облачные, так и локальные развертывания. Things Board сочетает в себе масштабируемость, отказоустойчивость и производительность, так что вы никогда не потеряете свои данные [3].

Платформа позволяет [4]:

* быстро разрабатывать, сопровождать и масштабировать IoT проекты
* интегрироваться с другими сервисами обработки IoT данных (AWS IoT, Apache Spark, SigFox, IBM Watson IoT)
* ретранслировать данные устройств в другие системы
* подключать устройства через стандартные протоколы IoT — MQTT, CoAP и HTTP
* создавать панели мониторинга для визуализации данных

Можно выделить следующие основные особенности платформы ThingsBoard.

Сбор телеметрических данных

Сбор и хранение телеметрических данных надежным способом, выдерживая сетевые и аппаратные сбои. Доступ к собранным данным осуществляется с помощью настраиваемых веб-панелей мониторинга или серверных API.

Многоквартирный дом

Поддержка многопользовательских установок "из коробки". У одного клиента может быть несколько администраторов клиентов и миллионы устройств и клиентов.

Визуализация данных

Предоставляет более 30 настраиваемых виджетов из коробки и возможность создавать свои собственные виджеты с помощью встроенного редактора. Встроенные линейные диаграммы, цифровые и аналоговые датчики, карты и многое другое.

Горизонтальная масштабируемость

Количество поддерживаемых серверных запросов и устройств линейно увеличивается по мере добавления новых серверов things board в режиме кластеризации. Никаких простоев, перезапусков сервера или ошибок приложений.

Механизм правил IoT

Обрабатывайте входящие данные устройства с помощью гибких цепочек правил, основанных на атрибутах объектов или содержимом сообщений. Пересылка данных во внешние системы или запуск аварийных сигналов с использованием пользовательской логики. Настройте сложные цепочки уведомлений о тревогах. Расширяйте функциональность на стороне сервера или управляйте своими устройствами с помощью настраиваемых правил. Определите логику вашего приложения с помощью конструктора цепочки правил drag-n-drop.

Отказоустойчивость

Все вещи на бортовых серверах идентичны. Никаких мастеров-рабочих или горячего ожидания. Сбой узла обнаруживается автоматически. Вышедшие из строя узлы могут быть заменены без простоев. Сохраненные данные реплицируются с использованием надежной базы данных NoSQL.

Управление устройствами

Предоставляет возможность регистрации устройств и управления ими. Позволяет отслеживать атрибуты устройств на стороне клиента и предоставлять их на стороне сервера. Предоставляет API для серверных приложений для отправки команд RPC на устройства и наоборот.

Безопасность

Поддерживает шифрование транспорта как для протоколов MQTT, так и для протоколов HTTP(s). Поддерживает проверку подлинности устройства и управление учетными данными устройства.

Управление активами

Предоставляет возможность регистрировать активы и управлять ими. Позволяет предоставлять атрибуты активов на стороне сервера и отслеживать связанные с ними аварийные сигналы. Возможность построения иерархии сущностей с использованием связей.

Настройка и интеграция

Расширяйте функциональность платформы по умолчанию с помощью настраиваемых цепочек правил, виджетов и транспортных реализаций. В дополнение к поддержке MQTT, CoAP и HTTP пользователи Things Board могут использовать свои собственные транспортные реализации или настраивать поведение существующих протоколов.

Управление аварийными сигналами

Предоставляет возможность создавать и управлять сигналами тревоги, связанными с вашими объектами: устройствами, активами, клиентами и т.д. Позволяет отслеживать сигналы тревоги в режиме реального времени и распространять сигналы тревоги по иерархии связанных объектов. Подавать сигналы тревоги при отключении устройства или событиях бездействия.

100% открытый исходный код

Things Board лицензирована под лицензией Apache License 2.0, поэтому вы можете бесплатно использовать любые ИТ-технологии в своих коммерческих продуктах. Вы даже можете разместить его как решение SaaS или PaaS.

Микросервисы или монолитность

Поддерживает монолитное развертывание для начала работы или небольших сред. Предоставляет возможность обновления до микросервисов для обеспечения высокой доступности и горизонтальной масштабируемости.

SQL, NoSQL и гибридная база данных

Поддерживает различные параметры базы данных и возможность выбирать, где хранить основные объекты и где хранить данные телеметрии [3].

# Развертывание платформы ThingsBoard

Так как системы интернета вещей имеют тенденцию сильно масштабироваться, а также должны быть гибкими в вопросе настройки и интеграции в системах пользователя, то рассмотрим наиболее универсальный способ развертывания платформы ThingsBoard, а именно развертывание посредством Docker-контейнера. Данный способ является системонезависимым (главное, чтобы под операционную систему используемой вычислительной системы была актуальная версия Docker), а также имеет быструю скорость развертывания, после одного раза настройки можно развернуть систему в пару команд.

В первую очередь необходимо установить Docker на вычислительную систему с официального сайта. Docker имеет версии под MacOS, Windows, Linux.

