

4.1. Лабораторная работа. Знакомимся с протоколом MQTT

Site: [Samsung Innovation Campus](#)

Course: Internet of Things

Book: 4.1. Лабораторная работа. Знакомимся с протоколом MQTT

Printed by: Антон Файтельсон

Date: Saturday, 21 October 2023, 7:37 PM

Table of contents

4.1.1. Что такое MQTT и зачем он нужен?

4.1.2. Установка mosquitto

4.1.3. Hello World в mosquitto

4.1.4. Графические клиенты

4.1.5. Качество обслуживания (QoS)

4.1.1. Что такое MQTT и зачем он нужен?

В настоящий момент MQTT – один из двух самых распространенных протоколов прикладного уровня, используемых в Интернете вещей (второй протокол – это CoAP). Практически все известные облачные платформы IoT так или иначе имеют интерфейс MQTT.

Немного о самом протоколе и его истории. Он был разработан в 1999 году для систем транспортировки нефти: передачи данных от различных сенсоров в SCADA-систему в реальном времени. Его основные характеристики – простота, открытость, легковесность (не загружает канал связи). В те времена стоимость связи была гораздо выше, чем сейчас, и это предопределило особенности протокола MQTT.

Интересно, что изначально название протокола расшифровывалось как Message Queue Telemetry Transport. Однако это вносило путаницу, поскольку в MQTT используется другой паттерн: "Издатель/Подписчик" (Publisher/Subscriber), а не "Очередь сообщений" (Message Queue). Поэтому от расшифровки отказались, и на данный момент MQ в названии официально ничего не значит.

MQTT в иерархии протоколов

Место протокола MQTT в сетевой модели TCP/IP вы можете видеть на схеме:

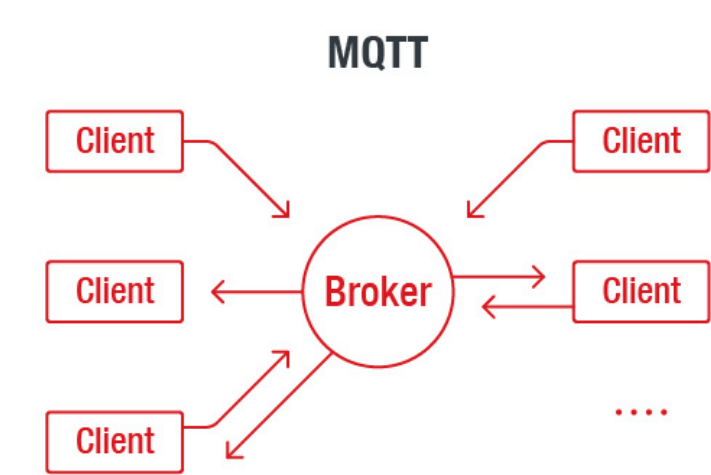
Уровень в модели TCP-IP	В Интернете людей	В Интернете вещей
Прикладной	HTTP, FTP, SMTP, IMAP	MQTT, CoAP
Транспортный	TCP, UDP	UDP, TCP
Межсетевой	IPv4, IPv6	IPv6, IPv4
Уровень доступа к сети	Ethernet, Wi-Fi, GSM	Ethernet, Wi-Fi, GSM, NB-IoT, LoRa, SigFox, 6LoWPAN, ZigBee

Как мы видим, он находится на самом верхнем уровне, прикладном.

Важно, что в своей канонической реализации MQTT ниже уровнем предполагает использование TCP в качестве транспортного протокола. Это делает его несколько тяжелым для использования в сетях низкопотребляющих устройств, где важен каждый переданный байт. Существует реализация MQTT-SN для сенсорных сетей, в которой уровнем ниже находится UDP.

Устройство MQTT

MQTT работает по модели "Издатель/подписчик":



Что делает MQTT интересным для нас протоколом? Если говорить языком книги Эндрю Ханта и Дэвида Томаса "Программист-прагматик", он позволяет нам соблюдать принцип ортогональности, то есть строить программу как набор слабо связанных между собой сущностей.

