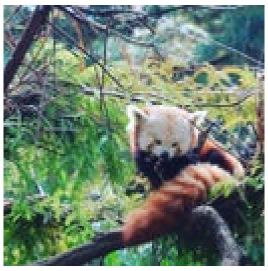
How to Use MQTT With the Raspberry Pi and ESP8266

instructables.com/How-to-Use-MQTT-With-the-Raspberry-Pi-and-ESP8266/

Thomas Varnish

By Thomas Varnish
More by the author:



About: I'm a 20-year-old aspiring Physicist, who enjoys coding and making stuff! More About Thomas Varnish »
In this Instructable, I will explain what the MQTT protocol is and how it is used to communicate between devices.Then, as a practical demonstration, I shall show you how to setup a simple two client system, where an ESP8266 module will send a message to a Python program when a button is pushed. Specifically, I am using an Adafruit HUZZAH module for this project, a Raspberry Pi and a desktop computer. The Raspberry Pi will be acting as the MQTT broker, and the Python client will be run from a separate desktop computer (optional, as this could be run on the Raspberry Pi).

To follow along with this Instructable, you will need to have some basic knowledge of electronics, and how to use the Arduino software. You should also be familiar with using a command line interface (for the Raspberry Pi). Hopefully, once you've gained the knowledge of what MQTT is, and how to use it in a basic scenario, you will be able to create your own IoT projects!

Required Parts

- 1 x Raspberry Pi, connected to a local network (running Jessie)
- 1 x ESP8266 Module (Adafruit HUZZAH)
- 1 x Breadboard
- 3 x Jumper Wires (Male-to-Male)
- 1 x Pushbutton

- 1 x 10k Ohm Resistor (Brown-Black-Orange colour code)

I've created this Instructable, as MQTT has always interested me as a protocol and there are many different ways it could be used. However, I couldn't seem to get my head around how to code devices to use it. This was because I didn't know/understand what was actually going on to take my "Hello, World!" from device A and send it to device B. Hence, I decided to write this Instructable to (hopefully) teach you how it works, and to also reinforce my own understanding of it!

Step 1: What Is MQTT?

MQTT, or MQ Telemetry Transport, is a messaging protocol which allows multiple devices to talk to each other. Currently, it is a popular protocol for the Internet of Things, although it has been used for other purposes - for example, Facebook Messenger. Interestingly MQTT was invented in 1999 - meaning it's as old as me!

MQTT is based around the idea that devices can **publish** or **subscribe** to **topics**. So, for example. If Device #1 has recorded the temperature from one of its sensors, it can **publish** a message which contains the temperature value it recorded, to a topic (e.g. *"Temperature"*). This message is sent to an MQTT Broker, which you can think of as a switch/router on a local area network. Once the MQTT Broker has received the message, it will send it to any devices (in this case, Device #2) which are **subscribed** to the same topic.

In this project, we will be publishing to a topic using an ESP8266, and creating a Python script that will subscribe to this same topic, via a Raspberry Pi which will act as the MQTT Broker. The great thing about MQTT is that it is lightweight, so it perfect for running on small microcontrollers such as an ESP8266, but it is also widely available - so we can run it on a Python script as well.

Hopefully, at the end of this project, you will have an understanding of what MQTT is and how to use it for your own projects in the future.

Step 2: Installing the MQTT Broker on the Raspberry Pi

To setup our MQTT system, we need a broker, as explained in the previous step. For the Raspberry Pi, we will be using the "*Mosquitto*" MQTT broker. Before we install this, it is always best to update our Raspberry Pi.

```
sudo apt-get update
sudo apt-get upgrade
```

Once you've done this, install **mosquitto** and then the **mosquitto-clients** packages.

```
sudo apt-get install mosquitto -y
sudo apt-get install mosquitto-clients -y
```

When you've finished installing these two packages, we are going to need to configure the broker. The mosquitto broker's configuration file is located at **/etc/mosquitto/mosquitto.conf**, so open this with your favourite text editor. If you don't have a favourite text editor or don't know how to use any of the command line editors, I'll be using **nano** so you can follow along:

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

At the bottom of this file, you should see the line:

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

Delete this line. Add the following lines to the bottom of the file.

```
allow_anonymous false
password_file /etc/mosquitto/pwfile
listener 1883
```

By typing those lines, we've told mosquitto that we don't want anyone connecting to our broker who doesn't supply a valid username and password (we'll get on to set these in a second) and that we want mosquitto to listen for messages on port number 1883.

If you don't want the broker to require a username and password, don't include the first two lines that we added (i.e. allow_anonymous... and password_file...). If you have done this, then skip to rebooting the Raspberry Pi.

Now close (and save) that file. If you are following along with the nano example, press **CTRL+X**, and type **Y** when prompted.

