MQTT on the Raspberry Pi

A short intro

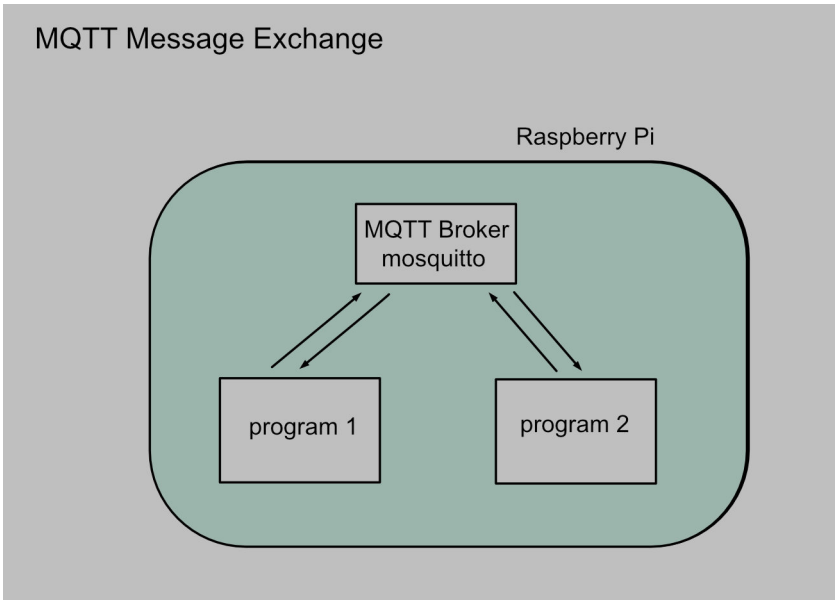
Version2.1

1. Setup MQTT on the Raspberry Pi

Read the docu **mqtt_setup.pdf** in this folder. Note: The code has been **updated** to cover version 2.x of mosquitto on the Pi. This version has enhanced security features (acl-file).

2. MQTT based communication

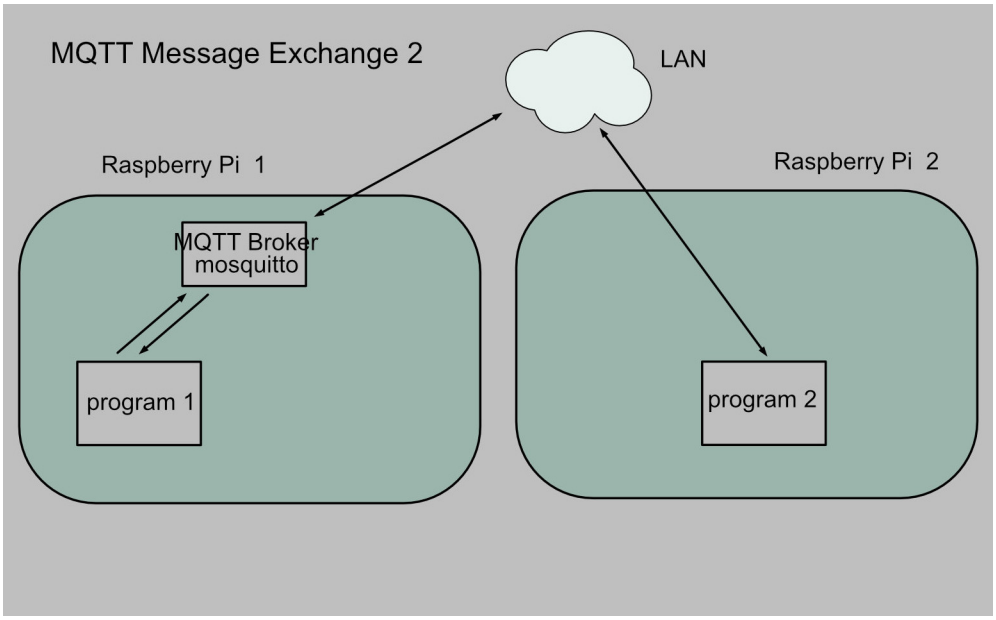
Here is a schematic on how Interprocess communication works using a MQTT broker with the PUB/SUB communication model. We look at an example using a Raspberry Pi.
Components are: MQTT broker, program 1 and program 2. These two programs will exchange messages.