После этого на Docker Hub найдем актуальный контейнер плтаформы ThingsBoard. Таким оказывается контейнер thingsboard/tb-postgres. Далее копируем с сайта Docker Hub команду установки данного контейнера и устанавливаем его на вычислительную машину:

docker pull thingsboard/tb-postgres

При корректной установке контейнера будет получено следующее сообщение:

Digest: sha256:b4b9d2094b80bb05e58e3ac157aeabc253b6aa6c401ba6d7fa338e13462bf19f

Status: Downloaded newer image for thingsboard/tb-postgres:latest

docker.io/thingsboard/tb-postgres:latest

Далее с помощью следующих команд создадим директории для хранения данных и логов, а также для выдачи соответствующих прав доступа к этим директориям:

mkdir -p ~/.mytb-data && sudo chown -R 799:799 ~/.mytb-data

mkdir -p ~/.mytb-logs && sudo chown -R 799:799 ~/.mytb-logs

Проверим, что все установилось запустив контейнер с помощью команды:

docker run -it -p 9090:9090 -p 1883:1883 -p 7070:7070 -p 5683-5688:5683-5688/udp -v ~/.mytb-data:/data -v ~/.mytb-logs:/var/log/thingsboard --name mytb --restart always thingsboard/tb-postgres

После чего перейдем по указанному порту и нашему ip:

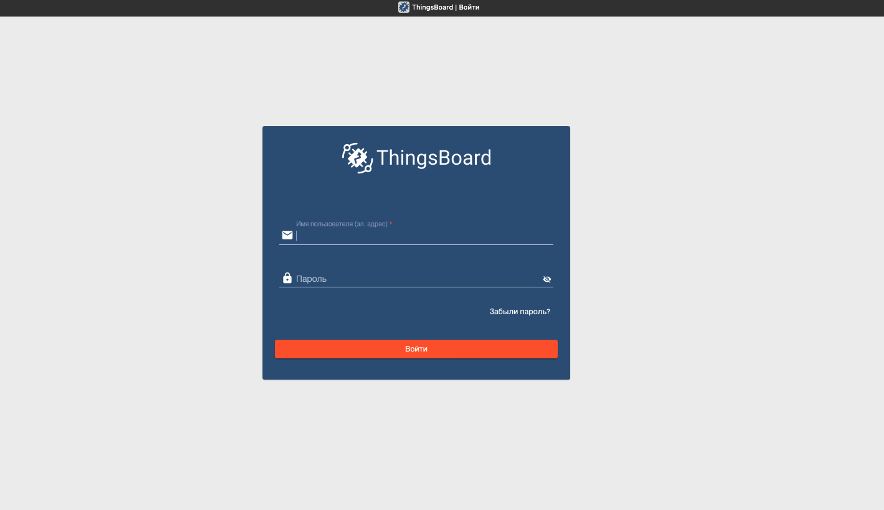


Рисунок 1– Запуск ThingsBoard

# Виртуальные усторйства

Виртуальное устройство — это описание устройства, применяемое для установления соединения между пользователем и физической рабочей станцией, подключенной к удаленной системе. Виртуальные устройства содержат информацию о физическом устройстве (дисплее или принтере), необходимую программам системы [5].

Создадим 3 виртуальных устройства в ThingsBoard. Для этого перейдем в раздел «Устройства» после чего нажмем значок плюса в правом верхнем углу, чтобы добавить новые устройства.

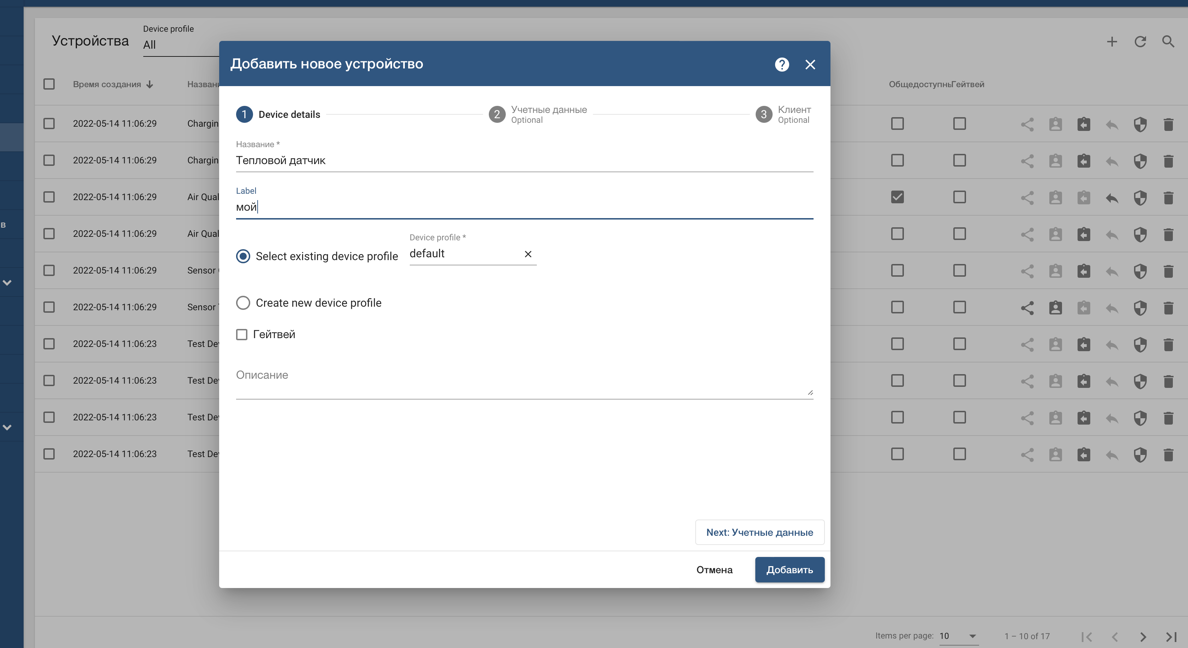
После этого заполним необходимые данные об устройстве.