Because we've just told mosquitto that users trying to use the MQTT broker need to be authenticated, we now need to tell mosquitto what the username and password are! So, type the following command - replacing **username** with the username that you would like - then enter the password you would like when prompted (Note: if, when editing the configuration file, you specified a different **password_file** path, replace the path below with the one you used).

```
sudo mosquitto_passwd -c /etc/mosquitto/pwfile username
```

As we've just changed the mosquitto configuration file, we should reboot the Raspberry Pi.

```
sudo reboot
```

Once the Raspberry Pi has finished rebooting, you should have a fully functioning MQTT broker! Next, we are going to try to interact with it, using a number of different devices/methods!

Step 3: Testing the Broker

Once you've installed mosquitto on the Raspberry Pi, you can give it a quick test - just to make sure everything is working correctly. For this purpose, there are two commands that we can use on the command line. **mosquitto_pub** and **mosquitto_sub**. In this step, I will guide you through using each of these to test our broker.

In order to test the broker, you will need to open two command line windows. If you are using Putty or another SSH client, this is as simple as opening another SSH window and logging in as usual. If you are accessing your Pi from a UNIX terminal, this is exactly the same. If you are using the Raspberry Pi directly, you will need to open two terminal windows in the GUI mode (the command **startx**can be used to start the GUI).

Now that you have opened two windows, we can get started on the testing. In one of the two terminals, type the following command, replacing **username** and **password** with the ones you setup in the previous step.

```
mosquitto_sub -d -u username -P password -t test
```

If you decided not to set a username and password in the previous step, then from now on, ignore the -u and -P flags in the commands. So, as an example, the mosquitto_sub command would now be:

```
mosquitto_sub -d -t test
```

The mosquitto_sub command will subscribe to a topic, and display any messages that are sent to the specified topic in the terminal window. Here, **-d** means **debug mode**, so all messages and activity will be output on the screen. **-u** and **-P** should be self-explanatory. Finally, **-t** is the name of the **topic** we want to subscribe to - in this case, "test".

Next, in the other terminal window, we are going to try and publish a message to the "test" topic. Type the following, remembering again to change **username** and **password**:

```
mosquitto_pub -d -u username -P password -t test -m "Hello, World!"
```

When you press enter, you should see your message **"Hello, World!"** appear in the first terminal window we used (to subscribe). If this is the case, you're all set to start working on the ESP8266!

Step 4: Setting Up the ESP8266 (Adafruit HUZZAH)

This step if specific to the Adafruit HUZZAH (as that is what I am using to complete this project). If you are using a different Arduino / ESP8266 device, you may wish to skip this step. However, I would advise you skim read it, just in case there is any information here that may be relevant to you.

For this project, I am going to be programming the HUZZAH with the Arduino software. So, if you haven't already, make sure to install the Arduino software (**newer** than **1.6.4**). You can download it here.

Once you have installed the Arduino software, open it and navigate to *File->Preferences*. Here you should see (near the bottom of the window) a text box with the label: **"Additional Boards Manager URLs"**. In this text box, copy and paste the following link:

```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

Click OK to save your changes. Now open the Board Manager (*Tools->Board->Board Manager*) and search for ESP8266. Install the **esp8266 by ESP8266 Community** package. Restart the Arduino software.

Now, before we can program the board, we need to select a few different options. In the Tools menu option, select **Adafruit HUZZAH ESP8266** for Board, **80 MHz** for the CPU Frequency (you can use 160 MHz if you wish to overclock it, but for now I'm going to use 80 MHz), **4M (3M SPIFFS)** for the Flash Size, and **115200** for the Upload Speed. Also, make sure to select the COM port that you are using (this will depend on your setup).

Before you can upload any code, you need to make sure that the HUZZAH is in bootloader mode. To enable this, hold down the button on the board marked **GPIO0**, and whilst this is held, hold down the **Reset** button as well. Then, release the **Reset** button, and then **GPIO0**. If you have done this correctly, the red LED that came on when you pressed GPIO0 should now be dimly lit.

To upload code to the microcontroller, first make sure the HUZZAH is in bootloader mode, then simply click the upload button in the Arduino IDE.

If you are having any trouble setting up the HUZZAH, further information can be found at Adafruit's own tutorial.

Step 5: Programming the ESP8266

Now we will begin to program the ESP8266, but before we can start, you will need to install the following libraries in the Arduino Library manager (*Sketch->Include Libraries->Manage Libraries*)

- **Bounce2**
- **PubSubClient**

Once you've installed those libraries, you will be able to run the code I've included in this Instructable (MQTT_Publish.zip). I've made sure to comment it so that you can understand what each section is doing, and this should hopefully enable you to adapt it to your needs.

Remember to change the constants at the top of the code so that your ESP8266 can connect to your WiFi network and your MQTT Broker (the Raspberry Pi).

If you decided not to set a username and password for the MQTT Broker, then download the MQTT_PublishNoPassword.zip file instead.

MQTT_Publish.zip

Download

MQTT_PublishNoPassword.zip

Download

Step 6: Installing Python Client (paho-mqtt)

Thankfully, this step is very simple! To install the mosquitto python client, you just need to type the following into the command line (Linux/Mac) or even command prompt (Windows).