See YouTube video
MQTT on Wikipedia

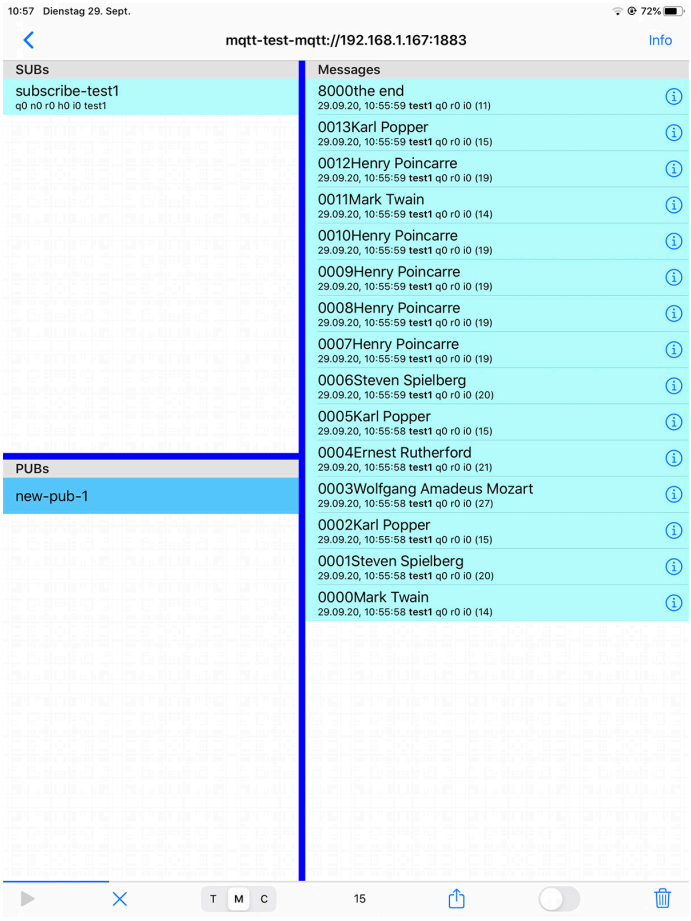
In the above schematic the broker and all parties are on the same machine: a Raspberry Pi. **The beauty of this MQTT thing is this**: program2 could easily run on another Raspberry Pi - with no change **at all**. Program2 simply needs to know the ip-address of the machine that runs the broker (so it cann connect to this broker).
If program1 and program2 and the broker run on the same machine they simply use the local ip-address 127.0.0.0.

This is the reason MQTT is used throughout the IOT world.
[Video on YouTube]



And not only that: on the **iPad or iPhone** one could have an app such as **MQTT Inspector** (similar programs also available for Windows or Mac Systems) and that app is able to inspect all traffic the broker handles.
It simply needs to connect to the broker (its ip-address is needed) and you can enter subscriptions. It then subscribes to the specified topics. I use this often during testing. The iPad app shows me all the messages program1 and prgramm2 publish. I can also inject message into the stream.

Thats very cool and looks like this on my iPad Pro



MQTT Inspector app on the iPad connected to the broker

3. This distribution

There are two ways to get the files in my GitHUb repository python_stuff.

- On your PC or MAC: point your broser to https://github.com/dakota127/python_stuff and klick th green Code Button und download the zip file. After download transfer the folder python_stuff to the home folder of user pi.
- Use a terminal window and login to your pi with user pi. In the users home dir execute this command on the commandline:

You wil end up finding a folder **python_stuff** on your pi.

