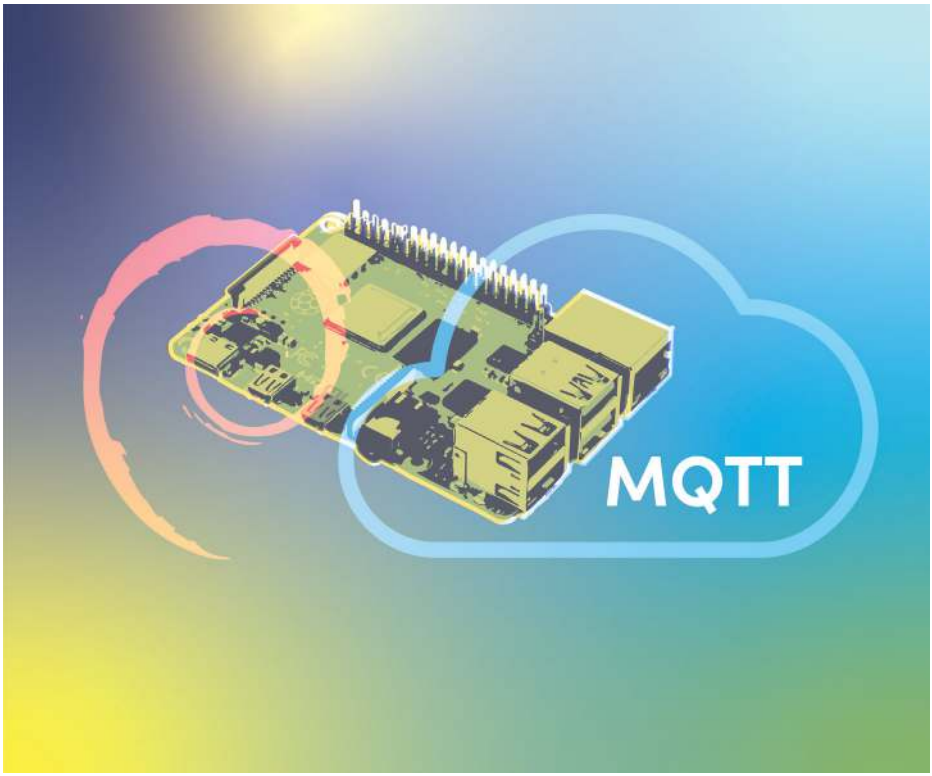
Skoltech

MQTT брокер на базе одноплатного компьютера Raspberry Pi



Учебно-методическое пособие

Сомов А.С., Лыжин И.Г. Методическое пособие «MQTT брокер на базе одноплатного компьютера Raspberry Pi» – М: Сколковский институт науки и технологий, 2020. – 32 с.

УДК 004.7

Учебно-методическое пособие предназначено для всех, кто интересуется интернетом вещей (IoT), в первую очередь - преподавателей и обучающихся средних общеобразовательных школ. В пособии кратко описывается протокол MQTT, приводятся примеры его использования в домашней автоматизации, даются инструкции по установке операционной системы Raspbian, работе с одноплатным компьютером Raspberry Pi, установке и настройке MQTT-брокера Mosquitto, а так же MQTT клиентов для операционных систем Windows и Android. Для успешного освоения материалов, представленных в данном пособии, необходимо быть уверенным пользователем ПК и иметь базовые навыки работы в командной строке Linux/Unix.

Центр компетенций НТИ по направлению «Технологии беспроводной связи и интернета вещей» основан в июне 2018 года на базе Сколковского института науки и технологий для содействия российским коммерческим и государственным компаниям в преодолении технологических барьеров и создании конкурентоспособных продуктов и услуг для мирового рынка в области технологий интернета вещей и беспроводной связи, прежде всего сотовой связи следующих поколений (5G и 6G), индустриального интернета вещей (IIoT) и обработки промышленных данных.

Содержание:

Введение

Глобальное проникновение мобильной связи и Интернета дало возможность любому физическому объекту («вещи») быть в сети и «общаться» с другими вещами. Наверняка Вы слышали про «умные дома», «умные чайники» и даже «умные города», в чем же выражается «ум» вещей? Как раз возможности общаться с другими вещами и «принимать» решения, исходя из данных об окружающем мире. Но для того чтобы вещи могли общаться друг с другом, им нужно общаться на одном «языке» - протоколе. Об одном из протоколов, предназначенном для взаимодействия между машинами (устройствами), и пойдет речь в данном пособии.

Для того, чтобы попробовать на практике выполнить все то, о чем говорится в пособии вам потребуется следующее оборудование:

- Одноплатный ПК Raspberry Pi
- ПК/Ноутбук с OS Windows
- Смартфон с OS Android
- Маршрутизатор

Но если у Вас нет Raspberry Pi, вы можете использовать VPS с установленной OS Debian или же виртуальную машину на ПК/Ноутбуке с Debian (или версией Raspbian для ПК).

Установка и настройка операционной системы Raspbian

У нас есть Raspberry Pi, на который нужно установить операционную систему. Первый вариант подходит, если у пользователя имеется возможность подключить одноплатный компьютер к внешнему монитору, клавиатуре и компьютерной мыши. Если такой возможности нет, можно переходить к варианту 2.

Вариант 1

Необходимо скачать последнюю версию утилиты https://www.raspberrypi.org/downloads/noobs/

Для скачивания доступны два варианта: NOOBS и NOOBS Lite. В первом случае образы различных операционных систем будут загружены сразу. Для использования NOOBS Lite необходимо стабильное подключение к сети интернет, так как необходимые файлы сразу будут загружаться на Raspberry Pi.