```
pip install paho-mqtt
```

Note: Windows command prompt may have an issue running the ***pip*** *command if you didn't specify that you wanted pip installed and python added to your PATH variable when you installed Python. There are a number of ways of fixing this, but I think just reinstalling Python is the easiest way. If in doubt - give it a google!*

Step 7: Python Client - Subscribing

In this step, we are going to setup the Python script (either on the Raspberry Pi itself or on another computer connected to the network) to handle all of the messages that are sent (published) by the ESP8266 to the MQTT topic.

I have included the python code below (PythonMQTT_Subscribe.py), which has been commented to help you understand what is going on, but I will explain some of the main features here as well.

If you didn't set a username and password for the MQTT connection earlier, download the PythonMQTT_SubscribeNoPassword.py file instead.

PythonMQTT_Subscribe.py

Download

PythonMQTT_SubscribeNoPassword.py

Download

Step 8: Communicating Between ESP8266 Devices

If you want to set up an IoT network, for example, you may wish to communicate between ESP8266 devices. Thankfully, this isn't much more complex than the code we've written before, however, there are a couple of notable changes.

For one ESP to send data to another, the first ESP will need to **publish** to the topic, and the second ESP will need to **subscribe** to that topic. This setup will allow for a one-way conversation - ESP(1) to ESP(2). If we want ESP(2) to talk back to ESP(1), we can create a new topic, to which ESP(2) will publish, and ESP(1) will subscribe. Thankfully, we can have multiple subscribers on the same topic, so if you want to send data to a number of systems, you will only need one topic (to which they all subscribe, except the device which is sending the data, as that will be publishing).

If you need help figuring out what each device needs to do, think about the system as a room of people. If ESP(1) is publishing, you can imagine this device as a "speaker", and any devices that are subscribing to the topic are "listeners" in this example.

I have included some example code below, which demonstrates how an ESP8266 can subscribe to a topic, and listen for certain messages - 1 and 0. If 1 is received, the on-board LED (for the HUZZAH - GPIO 0) is switched on. If 0 is received, this LED is switched off.

If you want to process more complex data, this should be done in the **ReceivedMessage** function (see code).

For your own projects, if you need to both send and receive data, you can incorporate the publish function from the previous example into the code included in this step. This should be handled in the main Arduino **loop()** function.

Remember to change the variables at the top of the code to suit your network!

MQTT_Subscribe.zip

Download

Participated in the
Microcontroller Contest 2017

2 People Made This Project!

Did you make this project? Share it with us!

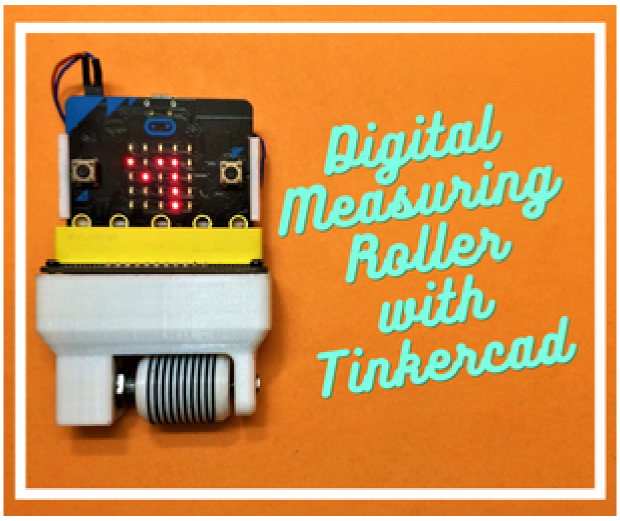
Recommendations

Automatic Rainbow Staircase Lighting by MagicManu in LEDs
22 1.8K

BEAM Solar Powered Pummer (Heart Shaped PCB) by NanoRobotGeek in Art
14 1.7K

Digital Measuring Roller Using Microbit & Tinkercad by Makertronics in Tools
6 405

Pocket Dice! Electronic Dice for Liars Dice and More by sunyecz06 in Electronics
117 12K



47 Discussions

Samm2017
Question 18 days ago

Hello, thanks for the tutorial but I have a qustion, can we just resend the messege via sockets to another computer?
Thank you.

BeerC1
Tip 1 year ago

Nice example. I had a problem with connection to broker server. Just set wifi mode to STA and it works perfect to me.
...

WiFi.mode(WIFI_STA);
...
WiFi.begin(ssid, wifi_password);

sp.fy18
1 year ago

Did you make the raspberry pi as an accesspoint (hotspot)

rfire88
Question 1 year ago

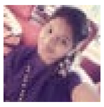
Hey man thanks for the tutorial but I am facing a problem. I tried running the python script but it throws out this error:

```
execfile('PythonMQTT_Subscribe.py')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "PythonMQTT_Subscribe.py", line 44, in <module>
client.connect(mqtt_broker_ip, 1883)
File "/home/rfire88/.local/lib/python2.7/site-packages/paho/mqtt/client.py", line 937, in connect
return self.reconnect()
File "/home/rfire88/.local/lib/python2.7/site-packages/paho/mqtt/client.py", line 1071, in reconnect
sock = self._create_socket_connection()
File "/home/rfire88/.local/lib/python2.7/site-packages/paho/mqtt/client.py", line 3522, in _create_socket_connection
return socket.create_connection(addr, source_address=source, timeout=self._keepalive)
File "/usr/lib/python2.7/socket.py", line 557, in create_connection
for res in getaddrinfo(host, port, 0, SOCK_STREAM):
socket.gaierror: [Errno -2] Name or service not known
```

Please help.
Also I have one more doubt. What is to be put in MQTT broker IP address. Where to get that address.
Thanks.
1

ShubhadaA3
Question 1 year ago on Step 8

How to store data from clients on text file?
What is the procedure to connect multiple esp8266 to raspberry pi broker?
Please guide



shyamalas2
2 years ago

```
Traceback (most recent call last):

File "C:\Users\hp\Downloads\F02IQZQIZ6CAGZ9 (1).py", line 47, in <module>

client.loop_forever()

File "C:\Users\hp\lib\site-packages\paho\mqtt\client.py", line 1481, in loop_forever

rc = self.loop(timeout, max_packets)

File "C:\Users\hp\lib\site-packages\paho\mqtt\client.py", line 1003, in loop

rc = self.loop_read(max_packets)

File "C:\Users\hp\lib\site-packages\paho\mqtt\client.py", line 1284, in loop_read

rc = self._packet_read()

File "C:\Users\hp\lib\site-packages\paho\mqtt\client.py", line 1849, in _packet_read

rc = self._packet_handle()

File "C:\Users\hp\lib\site-packages\paho\mqtt\client.py", line 2311, in _packet_handle

return self._handle_connack()

File "C:\Users\hp\lib\site-packages\paho\mqtt\client.py", line 2372, in _handle_connack

self.on_connect(self, self._userdata, flags_dict, result)

TypeError: on_connect() takes 3 positional arguments but 4 were given
```

this is the error i am facing while executing PythonMQTT_Subscribe.py file..please help

ahsan495
2 years ago

I'm able to send messages to the mqtt broker. However, my python client is unable to connect to mqtt broker. What could be the reason? Please help.

emnavarro02
Question 2 years ago

Hi,
I'm with a problem and hope you could help me.

I was able to configure the MQTT broker and could get messagens from the subscriber in my machine.

No I'm trying to implement the python script, but it seems is not work as expected as I can't get messages through the script. I'm running this through the terminal.

To perform tests I've inserted some *print* statements outside the *def* functions and I was able to see this messages on the terminal. However apparently all the print statements inside *def* functions are not touched by the script.

it seems that a lot of people could use the script successfully so I dont know what I'm doing wrong. Could you help me please?

Thanks a lot

flaska9999
Question 1 year ago

Hello, I have a little problem. I'm sorry for my bad English.
I completed 6 steps. In step 7 I modified the Python script with brackets. Then I started the script with Python 3.x. **Script started without errors but nothing happens**, I see only first line with RESTART and lots of equals and file location. Mosquitto works fine and receiving messages from ESP.

1

SR_Hashemi
1 year ago

Thanks a lot for sharing this tutorial. It is beneficial. But when I try to make it, a straightforward bug annoying me to get the correct answer. I think some of the users above also scrimmage with this bug. But finally, I successfully found it.
After all, scripts ran correctly, no message printed in the terminal. The reason is \`on_connect\` method is incorrect. This method MUST have four args, but in the above python code, just three passed. So the correct version of it is like this:
def on_connect(client, userdata, flags, rc).

Thank you very much.

PythonMQTT_Subscribe_NEW_VER.py

Srijal97
1 year ago

This was helpful, thanks!

JackWork
1 year ago

the test on Arduino working good but ...I only make the modify to log in parameter in F02IQZQIZ6CAGZ9.py file , in my case usermqtt=username, passwordmqtt=password , hi i have a problem at line 19 from Terminal ,:

```
Traceback (most recent call last):
File "F02IQZQIZ6CAGZ9.py", line 19, in
username# to the broker, and what happens then the topic receives a message
NameError: name 'username' is not defined
```