- mosquitto_config files for user mosquitto config, see docu setup mqtt.
- sub Python class files
- documentation this and other pdf's

4. Running the demo programs

Install Paho

Install Paho on the pi
install this for Python3 by using pip3, For Python 2 use pip2

```
pip3 install paho-mqtt
```

This is needed to run mqtt programs on the Pi.

Two programs come with this demo. Also included are new versions of the classes MyPrint and ConfigRead.

To put these programs onto the pi proceed as follows:

- Open a Filetransfer-Session to the pi, create a folder **mqtt** within the user pi folder and copy **everything** from the **src** folder into this newly created mqtt folder. Make sure the sub folder is also copied to the mqtt folder.
- Open two terminal windows (use putty on Windos machine), login to the pi and navigate to the new directory. You should see these two programs.
- Run pgm_sub2.py in one of the terminal window
- Run pgm_pub2.py in the other terminal window

There is also a file **mqtt_config.ini.** This is the config file read by the two programs (with the help of the ConfigRead class). This file contains user-id and password used for the connection to the broker. User-Id is set to mqtt_demo and the password is year2020. If you run the broker without authentication you do not need to change either the programs nor the config file. The broker simply lets everybody connect, since **allow_anonymous true** is set in the original mosquitto config file. Only the user config file mentioned above will specify **allow_anonymous false**.

I run the programs under Python 3 (Python 2 is simply to old for me now and I do not bother with it anymore). However, they should run under Python 2. I have not tested this, however.

- pgm_sub21.py: this program subscribes to the topic **test1** and will therefore receive messages from the broker if available. The program also publishes messages with the topic **test2.** End the program with Control-C on the keyboard.
- pgm_pub2.py: This program publishes messages with the topic test1. It sends a random number of messages, waits for a while and the repeats this forever. It also subscribes to the broker for messages with topic **test2**. End the program with Control-C on the keyboard.

Both programs can be started with commandline Parms as follows

- -d small debug output
- -D more debug output
- -A even more debug output
- -r repeat endlessly in case of no connection to broker (try again)
- -i to give an IP-address of the machine the broker is running. If this parm is omitted, the programs assume that the broker runs on the local machine

Run these program as follows:

```
python3 pgm_sub2.py

python3 pgm_sub2.py -D
```

4.1 Further Tests

Try also these scenarios which demonstrate the robustness of the implementation.

- stop the broker (command see above) and the start one or both programs, Watch what they do, also use commandline parm -r.
- run the programs and fom another terminal stop the broker for two minutes, then start it again.
- set a false userid in the config.ini file and start the programs

5. Comments on the programs

The two programs work well and they both use three class definitions (located in folder sub)

- Class **myPrint** replaces the print() statement, includes logging to a logfile
- Class **ConfigRead** to read from config files
- Class **MQTTCon** does everything MQTT

In addition, these programs show other good practices in programming Python

- reading of commandline argument
- graceful termination if control-c is pressed on the keyboard
- using a main-clause (if __name__ == "__main__":). Python programs are better structured if one uses this.
-

5.1 Using an MQTT Inspector to control these programs

If you have a tablet or computer running an app that allows you to inspect and inject messages, you can connect to the broker on the pi and send messages with the topic commands.
Check the code for the subscription the this topic.