Рисунок 2 – Добавление нового виртуального устройства

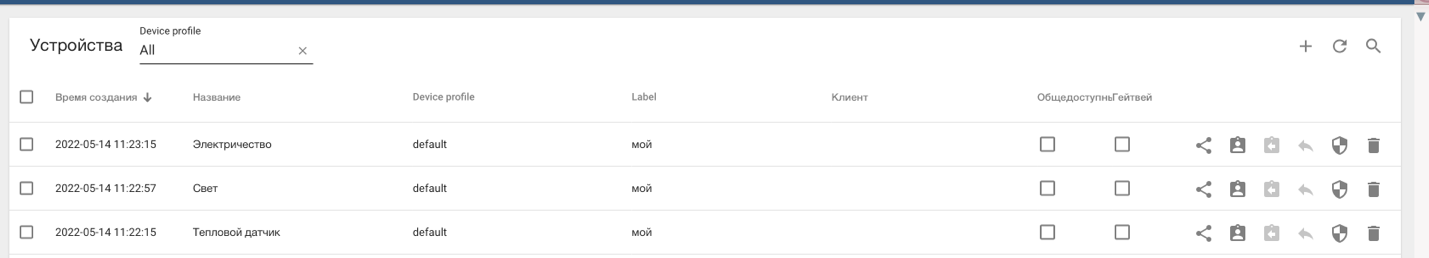


Рисунок 3 – Добавленные устройства

Протестируем возможность получения телеметрии устройств, получим напряжение устройства «Свет». Для этого скопируем уникальный токен (“ij5B5f0mw4m8QAN8EBLe”) данного устройства и воспользуемся следующей командой:

mqtt pub -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u " ij5B5f0mw4m8QAN8EBLe " -m "{'U':145}"

# Mosquitto server

Eclipse Mosquitto - это брокер сообщений с открытым исходным кодом (лицензия EPL / EDL), который реализует протокол MQTT версий 5.0, 3.1.1 и 3.1. Mosquitto имеет небольшой вес и подходит для использования на всех устройствах, от маломощных одноплатных компьютеров до полноценных серверов.

Протокол MQTT предоставляет упрощенный метод обмена сообщениями с использованием модели публикации/подписки. Это делает его подходящим для обмена сообщениями через Интернет вещей, например, с помощью маломощных датчиков или мобильных устройств, таких как телефоны, встроенные компьютеры или микроконтроллеры.

Проект Mosquitto также предоставляет библиотеку C для реализации клиентов MQTT, а также очень популярные клиенты MQTT командной строки mosquitto\_pub и mosquitto\_sub.

Mosquitto является частью Фонда Eclipse Foundation, является iot.eclipse.org проект и спонсируется cedalo.com [5].

Установка и безопасность

Mosquito обладает высокой портативностью и доступен для широкого спектра платформ. Перейдите на специальную страницу загрузки, чтобы найти исходный код или двоичные файлы для вашей платформы.

Ознакомьтесь с списком изменений, чтобы узнать о последних выпусках.

Используйте страницу безопасности, чтобы узнать, как сообщать об уязвимостях или ответах на прошлые проблемы безопасности [5].

Тестирование

Вы можете запустить свой собственный экземпляр Mosquito за считанные минуты, но, чтобы сделать тестирование еще проще, проект Mosquitto запускает тестовый сервер по адресу test.mosquitto.org где вы можете протестировать своих клиентов различными способами: обычный MQTT, MQTT через TLS, MQTT через TLS (с сертификатом клиента), MQTT через WebSockets и MQTT через WebSockets с помощью TLS [5].

Для установки Mosquitto на локальную вычислительную машину необходимо перейти в ссответствующий раздел официального сайта ПО и проследовать описанной там инструкции. Для Windows необходимо скачать установщик, а для Linux и Mac воспользоваться специальными командами, чтобы установить данное ПО.

Воспользуемся следующей комнадой, чтобы поставить Mosquitto на локальную вычислительную машину:

brew install mosquitto

После чего получаем следующее сообщение:

==>Downloading https://ghcr.io/v2/homebrew/core/mosquitto/manifests/2.0.14

Already downloaded: /Users/i.voblenko/Library/Caches/Homebrew/downloads/44b7c5d3b35e8a3232dded872ea031bff5c194f8b3b34dd5d69016abb781342e--mosquitto-2.0.14.bottle\_manifest.json

==>Downloading https://ghcr.io/v2/homebrew/core/mosquitto/blobs/sha256:0e314cf5

Already downloaded: /Users/i.voblenko/Library/Caches/Homebrew/downloads/a9a778877d9da0b6699dff0a6a49a8eb0b526c3e2c48e688898d6f352b5fd43c--mosquitto--2.0.14.monterey.bottle.tar.gz

==> Pouring mosquitto--2.0.14.monterey.bottle.tar.gz

==> Caveats

mosquitto has been installed with a default configuration file.

You can make changes to the configuration by editing:

/usr/local/etc/mosquitto/mosquitto.conf

To restart mosquitto after an upgrade:

brew services restart mosquitto

Or, if you don't want/need a background service you can just run:

/usr/local/opt/mosquitto/sbin/mosquitto -c /usr/local/etc/mosquitto/mosquitto.conf

==> Summary

🍺 /usr/local/Cellar/mosquitto/2.0.14: 43 files, 1.6MB

# Работа с MQTT сообщениями

Запустим ранее установленный mosquito с помощью следующей команды:

brew services start mosquito

После выполнения этой команды в терминале появляются сообщения, подтверждающие запуск программы:

==> **Tapping homebrew/services**

Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services'...

remote: Enumerating objects: 1985, done.

remote: Counting objects: 100% (492/492), done.

remote: Compressing objects: 100% (213/213), done.

remote: Total 1985 (delta 337), reused 353 (delta 271), pack-reused 1493

Receiving objects: 100% (1985/1985), 545.51 KiB | 1.31 MiB/s, done.