Распаковываем содержимое архива на SD-карту.

Вставляем карту в Raspberry Pi, подключаем монитор, клавиатуру и следуем указанием установщика, на вопрос, какую систему устанавливать выбираем - Raspbian.

После установки мы сможем войти в систему: по умолчанию для дистрибутива Raspbian логин: **pi**, а пароль: **raspberry**.

Если установка прошла успешно, вы увидите рабочий стол.

Однако в Raspbian есть особенность - невозможность переключения раскладки клавиатуры сразу после установки. Если настроить переключения раскладки через графический интерфейс, то после перезагрузки проблема возникает вновь.

Для решения этой проблемы надо открыть текстовый редактор:

```
sudo leafpad
```

Системные настройки клавиатуры, хранящиеся в файле `/etc/default/keyboard`, можно исправить через редактор.

Следует заменить строки:

```
XKBLAYOUT="us"
XKBOPTIONS=""
```

На:

```
XKBLAYOUT="us,ru"
XKBOPTIONS="grp:alt_shift_toggle,grp_led:scroll"
```

Далее необходимо добавить в систему графический индикатор раскладки клавиатуры:

```
sudo apt-get install gxkb
```

И поместить его в автозагрузку, открыв файл:

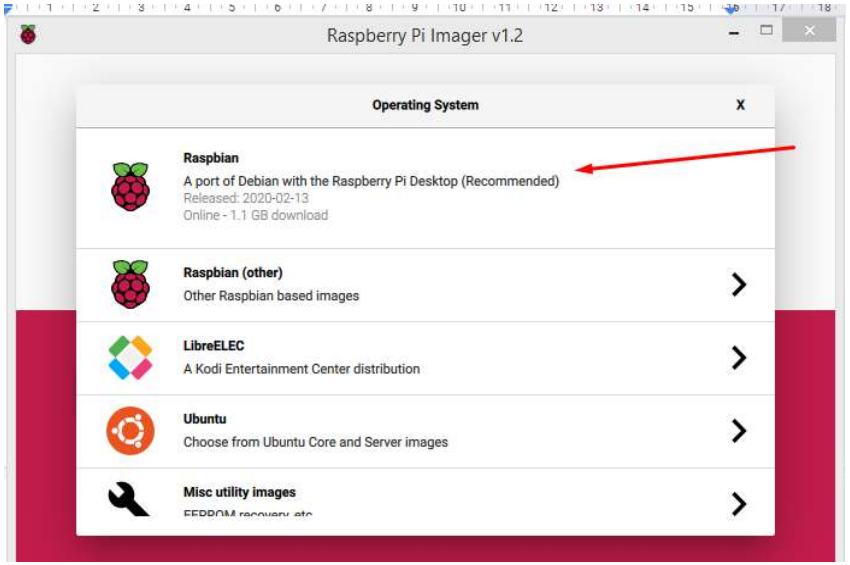
```
sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

где перед строкой `@xscreensaver -no-splash` надо добавить строку `@gxkb`. После этого сохраните файл и выполните перезагрузку командой `sudo reboot`.

Вариант 2

При невозможности подключить Raspberry Pi к монитору, пользователь может зайти на официальную страницу Raspberry Pi https://www.raspberrypi.org/downloads/ и скачать утилиту Raspberry Pi Imager. После этого шага надо вставить чистую карту памяти (micro sd) и запустить Raspberry Pi Imager.

В поле "Operating system" выбираем свежий рекомендованный образ Raspbian. Затем в поле "SD Card" выбираем карту памяти и нажимаем Write.

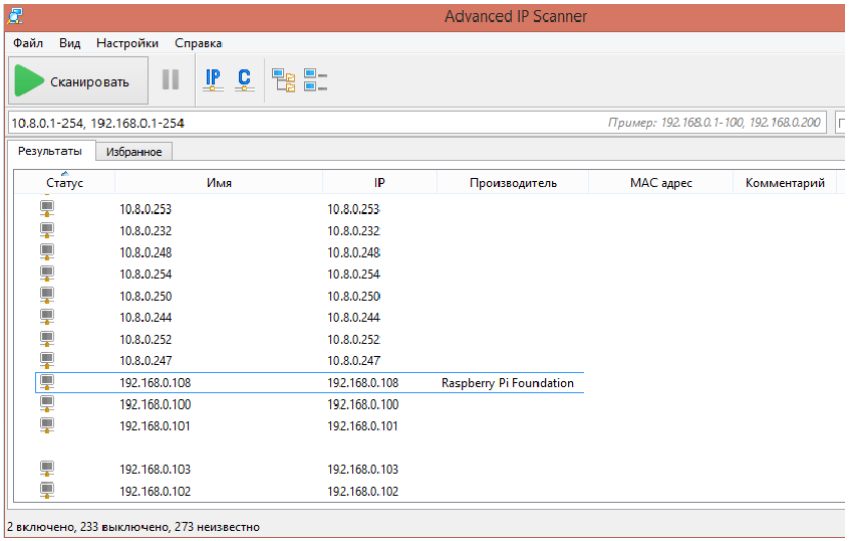


Подключение к Raspberry Pi реализовывается через SSH, но по умолчанию SSH отключен. Для его включения на карте памяти в разделе (папке) boot создаем пустой файл ssh (без расширения). Там же создаем файл с названием wpa_supplicant.conf. Он необходим, чтобы устройство смогло подключиться к wi-fi сети и пользователь получил возможность подключиться к Raspberry Pi при помощи SSH-клиента.

Содержимое файла wpa_supplicant.conf может быть следующим (обязательно впишите имя сети и пароль):