6. Node Red

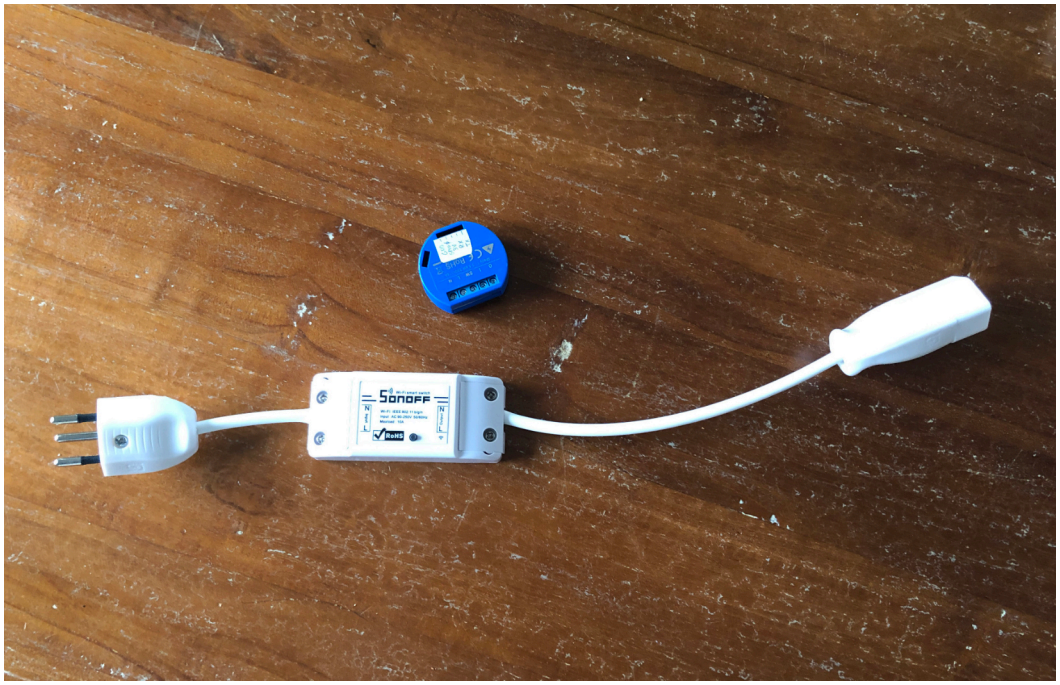
A similar working solution could have been done with NODE-RED.
If working with the Pi in IOT situations is what you like to do, then you are well advised to study RED-Node. It runs on the Pi and there is plenty of info on the net. Here are two good videos on the subject:

Node Red 1
Node Red 2**SSS**

7. Clients on other devices

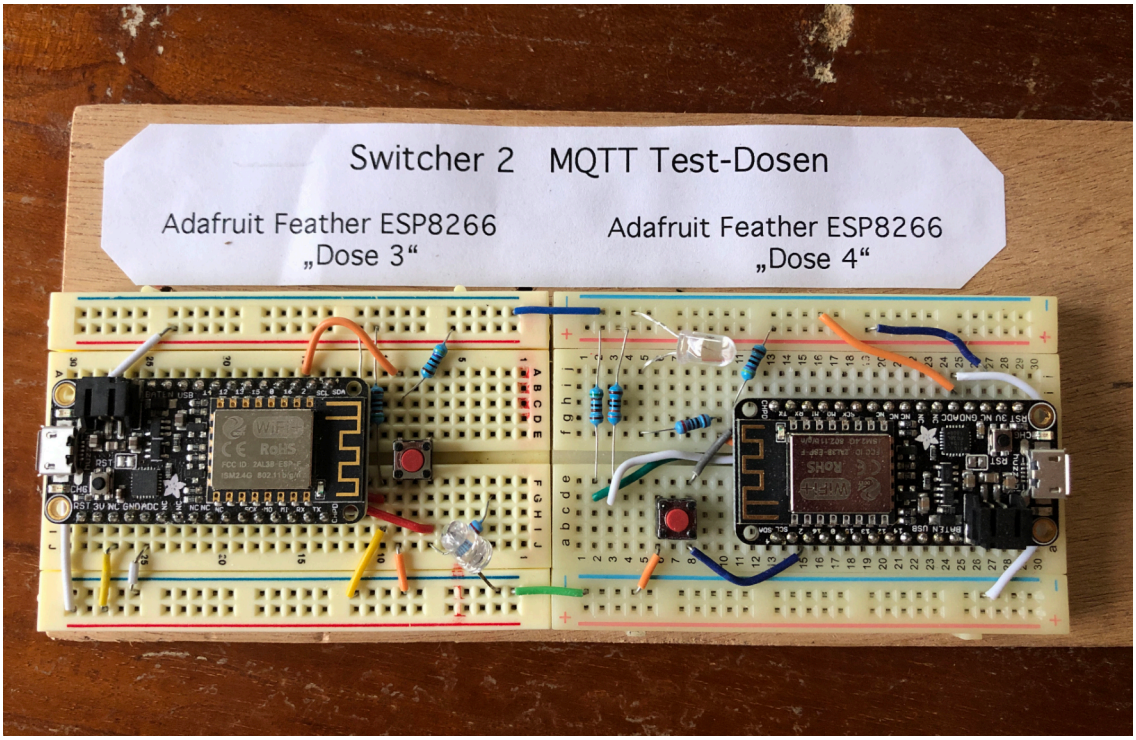
MQTT clients can be run on many microprocessors that have wifi capability, such as ESP8266 or the more modern ESP32. For home automation MQTT-aware smart switches are on the market. These can remotely switch appliances on or off.
I the past I tried these two: the SONOFF smart switch (bottom, with plugs attached) and the tiny Shelly 1 (top). The connect to the WLAN and can subscribe to an MQTT broker on a Pi. The also can send