4.1.2. Установка mosquitto

На этом практикуме предстоит подробно изучить протокол MQTT. Для уверенной работы с ним стоит проделать несколько упражнений.

Сервер Mosquitto - лишь один из возможных вариантов сервера. Можно использовать и другие: HiveMQTT, HBMQTT. Но мы будем использовать Mosquitto, поскольку это самый популярный Open Source-сервер.

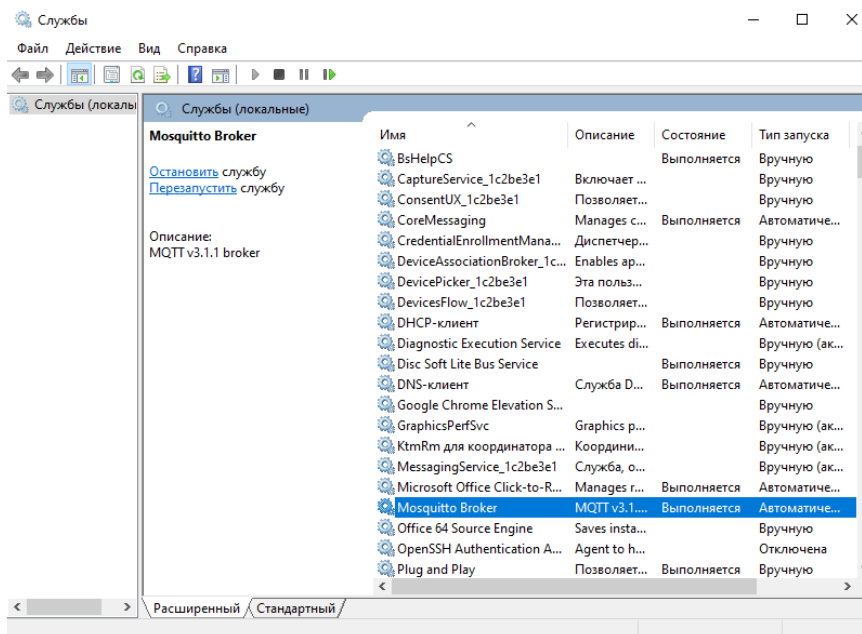
Установите себе на компьютер mosquitto. Это сервер, который можно запустить локально и потренироваться на нём. Установка на Ubuntu делается просто:

```
sudo apt-get install mosquitto
```

Mosquitto также существует в виде пакетов и для других дистрибутивов. Его можно поставить и в embedded-компьютере наподобие Raspberry Pi - это важно, если компьютер функционирует как шлюз между локальной сетью Интернета вещей (LoRa или ZigBee), и собственно Интернетом (WWW).



Можно даже установить mosquitto в Windows, просто скачав дистрибутив с сайта. Тогда он появится в “Службах” Windows, и его можно будет запускать и останавливать из этого окна:



Но всё же настоятельно рекомендуется работать именно в Linux.

Вам предстоит освоить базовый функционал локального MQTT-сервера, чтобы далее выполнять упражнения с ним.

4.1.3. Hello World в mosquitto

1. Запуск сервера

Запустите сервер через терминал:

```
mosquitto
```

Результат:

```
volkova_ta@volkova-ubuntu:~$ mosquitto
1501232660: mosquitto version 1.4.8 (build date Tue, 23 May 2017 22:14:40 +0100)
starting
1501232660: Using default config.
1501232660: Opening ipv4 listen socket on port 1883.
1501232660: Opening ipv6 listen socket on port 1883.
```

Если запустить не получается из-за того, что порт уже используется, выберите другой порт через ключ `-p`.

Вообще обычно по умолчанию сервер стартует на 1883 порту и запускается в режиме демона, то есть без вашего участия. И порт может быть занят просто потому, что mosquitto уже стартовал в вашей системе (можете проверить это, выведя список всех процессов). Но нам сейчас важно потренироваться, поэтому запустим на другом порту для наглядности.