```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1
network={
    ssid="SSID, название точки доступа"
    psk="пароль сети"
    key_mgmt=WPA-PSK
}
```

На этом этапе можно вставить карту памяти и запустить Raspberry Pi. Далее необходимо просканировать сеть, чтобы найти ip-адрес, который получил Raspberry Pi. Для этого воспользуемся утилитой Advanced IP Scanner (https://www.advanced-ip-scanner.com/ru/). Обратите внимание, что компьютер, на котором вы запускаете Advanced IP Scanner, должен быть подключен к той же Wi-Fi сети, что и Raspberry Pi. Запустив утилиту и нажав «Сканировать», через некоторое время вы увидите список устройств, подключенных к сети.



В этом списке надо найти Raspberry Pi. Мы знаем ip адрес одноплатного компьютера и можем подключиться к нему. Для этого можно использовать любой SSH клиент, например, Terminus (https://github.com/Eugeny/terminus).

Запускаем Terminus и переходим в настройки. Затем находим пункт SSH и нажимаем “Add connection”.

Заполняем поля данными: имя подключения может быть произвольным, ip адрес тот, который мы узнали ранее, порт по умолчанию 22, логин: pi, пароль: raspberry. Нажимаем “Save”.

Выходим из окна настроек и нажимаем на иконку:

Выбираем только что созданное подключение и подключаемся к Raspberry Pi. Если все прошло успешно, то команды, которые мы пишем в окне Terminus, будут выполняться на Raspberry Pi. Попробуем вывести информацию о системе при помощи команды `uname -a`.

MQTT

MQTT (Message Queuing Telemetry Transport) - это облегченный сетевой протокол, который чаще всего используется для межмашинного взаимодействия. Протокол обычно работает в рамках стека TCP/IP, но возможны и иные варианты.

Для того, что бы перейти к дальнейшему описанию протокола MQTT и практической работе с ним, необходимо разобраться с понятием протокол. Как правило, под протоколом понимают набор определенных правил взаимодействия. В зависимости от того, посредством чего происходит это взаимодействие, на каком уровне и между чем, мы можем разделять протоколы на различные виды. Более строго протоколы связи (или коммуникации) можно определить так:

Протокол связи – это система правил, позволяющие двум и более субъектам передавать информацию. Протокол определяет правила, синтаксис, семантику и синхронизацию коммуникации, а также определяет возможные способы устранения ошибок или обозначает их отсутствие.

Теперь вернемся к MQTT и выделим следующие отличительные особенности:

1. Протокол MQTT ориентирован на соединения между устройствами, то есть обмен данными, например, в IoT. MQTT считается энергоэффективным протоколом, позволяющим уменьшить энергопотребление устройства за счет минимизации размера отправляемых данных.
2. Хорошо подходит для рассылки сообщений «один ко многим».
3. Присутствует механизм уведомления о ненормальном (аномальном) отключении.
4. Любые данные, опубликованные или полученные брокером MQTT, будут закодированы в двоичном формате.
5. Позволяет настроить уровень надежности доставки сообщений:
   - «Максимум один раз»: в таком режиме может произойти потеря сообщения. Этот уровень может использоваться, например, с данными датчика атмосферного давления, так как даже если последнее показание было не доставлено, то следующее будет опубликовано вскоре после этого.
   - «По крайней мере, один раз»: в этом режиме сообщения гарантированно поступают, но могут возникать дубликаты.

   - «Ровно один раз»: в этом режиме сообщение гарантированно поступит ровно один раз. Этот уровень можно использовать, например, в биллинговых системах, в которых дубликаты или утерянные сообщения могут привести к неправильной оплате.

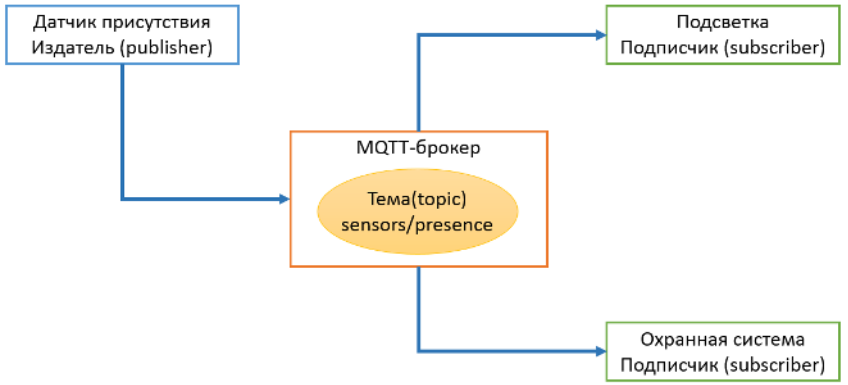
Также важной особенностью протокола является работа по модели «издатель-подписчик» (Pub/Sub) согласно которой отправитель сообщения (издатель), например, датчик температуры, не привязан к получателю (подписчику) - вместо этого сообщения публикуются издателем в определенный раздел, и уже на этот раздел сообщений может подписаться один или несколько подписчиков.

В MQTT структура разделов древовидная, разделы называют темами или топиками (от английского topic), что может напоминать форумы, в которых есть разделы, подразделы, а в них темы.

MQTT-брокер отслеживает подписки и структуру тем. Брокер должен быть доступен всем издателям и подписчикам и может располагаться как в локальной сети, так и на удаленном сервере.

Существует различное программное обеспечение для реализации функционала MQTT-брокера, например, IBM WebSphere MQ, emqttd, Mosquitto. В данном пособии в качестве MQTT-брокера будет использоваться именно Mosquitto как одна из наиболее распространенных и открытых реализаций.

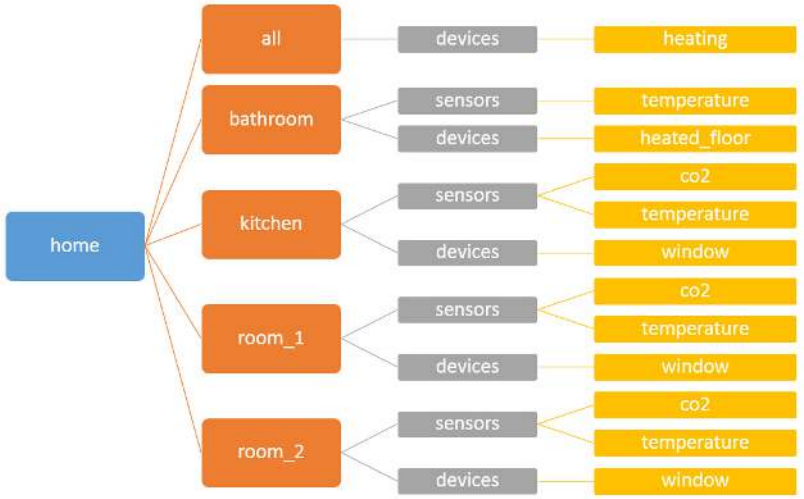
Рассмотрим схему взаимодействия брокера с издателем и подписчиками:



В данной схеме ИК-датчик движения (присутствия) при обнаружении движения отправляет сообщение о событии в тему (топик) sensors/presence. В свою очередь брокер принимает сообщение и направляет его тем устройствам, которые подписаны на данную тему, например, охранной системе и подсветке, установленной в помещении. При этом устройства, которые не подписаны на данную тему, не получают сообщение, поступающее с датчика. Это базовый пример, но схема может быть гораздо сложнее. В один и тот же топик (тему) могут публиковаться данные от нескольких издателей, темы могут разделяться и группироваться по различным параметрам (расположение, назначение, производитель, и т.д.).