messages to the broker about the state of the switch.



Shelly 1 (top) and Sonoff (bottom)

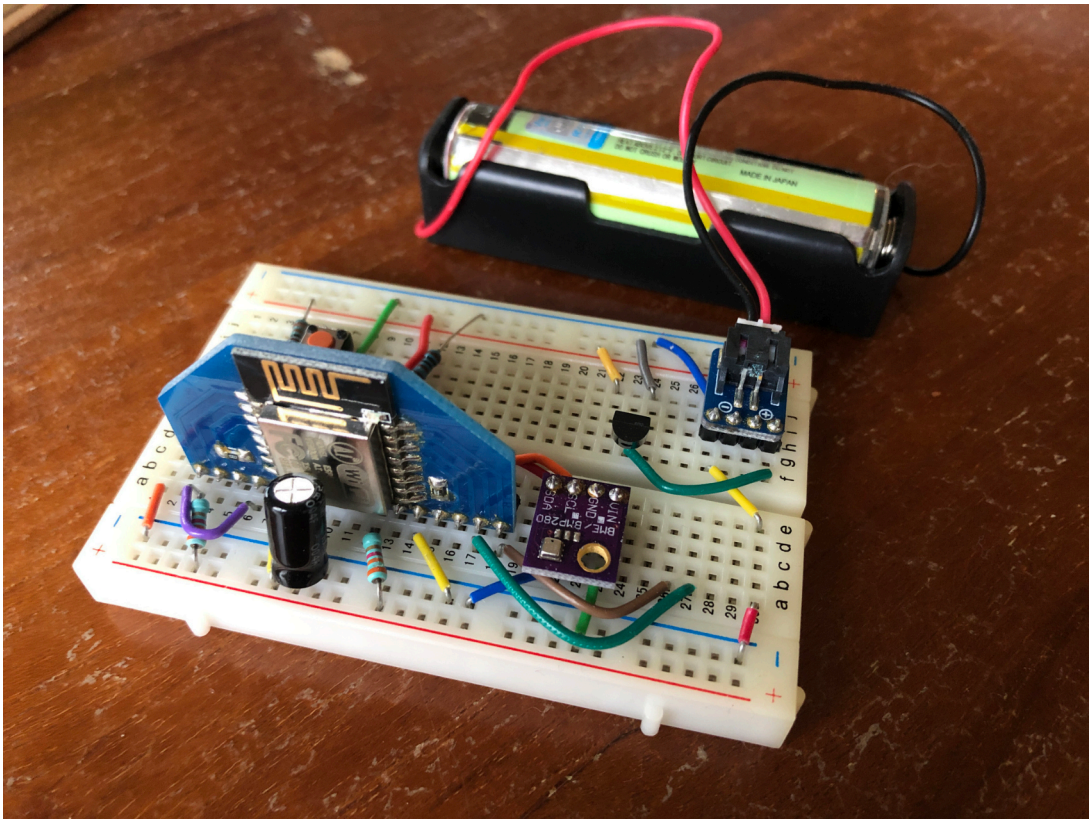
I also build my own 'switches' with ESP8266 that for testing simply switch on or off a led. A relay could be added. But this was before these smart switches appeared on the market.