2. Подписка на топик

В другом окне терминала введите команду подписки на топик:

```
mosquitto_sub -h "localhost" -t "mytopic" -q 1
```

В качестве хоста указан localhost (то есть сам компьютер), а в качестве топика — вымышленный mytopic

Результат: программа будет ждать сообщений из этого топика.

```
volkova_ta@volkova-ubuntu:~$ mosquitto_sub -h "localhost" -t "mytopic" -q 1
█
```

Обязательно посмотрите в предыдущее окно, где стартовал сервер: вы увидите, что там добавилась новая информация.

3. Отправка сообщения

Наконец, в третьем окне терминала введите команду, отправляющую сообщение — mosquitto_pub:

```
mosquitto_pub -h "localhost" -t "mytopic" -m "Hello World" -q 1
```

```
volkova_ta@volkova-ubuntu:~$ mosquitto_pub -h "localhost" -t "mytopic" -m "Hello
World" -q 1
```

Параметры следующие:

- h — Адрес сервера
- t — Название топика, в котором публикуется сообщение (в данном примере это mytopic)
- m — Текст сообщения
- q — Качество обслуживания

В результате, вы увидите в окне с подпиской текст «Hello World»:

```
volkova_ta@volkova-ubuntu:~$ mosquitto_sub -h "localhost" -t "mytopic" -q 1
Hello World
```

4. Один сервер, много клиентов

Теперь попробуем общаться по протоколу MQTT в рамках класса, между отдельными машинами. Выберите один компьютер — он будет сервером. Узнайте его IP-адрес (если это Linux, то самый простой способ - `ifconfig`). Запустите на нём сервер MQTT командой mosquitto.

Все остальные могут публиковать свои сообщения командой mosquitto_pub. Не заканчивайте упражнение, пока сообщение каждого из участников группы не достигнет назначения.

При желании можно посмотреть, какие сообщения приходят на сервер, командой:

```
mosquitto_sub -h "192.168.1.15" -t "#"
```

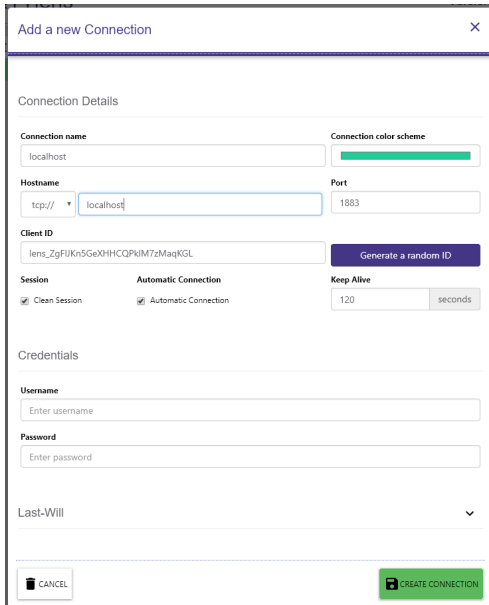
Вы будете получать сообщения из всех топиков (символ # означает все нижестоящие).