Далее разберем модель взаимодействия издателя, подписчика и брокера на примере системы контроля климата двухкомнатной квартиры. Допустим, что в квартире, в каждой комнате, ванне и кухне установлены датчики температуры, датчики, определяющие уровень углекислого газа (CO2), установлены в помещениях, имеющих окна (кухня и комнаты). Окна оборудованы устройствами для открывания/закрывания окон. Квартира отапливается от газового котла (температура выставляется сразу для всей квартиры), но в ванной также есть система теплого пола - поэтому в ванной комнате может быть теплее, чем в остальной квартире. Все датчики и устройства обозначены на схеме ниже и привязаны к помещению.

Темы (топики) разбиты, опираясь на привязку к помещению. Это дерево тем может выглядеть следующим образом:



Структура тем (топиков) с показаниями на MQTT брокере:

Теперь определим, на какие темы должны быть подписаны устройства в рассматриваемой нами системе контроля климата.

- Отопительный котел (heating, прим. далее в скобках будут указываться названия устройств в структуре тем) - должен получать информацию со всех датчиков температуры, анализировать их и затем увеличивать, уменьшать или оставлять неизменной мощность.
- Устройство, открывающее окна (window) - должно получать информацию об уровне углекислого газа только того помещения, в котором оно установлено.
- Теплый пол в ванной (heated_floor) - должен получать информацию о температуре только в ванной.

С последними двумя устройствами все просто: им нужны показания с одного датчика и нужно подписаться на соответствующий топик:

1. home/kitchen/sensors/co2 (для устройства, которое открывает и закрывает окно на кухне)
2. home/room_1/sensors/co2 (для устройства, которое открывает и закрывает окно в первой комнате)
3. home/room_2/sensors/co2 (для устройства, которое открывает и закрывает окно во второй комнате)
4. home/bathroom/sensors/temperature (для теплого пола в ванной)

Первый случай более сложный, т.к. нам нужны показания со всех датчиков температуры. В этом случае можно по отдельности подписаться на каждый топик:

- home/kitchen/sensors/temperature
- home/room_1/sensors/temperature
- home/room_2/sensors/temperature
- home/bathroom/sensors/temperature

Но для подобных случаев, когда нужно сразу подписаться на несколько топиков, есть более лаконичный способ: для этого используются специальные символы “+” и “#”.

- подписка на тему home/# означает подписку на тему home и все дочерние (вложенные темы)
- подписка на тему home/+ - это подписка на темы на один уровень под home (можно сказать, дочерние первого поколения), но не на саму тему home.
- подписка на тему home/+/# - это подписка на все темы, дочерние home, но не на саму тему home.

Таким образом, оформить подписку на все темы с показаниями температуры можно следующим способом:

home/+/sensors/temperature

Подписаться на все показания датчиков, например, для вывода показаний на панель управления в смартфоне, можно при помощи команды:

home/+/sensors/+/#

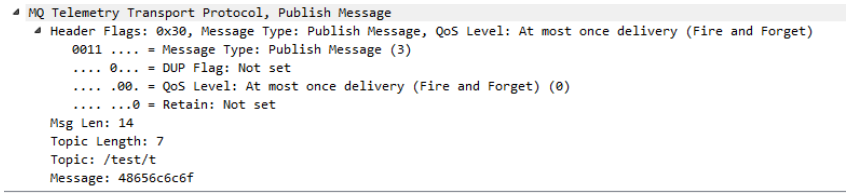
Такая запись позволяет очень кратко, быстро и гибко настраивать подписки на десятки и сотни датчиков, при условии, что изначально дерево тем было корректно спланировано. Если названия темы (топики) получали хаотично, то применить вышеописанный способ будет сложно.

Принимая во внимание относительно малый размер отправляемых данных, использование модели «издатель-подписчик», наличие возможности разделения взаимодействия устройств по времени (сенсор и исполнительное устройство не обязаны подключаться к брокеру одновременно), наличие гибкости в настройке надежности доставки сообщений и управление подписками делают протокол MQTT удобным для использования в IoT системах.

Что происходит при отправке показаний датчика и как выглядит сообщение, отправленное при помощи протокола MQTT?

Для того, чтобы просмотреть отправляемые по сети сообщения, воспользуемся программой-анализатором трафика компьютерных сетей wireshark.

Данное программное обеспечение можно скачать с официального сайта разработчиков https://www.wireshark.org/ и посмотреть, как выглядят сообщения, отправленные при помощи других протоколов (например, HTTP или POP3)



Рассмотрим данный фрагмент. Сначала идет заголовок, в котором указывается тип сообщения и различные настройки. Затем идут поля, сообщающие длину всего сообщения (не только отправляемого значения, но и топика) и длину темы (топика). После этих полей идут названия темы

и текст сообщения (в данном случае в сообщение отправлялась фраза “Hello world” в тему test/t).

Разберем подробнее флаги настроек сообщения типа “PUBLISH”. Имеется три флага:

- DUP - занимает один бит и устанавливается, когда совершается повторная отправка сообщения. При установленном флаге переменный заголовок (следует за постоянной частью заголовка) должен содержать Message ID (идентификатор сообщения).
- QoS - занимает два бита и указывает качество обслуживания, то есть уровень надежности доставки сообщений («максимум один раз»(0), «По крайней мере, один раз»(1), «Ровно один раз»(2)).
- RETAIN – данный флаг занимает 1 бит и, если значение равно 1, то при получении брокер сохранит его и при следующей подписке на этот топик отправит сообщение с флагом RETAIN подписчику.

Фиксированная часть заголовка состоит из 1 байта и делится на две части. Схематично в виде таблицы фиксированную часть заголовка можно представить так:

БИТЫ	7	6	5	4	3	2	1	0
	Тип пакета				Флаги, специфичные для каждого типа пакета (сообщения)			

Далее представлены несколько примеров фрагментов сообщений, отправленных MQTT-брокеру:

```
MQ Telemetry Transport Protocol, Publish Message
  Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
      0011 .... = Message Type: Publish Message (3)
      .... 0... = DUP Flag: Not set
      .... .00. = QoS Level: At most once delivery (Fire and Forget) (0)
      .... ...0 = Retain: Not set
    Msg Len: 11
    Topic Length: 8
    Topic: /test/t2
    Message: 31
```


(Число 1 в тему test/t2)

```
MQ Telemetry Transport Protocol, Publish Message
  Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
      0011 .... = Message Type: Publish Message (3)
      .... 0... = DUP Flag: Not set
      .... .00. = QoS Level: At most once delivery (Fire and Forget) (0)
      .... ...0 = Retain: Not set
    Msg Len: 19
    Topic Length: 14
    Topic: /room1/sensor2
    Message: 313031
```

(Число 101 в тему room1/sensor2).

Мы разобрались с общими понятиями MQTT и его применением. Теперь перейдем к практике - установке и настройке MQTT-брокера и отправке сообщений с использованием протокола MQTT.

Установка MQTT-брокера

В качестве MQTT-брокера будем использовать Mosquitto, который является одним из самых популярных MQTT-брокеров.

Установим Mosquitto с помощью менеджера пакетов apt:

```
sudo apt install mosquitto
```

В командной строке появится необходимое количество дискового пространства для установки и вопрос, действительно ли мы хотим установить пакет. Ответим утвердительно и через некоторое время установка будет завершена.

После установки MQTT-брокер Mosquitto автоматически запускается. Остановить его можно следующей командой:

```
sudo /etc/init.d/mosquitto stop
```

При необходимости запуска используется команда:

```
sudo /etc/init.d/mosquitto start
```

Написав вместо start “restart” или “status”, мы можем, соответственно, перезапустить или узнать текущий статус MQTT-брокера.

Остановив Mosquitto, мы можем изменить настройки, которые описываются в файле конфигурации, находящемся по адресу etc/mosquitto/mosquitto.conf

Если вы работаете с Raspberry Pi, то подключив монитор, вы можете запустить файловый менеджер и найти данный файл по указанному пути. Если вы подключились к Raspberry Pi через ssh, то воспользуйтесь текстовым редактором nano:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Подробно о всех настройках можно узнать в документации: https://mosquitto.org/man/mosquitto-conf-5.html

Например, возможно, переопределить порт для сообщений (по умолчанию 1883), настроить логирование различных событий или шифрование. Но в данном пособии мы будем использовать конфигурационный файл по умолчанию:

```
# Place your local configuration in /etc/mosquitto/
conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.
example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
```

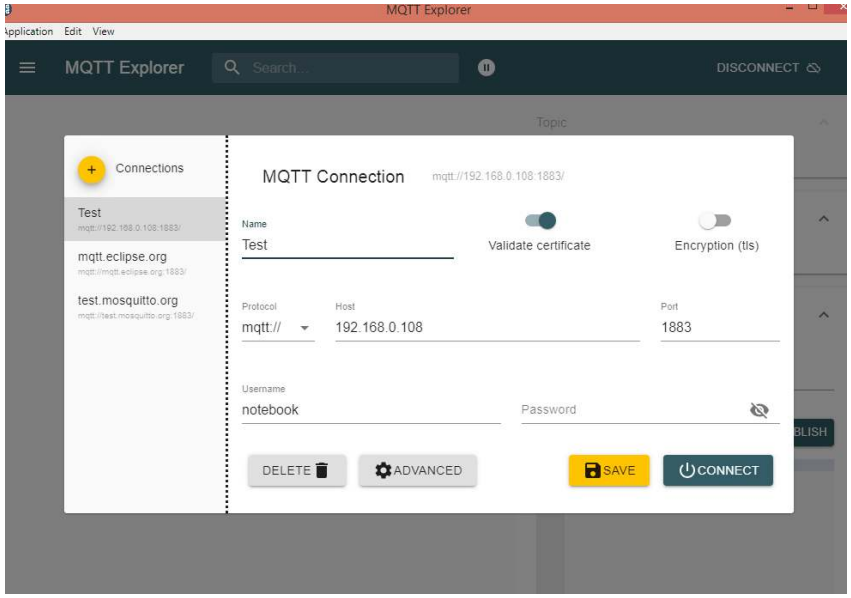
Проверим, что брокер работает, запросив информацию командой:

```
sudo /etc/init.d/mosquitto status
```

Установка MQTT-клиента в ОС Windows

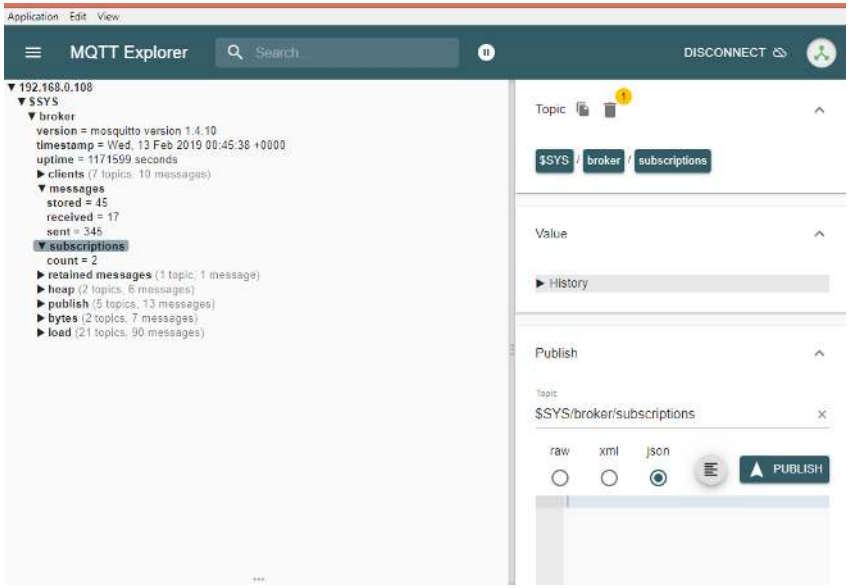
Скачиваем MQTT-клиент (https://mqtt-explorer.com/) или собираем из исходников (https://github.com/thomasnordquist/MQTT-Explorer).

Запускаем MQTT Explorer. Далее перед нами возникает окно настроек:



Вводим в поле host ip-адрес Raspberry Pi и порт, который прослушивает MQTT брокер Mosquitto (по умолчанию 1883). Указываем имя устройства (username), пароль можно оставить пустым, если не включали авторизацию. Сохраняем настройки и пробуем подключится, нажав “Connect”.

Если подключение прошло успешно, перед нами появляется дерево топиков (тем):

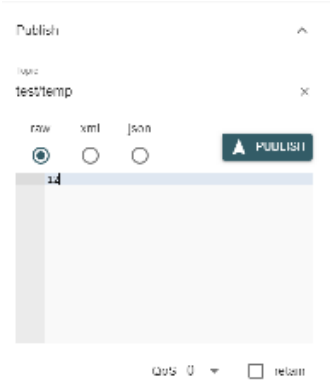


В терминале Raspberry Pi введем Mosquitto:

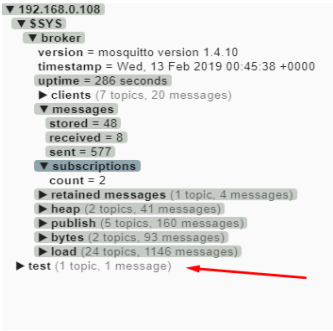
```
pi@raspberrypi:~ $ mosquitto
1589501268: mosquitto version 1.4.10 (build date Wed, 13 Feb 2019 00:45:38 +0000) starting
1589501268: Using default config.
1589501268: Opening ipv4 listen socket on port 1883.
1589501268: Opening ipv6 listen socket on port 1883.
1589501268: New connection from 192.168.0.106 on port 1883.
1589501268: New client connected from 192.168.0.106 as mqtt-explorer-53c68a61 (c1, k60, u'notebook').
```

Это позволит увидеть, что к брокеру подключился новый клиент с именем, которое мы указали.

Отправим сообщение в топик test/temp (представим, что это значение температуры).



Данный топик тут же отобразится в дереве топиков:



Развернув его, вы сможете посмотреть значение temp.

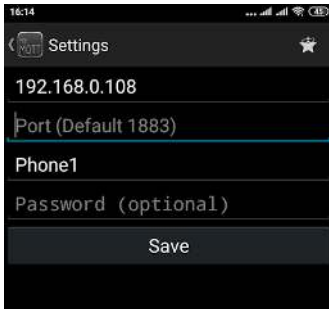
Теперь нажмем disconnect и отключимся от брокера. В терминале появится вывод:

```
Client mqtt-explorer-53c68a61 disconnected.
```

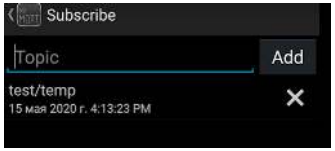

MQTT-клиент для ОС Android

Установим из PlayMarket приложение MyMQTT или аналогичное ему.

Запустим MQTT брокер на Raspberry Pi, а в приложении MyMQTT перейдем в раздел настроек и укажем необходимые данные для подключения, сохранив их:



После этого перейдем в раздел подписок (subscribe) и подпишемся на топик “test/temp”. Затем перейдем в dashboard.

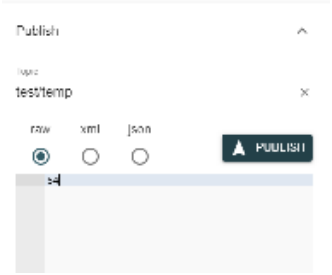


Далее необходимо подключиться к брокеру с помощью клиента MQTT Explorer.

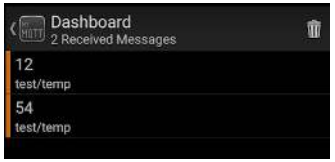
Проверим в терминале Raspberry Pi, что и смартфон, и компьютер подключены к брокеру.

```
1589548968: New client connected from 192.168.0.100 as root.1589548968422 (c1, k60, u'Phone1').
1589548977: New connection from 192.168.0.106 on port 1883.
1589548977: New client connected from 192.168.0.106 as mqtt-explorer-53c68a61 (c1, k60, u'notebook').
```

После этого отправим в топик test/temp любое значение:

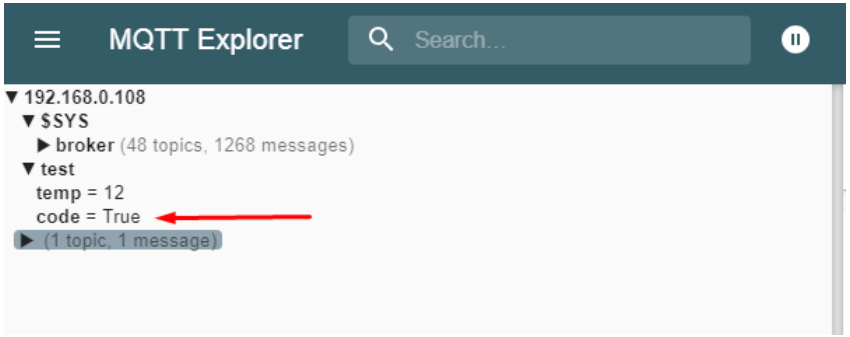


Это значение отобразится на dashboard приложения MyMQTT . При отправлении еще одного значения оно также отобразится на смартфоне.



Перейдем в раздел Publish в приложении MyMQTT и отправим сообщение в топик “test/code”:

Данное сообщение отобразится в MQTT Explorer:



Установка и использование MQTT-клиента на Raspbian

На данном этапе мы научились настраивать обмен сообщениями между устройствами. Но на практике с большой долей вероятности потребуется передавать информацию не только между пользовательскими устройствами (компьютер, смартфон), а подавать различные команды одноплатному компьютеру, например, о включении и выключении освещения.

В этом разделе мы продемонстрируем, как запускать MQTT-клиент на одноплатном компьютере под управлением операционной системы Raspbian. MQTT брокер и MQTT могут работать на одном устройстве.

Установим MQTT-клиент командой:

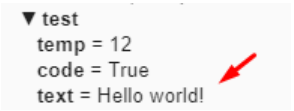
```
sudo apt install mosquitto-clients
```

Используя mosquitto_sub и mosquitto_pub, можно подписываться на сообщения в топиках и публиковать сообщения.

Попробуем опубликовать сообщение в топик “test/text”:

mosquitto_pub -t test/text -m “Hello world!”

С помощью mqtt клиента подпишемся на данный топик и проверим, что сообщение действительно публикуется в нужном топике.



Также при помощи ключа -f можно передавать в топик значения из файла, который может формироваться, например, скриптом управления освещением.

Создадим скрипт, который будет записывать в файл одно из двух значений 0 или 1, тем самым моделируя ситуацию с состоянием освещения (включено или выключено). Для этого создадим отдельную директорию и перейдем в неё:

mkdir ~/lighting

cd ~/lighting

Создадим скрипт, который будет генерировать случайным образом 0

или 1 и записывать значения в файл status в течении одной минуты.

nano generator.py

```
from time import time, sleep #Подключаем модули для работы с временем
from random import randint #Подключаем модуль для генерации случайного значения

timeFinish = time()+60 #получаем текущее время и добавляем к нему 60 секунд
while(time()<timeFinish): #проверяем не вышло ли время работы скрипта
    f = open('status', 'w') #открываем файл и очищаем его содержимое
    f.write(str(randint(0, 1))) #генерируем 0 или 1, превращаем число в строку и записываем в файл
    f.close() #закрываем файл
    sleep(1) #ждем 1 секунду
print('end generator') #когда цикл закончился выводим сообщение
```

Теперь создадим bash скрипт, который будет раз в пять секунд выполнять команду отправки сообщения из файла в нужный топик. Для этого выполним команду:

```
nano send
```

Введем следующее содержимое файла:

```
#!/bin/bash

for ((i=0;i<12;i++))
do
    mosquitto_pub -t test/lighting/status -f ./status
    sleep 5
done
echo “End send”
```

После этого запустим оба скрипта в фоновом режиме

```
python generator.py &
bash send &
```

В любом MQTT клиенте вы сможете подписаться на топик test/lighting/status и увидеть приходящие сообщения. Важно отметить, что данная конфигурация очень проста и предоставлена в качестве примера. Потенциально может возникать ситуация, при которой один скрипт только что открыл и очистил файл, а скрипт для отправки сообщения при отправке обнаружит пустой файл.

Далее продемонстрируем, как подписывать Raspberry Pi на различные топики. Например подпишемся на топик “test/text”:

```
mosquitto_sub -t test/text
```

После этого при помощи MyMQTT или MQTT Explorer отправим сообщение и проверим, отобразилось ли оно в терминале.