Resolving deltas: 100% (892/892), done.

Tapped 1 command (45 files, 697.5KB).

==> **Successfully started `mosquitto` (label: homebrew.mxcl.mosquitto)**

Теперь для проверки корректной установки и работы данного сервреа воспользуемся следующей командой, которая показывает список программ, принимающих соединение и на каком порту они это делают:

lsof -i | grep -E LISTEN

Посмотрев на список активных программ, видимо, что mosquito установлен и работает корректно:

identitys 625 i.voblenko 41u IPv6 0x360f713a22a6a299 0t0 TCP macbook-pro-ila.local:1024->[fe80:11::623d:6cdd:ac83:5556]:1024 (ESTABLISHED)

identitys 625 i.voblenko 48u IPv6 0x360f713a22a6a979 0t0 TCP macbook-pro-ila.local:cap->[fe80:11::623d:6cdd:ac83:5556]:blackjack (ESTABLISHED)

identitys 625 i.voblenko 49u IPv6 0x360f713a22a6a979 0t0 TCP macbook-pro-ila.local:cap->[fe80:11::623d:6cdd:ac83:5556]:blackjack (ESTABLISHED)

cloudd 641 i.voblenko 10u IPv4 0x360f713a3d9f8121 0t0 TCP 172.16.93.160:55378->17.248.150.151:https (ESTABLISHED)

cloudd 641 i.voblenko 11u IPv4 0x360f713a3d673681 0t0 TCP 172.16.93.160:55385->17.248.150.151:https (ESTABLISHED)

Telegram 660 i.voblenko 38u IPv4 0x360f713a36b94131 0t0 TCP 172.16.93.160:55068->149.154.167.41:https (ESTABLISHED)

Telegram 660 i.voblenko 45u IPv4 0x360f713a3cd4abe1 0t0 TCP 172.16.93.160:55401->149.154.167.41:https (ESTABLISHED)

corespeec 800 i.voblenko 4u IPv6 0x360f713a22a709b9 0t0 TCP [fe80:4::aede:48ff:fe00:1122]:51516->[fe80:4::aede:48ff:fe33:4455]:49164 (ESTABLISHED)

PowerChim 4761 i.voblenko 3u IPv6 0x360f713a22a6e759 0t0 TCP [fe80:4::aede:48ff:fe00:1122]:54204->[fe80:4::aede:48ff:fe33:4455]:49175 (ESTABLISHED)

(base) i.voblenko@MacBook-Pro-Ila ~ % lsof -i | grep -E LISTEN

rapportd 621 i.voblenko 3u IPv4 0x360f713a3ac9c661 0t0 TCP \*:51515 (LISTEN)

rapportd 621 i.voblenko 4u IPv6 0x360f713a1f825299 0t0 TCP \*:51515 (LISTEN)

ControlCe 670 i.voblenko 18u IPv4 0x360f713a20d37121 0t0 TCP \*:afs3-fileserver (LISTEN)

ControlCe 670 i.voblenko 19u IPv6 0x360f713a22a6b739 0t0 TCP \*:afs3-fileserver (LISTEN)

ControlCe 670 i.voblenko 20u IPv4 0x360f713a20d39131 0t0 TCP \*:commplex-main (LISTEN)

ControlCe 670 i.voblenko 21u IPv6 0x360f713a22a6be19 0t0 TCP \*:commplex-main (LISTEN)

figma\_age 822 i.voblenko 3u IPv4 0x360f713a20d34661 0t0 TCP localhost:44950 (LISTEN)

figma\_age 822 i.voblenko 4u IPv4 0x360f713a36b90bc1 0t0 TCP localhost:44960 (LISTEN)

figma\_age 822 i.voblenko 11u IPv4 0x360f713a36b90111 0t0 TCP localhost:18412 (LISTEN)

figma\_age 822 i.voblenko 51u IPv4 0x360f713a36b93681 0t0 TCP localhost:7335 (LISTEN)

com.docke 907 i.voblenko 8u IPv4 0x360f713a20d39be1 0t0 TCP localhost:49212 (LISTEN)

com.docke 908 i.voblenko 41u IPv6 0x360f713a22a6fbf9 0t0 TCP \*:https (LISTEN)

com.docke 908 i.voblenko 42u IPv6 0x360f713a22a702d9 0t0 TCP \*:http (LISTEN)

com.docke 908 i.voblenko 43u IPv6 0x360f713a22a6c4f9 0t0 TCP \*:amqp (LISTEN)

com.docke 908 i.voblenko 44u IPv6 0x360f713a22a6cbd9 0t0 TCP \*:15672 (LISTEN)

com.docke 908 i.voblenko 45u IPv6 0x360f713a22a6d2b9 0t0 TCP \*:6379 (LISTEN)

com.docke 908 i.voblenko 47u IPv6 0x360f713a22a6d999 0t0 TCP \*:mysql (LISTEN)

com.docke 908 i.voblenko 48u IPv6 0x360f713a22a6e079 0t0 TCP \*:6080 (LISTEN)

mosquitto 8730 i.voblenko 4u IPv4 0x360f713a3d9dc691 0t0 TCP localhost:ibm-mqisdp (LISTEN)

mosquitto 8730 i.voblenko 5u IPv6 0x360f713a22a6ee39 0t0 TCP localhost:ibm-mqisdp (LISTEN)

С помощью команды следующей комнады создадим пароль для временного пользователя:

mosquito\_passwd -c /etc/mosquito/passwd temp

После чего воспользуемся следующей командой, чтобы посмотреть на созданного пользоватлея:

more passwd

После чего получим:

temp:$777c534fd04b+2cc000819eaf0a63bfa135a62b42777ea4650c2743ca297b3ac6d33c001c664485c7cb3cd3a08475cd80+c434be670c01f16d61218f7f9fe0bde5==

Далее необходимо изменить конфигурационный файл, чтобы пароль начал работать.