4.1.4. Графические клиенты

Расширение для браузера MQTTLens

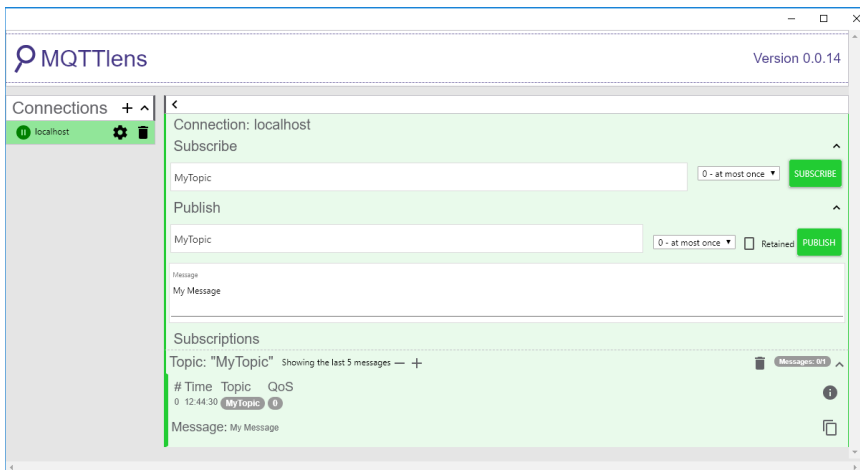
Простой способ проверить работу сервера и увидеть вывод прямо в браузере — установить расширение MQTTLens для браузера Chrome. Установите его на своем компьютере.

Войдите в расширение и добавьте новое соединение. Для подключения к серверу достаточно знать IP-адрес, порт, логин и пароль. Hostname — введите IP-адрес компьютера, где запущен сервер, порт — 1883, и логин с паролем. Хостом может быть как локальный, так и удаленный сервер.



The screenshot shows the 'Add a new Connection' dialog box. It has a title bar with a close button. The main area is titled 'Connection Details'. It contains several input fields: 'Connection name' (set to 'localhost'), 'Connection color scheme' (a green color bar), 'Hostname' (set to 'tcp://localhost'), 'Port' (set to '1883'), 'Client ID' (set to 'lens_ZgFUKnSGeXHCQPkiM7aMaqKGL'), and a 'Generate a random ID' button. There are also checkboxes for 'Clean Session' and 'Automatic Connection', and a 'Keep Alive' field set to '120 seconds'. Below this is a 'Credentials' section with 'Username' and 'Password' fields. At the bottom, there is a 'Last-Will' dropdown and two buttons: 'CANCEL' and 'CREATE CONNECTION'.

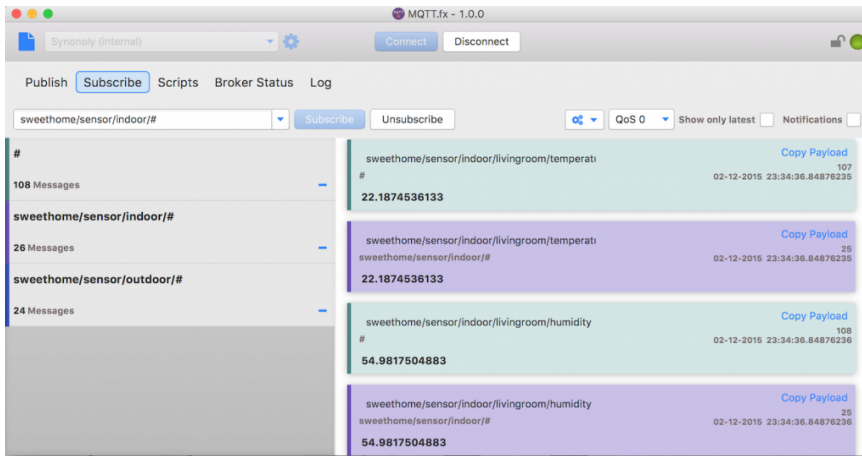
Попробуйте посмотреть в этом расширении вывод сообщений MQTT:



MQTT.fx

Есть программа и с более широким функционалом. Она называется MQTT.fx и в ряде случаев незаменима. Программа написана на Java, её интерфейс довольно приятен.

Ознакомьтесь с ней самостоятельно. Принцип работы схож с MQTTLens. Благодаря большому набору различных опций, эта программа может выручить и помочь найти проблему в сложной ситуации - например, когда не получается подключиться к внешнему MQTT-серверу, используя сторонние библиотеки (о них далее).