```
pi@raspberrypi:~/lighting $ mosquitto_sub -t test/text
Hello
```

Мы можем перенаправить вывод сообщений в файл, для того чтобы потом скрипт, который мы напишем (например, для включения и отключения освещения или открытия и закрытия двери), мог работать с поступившими сообщениями.

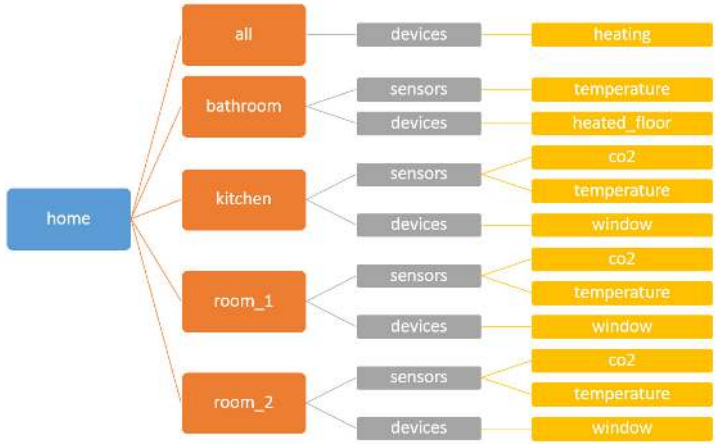
```
mosquitto_sub -t test/text > mes.txt
```

Отправим несколько сообщений. После этого откроем файл mes.txt и найдем в нем сообщения, которые мы отправляли.

Задания

Для закрепления навыков предлагаем выполнить несколько заданий:

1. Выберите любой объект для внедрения умной системы, например, свой дом, школу или университет. Подумайте, какие датчики и исполнительные устройства нужны на данном объекте, как они взаимодействуют друг с другом?
2. В какие темы (топики) можно сгруппировать сообщения от датчиков и исполнительных устройств?



3. Составьте схему топиков наподобие разобранной в разделе “MQTT”.
4. Настройте автоматическую генерацию и отправку показаний в топики, предназначенные для показаний с датчиков.
5. Напишите скрипт, который при поступлении определенного сообщения с мобильного устройства в топик “open_time” фиксирует и сохраняет в файл время поступления сообщения.

Заключение

В данном пособии мы кратко, просто и емко рассказали про протокол MQTT, его применение и дали инструкции по практической работе с MQTT.

Надеемся, что вы сочли представленные материалы полезными и интересными. Тем не менее, многие темы, как, например, авторизация устройств и шифрование передаваемых данных, в данном пособии затронуты не были. Если вас заинтересовал протокол MQTT и брокер Mosquitto, обратите внимание на список источников, которые использовались для подготовки пособия и в первую очередь на официальную документацию:

https://github.com/mqtt/mqtt.github.io/wiki
https://mosquitto.org/man/

Источники

1. Официальная документация mosquitto [Электронный ресурс]. URL: https://mosquitto.org/man/ (дата обращения: 08.07.2020).
2. Протокол MQTT: концептуальное погружение [Электронный ресурс] URL: https://habr.com/ru/post/463669/ (дата обращения: 08.07.2020).
3. MQTT [Электронный ресурс] URL: http://mqtt.org/ (дата обращения: 08.07.2020).
4. MQTT community [Электронный ресурс] URL: https://github.com/mqtt/mqtt.github.io/wiki (дата обращения: 08.07.2020).
5. Установка брокера сообщений mosquitto в debian 10 [Электронный ресурс] URL: https://www.8host.com/blog/ustanovka-brokera-soobshhenij-mosquitto-v-debian-10/ (дата обращения: 08.07.2020).
6. Установка MQTT брокера Mosquitto на Raspberry Pi и Orange Pi [Электронный ресурс] URL: https://robot-on.ru/articles/ystanovka-mqtt-brokera-mosquitto-raspberry-orange-pi (дата обращения: 08.07.2020).
7. Протокол MQTT. Особенности, варианты применения, основные процедуры MQTT Protocol. [Электронный ресурс] URL: http://lib.tssonline.ru/articles2/fix-corp/protokol-mqtt-osobennosti-varianty-primeneniya-osnovnye-protsedury-mqtt-protocol. (дата обращения: 08.07.2020).
8. The Massively Scalable MQTT Broker for IoT and Mobile Applications [Электронный ресурс]. URL: https://emqtt.io/ (дата обращения: 08.07.2020).
9. Raspberry Pi [Электронный ресурс]. URL: https://www.raspberrypi.org/ (дата обращения: 08.07.2020).

Skoltech

Сколтех
Территория Инновационного Центра
“Сколково”, Большой бульвар д.30, стр.1
Москва 121205
Телефон: +7 (495) 280 14 81

Email
iot@skoltech.ru
Сайт
https://iot.skoltech.ru/