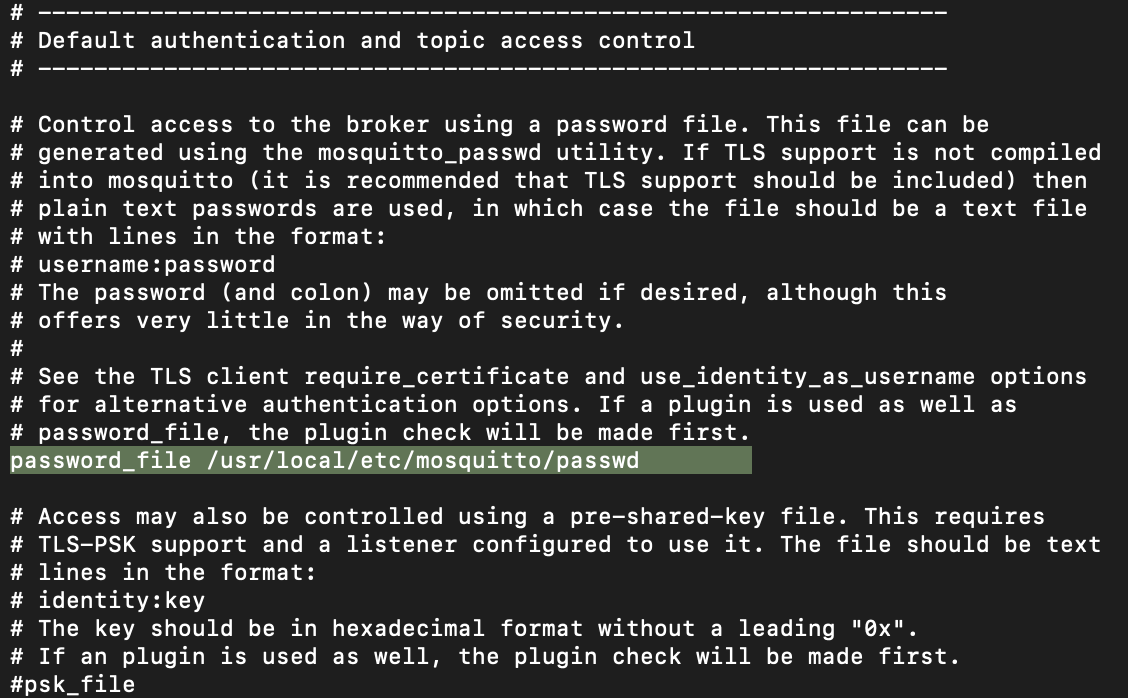


Рисунок 7 – Изменение конфигурационного файла

Чтобы настроить подключение по SSL, необходимо воспользоваться утилитой openssl. Сгенерируем сертификаты с помощью следующей команды:

openssl req -x509 -newkey rsa:4096 -keyout key\_reg.pem -out cert\_reg.pem -nodes -days 365 -subj '/CN=localhost'

Получим следующее подвтерждение выполнения команды:

Generating a RSA private key

......................................................................++++

..++++

writing new private key to 'key\_reg.pem'

После этого сгенерируем сертификаты для устройства:

openssl req -x509 -newkey rsa:4096 -keyout key\_dev.pem -out cert\_dev.pem -nodes -days 365 -subj '/CN=localhost'

Generating a RSA private key

.................................++++

...............++++

writing new private key to 'key\_dev.pem'