4.1.5. Качество обслуживания (QoS)

Чтобы далее успешно работать с MQTT-сервером, нужно знать, по какой схеме происходит коммуникация. Проведите самостоятельные эксперименты. Запустите сервер mosquitto в отладочном режиме:

```
mosquitto -v
```

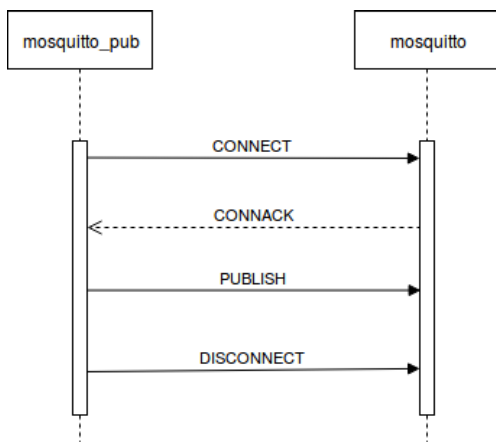
Теперь, проделывая все те же самые действия, что и в прошлом задании, вы увидите большое количество промежуточных шагов:

```
volkova_ta@volkova-ubuntu:~$ mosquitto -v
1501235402: mosquitto version 1.4.8 (build date Tue, 23 May 2017 22:14:40 +0100)
starting
1501235402: Using default config.
1501235402: Opening ipv4 listen socket on port 1883.
1501235402: Opening ipv6 listen socket on port 1883.
1501235404: New connection from ::1 on port 1883.
1501235404: New client connected from ::1 as mosqsub/6807-volkova-ub (c1, k60).
1501235404: Sending CONNACK to mosqsub/6807-volkova-ub (0, 0)
1501235404: Received SUBSCRIBE from mosqsub/6807-volkova-ub
1501235404: mytopic (QoS 0)
1501235404: mosqsub/6807-volkova-ub 0 mytopic
1501235404: Sending SUBACK to mosqsub/6807-volkova-ub
1501235409: New connection from ::1 on port 1883.
1501235409: New client connected from ::1 as mosqpub/6808-volkova-ub (c1, k60).
1501235409: Sending CONNACK to mosqpub/6808-volkova-ub (0, 0)
1501235409: Received PUBLISH from mosqpub/6808-volkova-ub (d0, q2, r0, m1, 'myto
pic', ... (11 bytes))
1501235409: Sending PUBREC to mosqpub/6808-volkova-ub (Mid: 1)
1501235409: Received PUBREL from mosqpub/6808-volkova-ub (Mid: 1)
1501235409: Sending PUBCOMP to mosqpub/6808-volkova-ub (Mid: 1)
1501235409: Sending PUBLISH to mosqsub/6807-volkova-ub (d0, q0, r0, m0, 'mytopic
', ... (11 bytes))
1501235409: Received DISCONNECT from mosqpub/6808-volkova-ub
1501235409: Client mosqpub/6808-volkova-ub disconnected.
```

Кроме того, вы можете увидеть промежуточные сообщения команд mosquitto_sub и mosquitto_pub, запустив их с ключом -d (debug):

```
volkova_ta@volkova-ubuntu:~$ mosquitto_pub -h localhost -p 1884 -t "all" -m "hel
lo" -q 0 -d
Client mosqpub/8060-volkova-ub sending CONNECT
Client mosqpub/8060-volkova-ub received CONNACK
Client mosqpub/8060-volkova-ub sending PUBLISH (d0, q0, r0, m1, 'all', ... (5 by
tes))
Client mosqpub/8060-volkova-ub sending DISCONNECT
```

Для изображения различных протоколов удобно использовать UML-диаграммы последовательности (Sequence Diagram). Вот, например, как выглядит жизненный цикл mosquitto_pub с качеством обслуживания 0, согласно вышеприведенному скриншоту:



Если вам незнакома такая диаграмма, вот короткий обучающий ролик, посвящённый диаграммам последовательности:

Попробуйте отправить сообщения с разным параметром качества обслуживания (Quality of Service). Рассмотрите все случаи: когда QoS равняется 0, 1 или 2.

[Reset user tour on this page](#)