DIY smart switches using an ESP8266 for each

This is a temperatur sensor together with a ESP8266 on a breadboard. It function as follows:

- Program runs in a endless loop: wakes up from deep sleep, reads sensor, connects to the WLAN, connects to the MQTT broker on a Pi, sends message and goes back to deep sleep for 15 minutes.
- Waking up and doing its thing does only take 2 seconds, then it goes to deep sleep for 15 minutes. It uses very litte battery power.



remote Temperature sensor with ESP8266

I recommend this for further reading

RandomNerdTutorials

About the ESP32

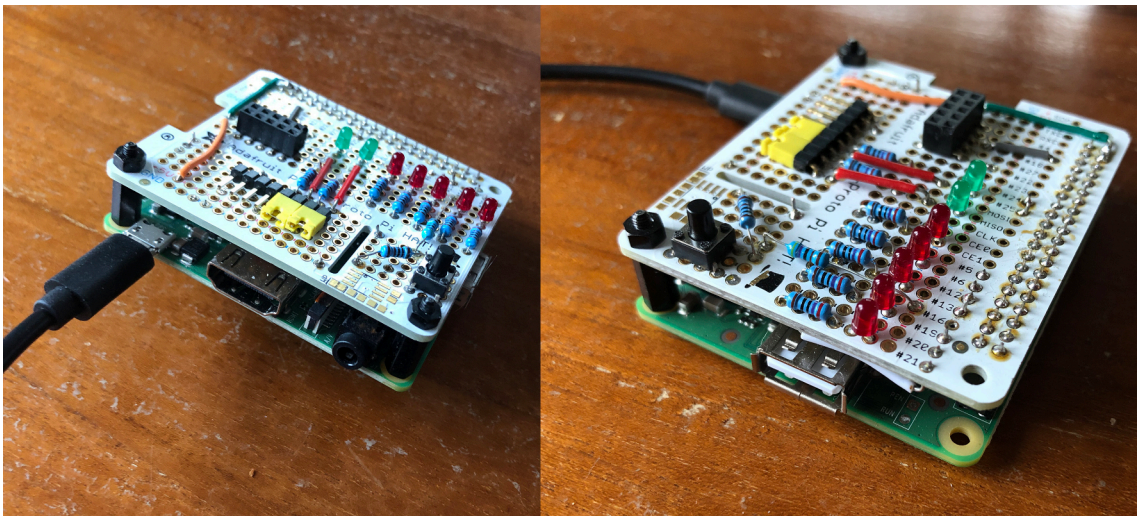
About ESP8266

MQTT Essentials

8. What Type of Raspberry to use

I often use the **Raspberry Pi 3 Model A+** for my projects. I uses less power and I do not need 4 USB port and also no network port. I **never** use the Pi 4 since it consumes a lot of power and I don't need the performance.
Model 3 A+

Here is a test-setup that I often use: a Pi model 3 A+ with a small board with a couple of leds and a switch. The board is from Adafruit and it is the perfect choice for this.
Proto Hat Adafruit



Test Setup Raspberry 3 Model A+

I also use the Pi Zero which is impressive.

Check out this video for a setup with MQTT and SQL Lite on a Pi zero. It is amazing what this litte machine can do.
Andreas Spiess Pi Zero

Also checkout my webpages:

Foto Galleries

Projects Page

YouTube Channel

October 2020, Peter K. Boxler