-----

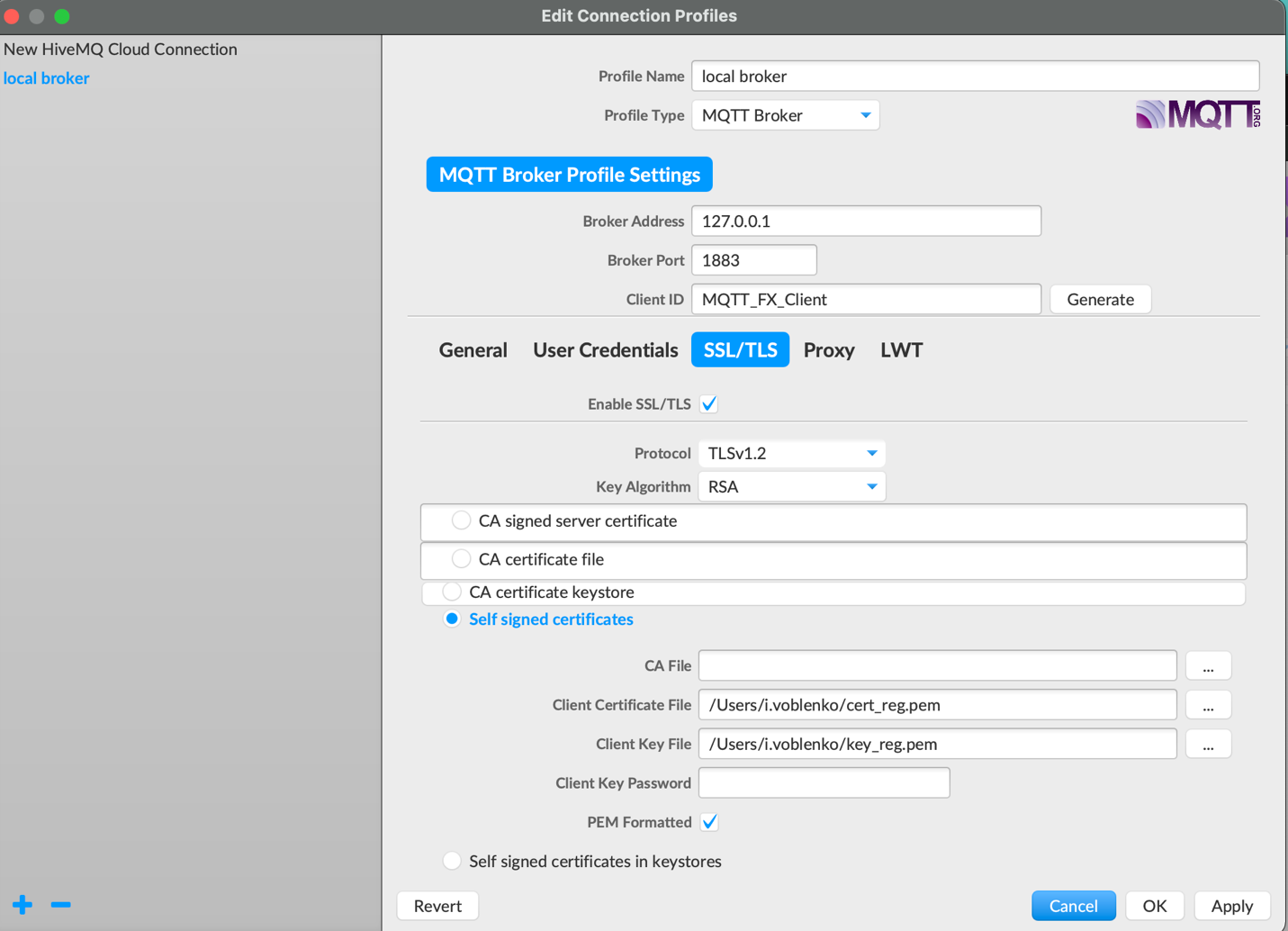
Далее необходимо настроить ssl сертификаты в приложении.

Рисунок 8 – Настройка сертификатов

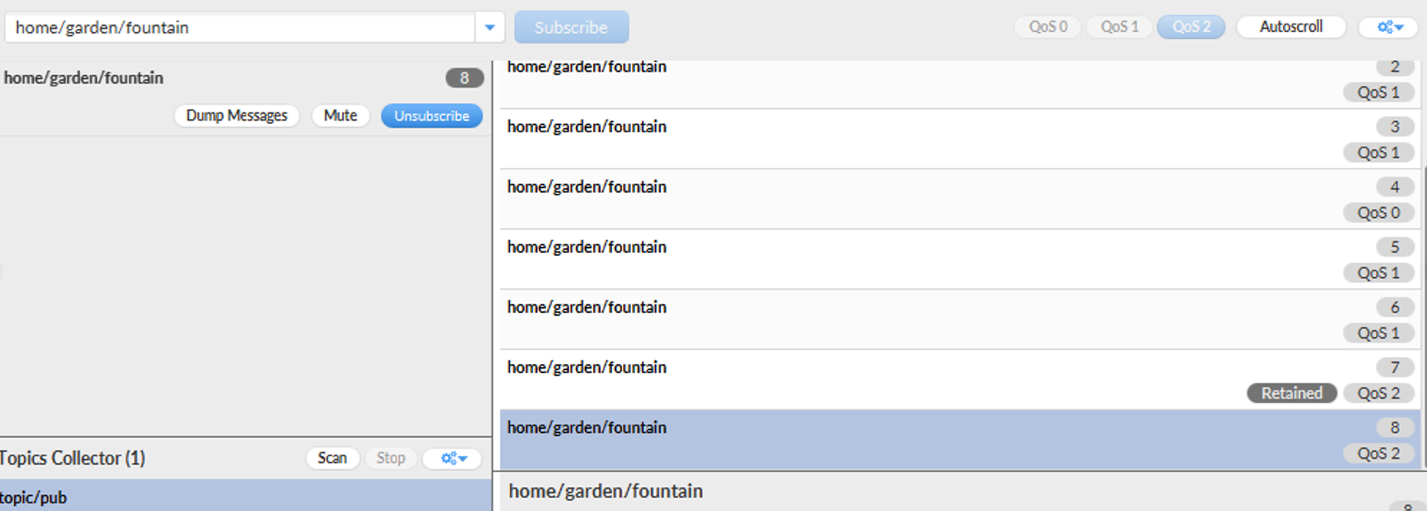
Теперь протестируем использование подстановочных одноуровневых и многоуровневых знаков, позволяющие получать данные сразу с нескольких топиков. Проведем тесты использования различных уровней качества обслуживания при обмене сообщениями брокерных серверов.

Рисунок 9 – Тестирование уровней качества обслуживания

Также протестируем использование сохраняемых сообщений.



Рисунок 10 – Тестирование сохраняемых сообщений

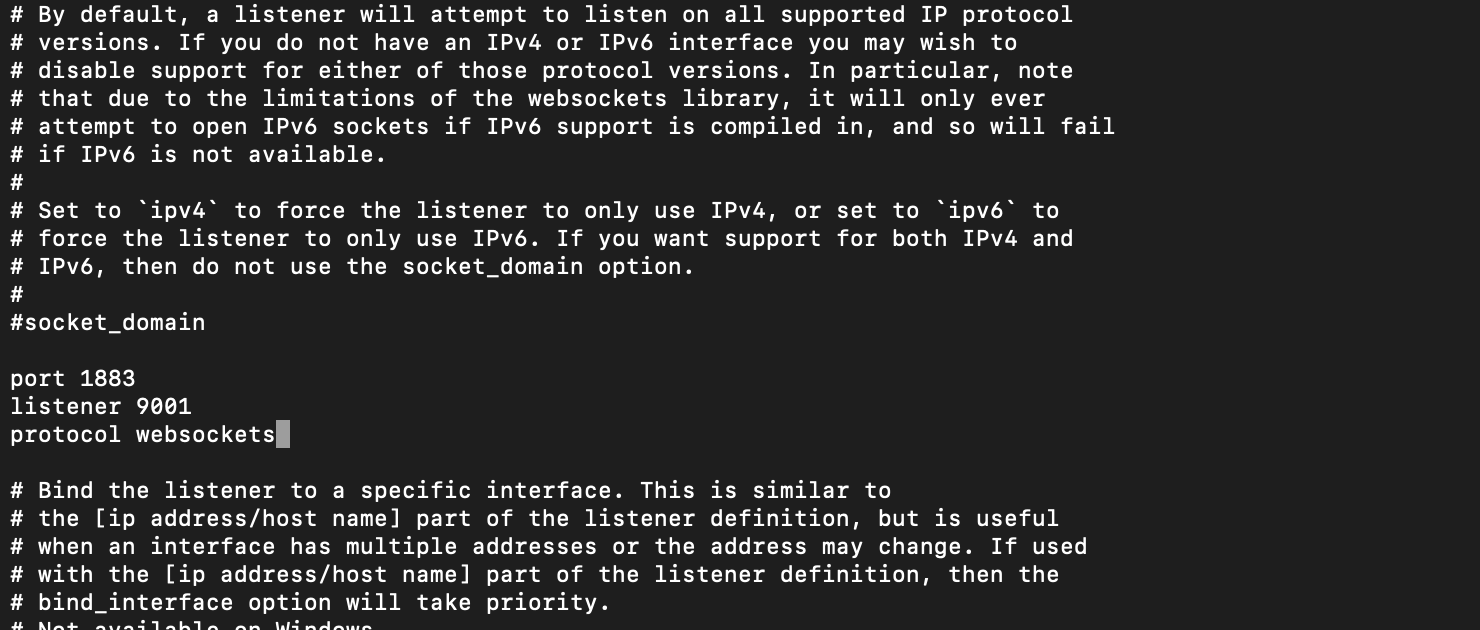
Чтобы отправлять и получать сообщения MQTT в браузере необходимо использовать WebSocket. Чтобы с ним работать нужно в конфигурационном файле добавить работу с ним.

Рисунок 11 – Настройка работы с WebSocket

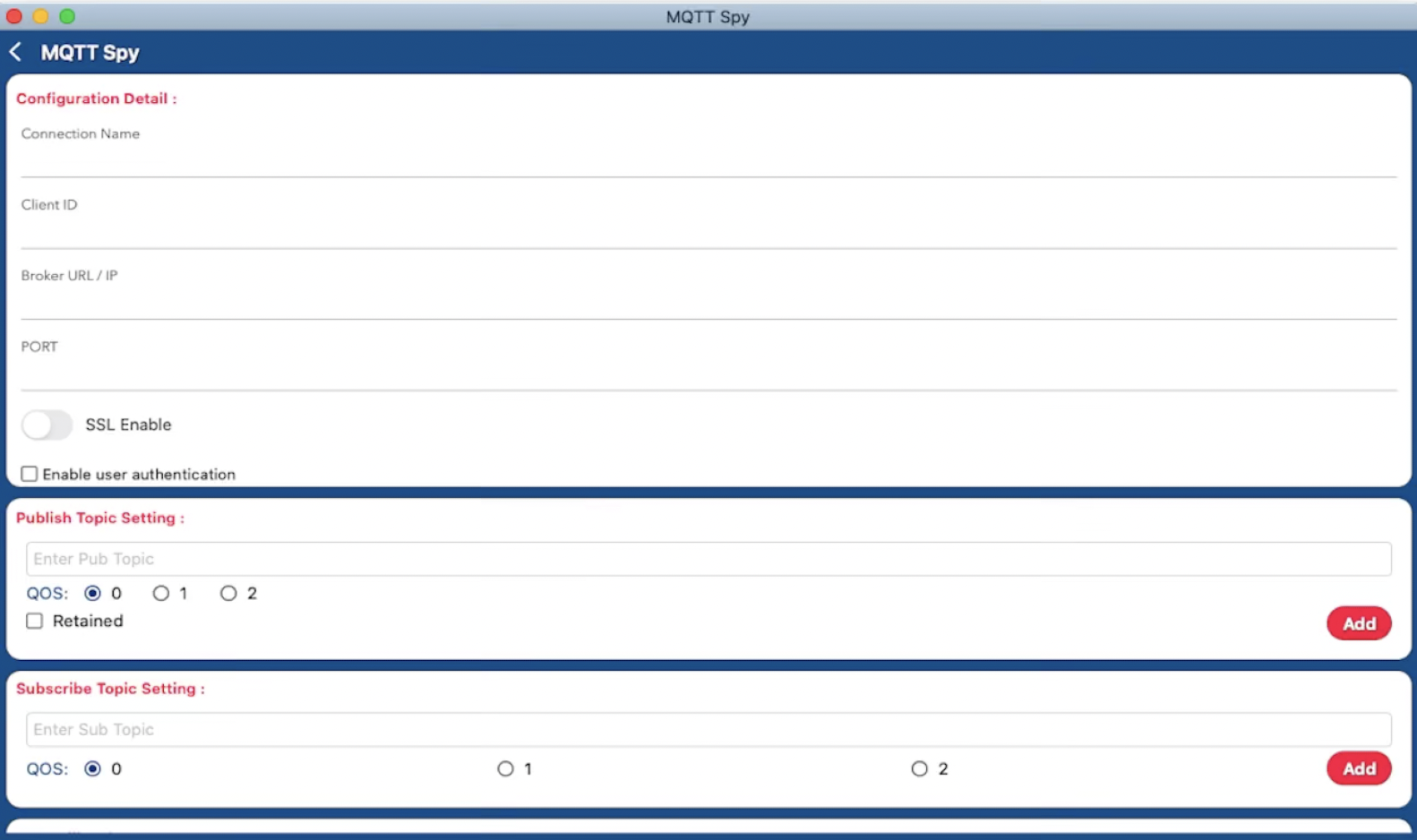
Чтобы была возможность работать с MQTT в графическом представлении, необходимо поставить на локальную машину соответствующий клиент. Для примера воспользуемся MQTT Spy. Перед началом использования необходимо настроить приложение, а именно указать имя хоста, на котором работает Mosquitto.

Рисунок 12 – Настройка MQTT Spy

После этого сделаем тестовое отправление сообщения в соответствующем разделе приложения.

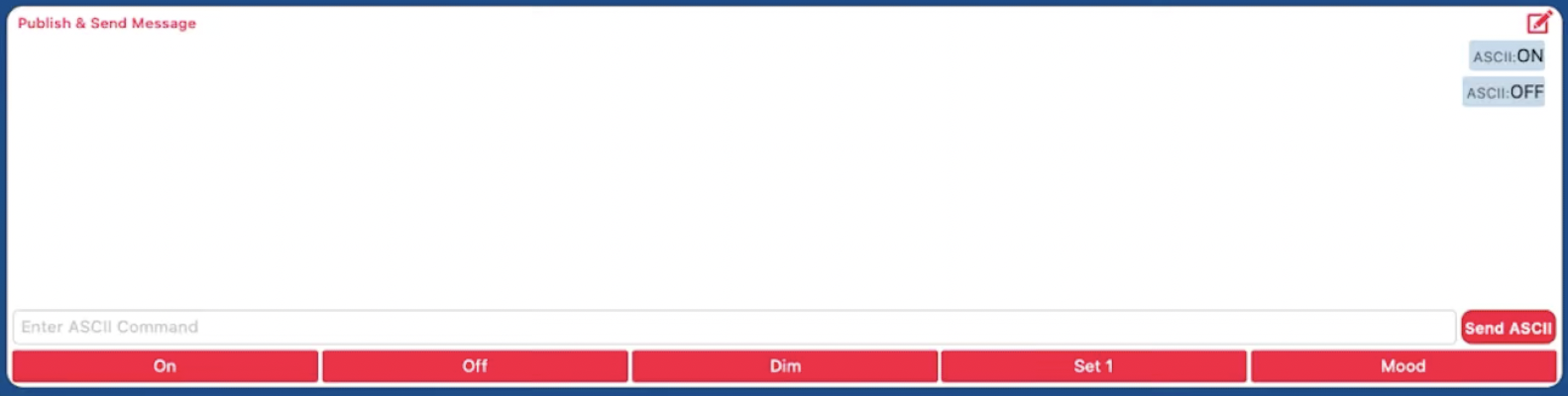


Рисунок 13 – Отправка сообщения

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной лабораторной работы мы познакомились с брокерами сообщений, а также с сетевым протоколом отправки сообщений MQTT. Проанализировали структуру и все особенности данного протокола. Провели сравнительный анализ самых популярных программных продуктов, реализующих работу с протоколом MQTT. Познакомились с Mosquitto server и изучили его основные характеристики. Произвели установку и последующую настройку сервера. Поработали с клиентскими приложениями для взаимодействия с Mosquitto server посредством MQTT протокола.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Центр2М «Воздействие Интернета вещей на различные отрасли» [Электронный ресурс] URL: https://center2m.ru/iot-business-model-zakupok#rec77621887 (дата обращения: 12.05.2022).
2. Intelvision «IoT платформа» [Электронный ресурс] URL: https://www.intelvision.ru/blog/internet-of-things (дата обращения: 12.05.2022)
3. ThingsBoard.io официальный сайт [Электронный ресурс] URL: https://thingsboard.io (дата обращения: 12.05.2022)
4. Habr «Демо доступ к IoT платформам» [Электронный ресурс] URL: https://habr.com/ru/company/euler2012/blog/329614/ (дата обращения: 12.05.2022)
5. IBM «Описание виртуальных устройств» [Электронный ресурс] URL: https://www.ibm.com/docs/ru/i/7.3?topic=server-virtual-device-descriptions (дата обращения 14.05.2022)