Assignment MQTT

ITT

Sam Auffenberg - io24m501

Contents

[Contents 2](#_Toc196312007)

[Mosquitto MQTT Broker 3](#_Toc196312008)

[Setup 3](#_Toc196312009)

[OS installation on Raspberry Pi 3](#_Toc196312010)

[Connect to RPI using ssh or serial 5](#_Toc196312011)

[Mosquitto MQTT Broker installation 5](#_Toc196312012)

[Setup avahi deamon 6](#_Toc196312013)

[Testing 7](#_Toc196312014)

[MQTT Client (Sensor) 8](#_Toc196312015)

[Setup 8](#_Toc196312016)

[Code Explanation 9](#_Toc196312017)

[Testing 10](#_Toc196312018)

[MQTT Client (Receiver) 11](#_Toc196312019)

[Setup 11](#_Toc196312020)

[Code explanation 11](#_Toc196312021)

[Table of Figures 13](#_Toc196312022)

[References 14](#_Toc196312023)

## Mosquitto MQTT Broker

A Raspberry Pi 4B (RPI) will be used to run the Mosquitto MQTT Broker. The setup process will be explained below. First, we will set up a fresh operating system (OS) installation on the RPI, after that we will set up the Mosquitto MQTT Broker. Fater that we can test if our installation was correct.

### Setup

#### OS installation on Raspberry Pi

In order to install a OS on our RPI we will need a empty micro SD-Card (min 8 GB) and the Raspberry Pi Imager Tool.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1 raspberry pi imager tool

We chose Raspberry Pi OS Lite (64-bit).

Following configuration will be set in the Raspberry Pi Imager Tool. The username and password can be chosen arbitrarily. The WiFi configuration needs to be changed to the network configuration of the desired network.

A screenshot of a login form

AI-generated content may be incorrect.

Figure 2 general configuration tab in raspberry pi imager

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3 services configuration tab in raspberry pi imager

#### Connect to RPI using ssh or serial

##### SSH

To connect to the RPI using ssh you need to get the IP address of your Raspberry Pi. You may retrieve this from the configuration page of you WiFi router. Afterwards you can access the device using:

ssh <username>@<IPAddress>

The username is the one chosen in the configuration above.

##### Serial

To connect to the RPI via Serial interface you need to plug in the RPIs SD-card back to your computer and edit the “config.txt” file. You will need to add the following line to the file:

enable\_uart=1

Save the file, disconnect the SD-card and put it back into the raspberry-pi and reboot it.

You can now use a TTL-to-USB adapter to access the RPIs serial interface on TH GPIO Pins. Connect the TTL\_to\_USB adapter to pins 6 (GND), 8 (TX) and 10 (RX). You can use a sowarte like putto to access the device now.

#### Mosquitto MQTT Broker installation

In order to install the Mosquitto MQTT Broker on our RPI we will run following commands:

To update all installed packages to their newest versions:

sudo apt update && sudo apt upgrade

To install the actual Mosquitto MQTT Broker software.

sudo apt install -y mosquito mosquito-clients

To add the MQTT Broker as a service to the auto start of the device:

sudo systemctl enable mosquitto.service

To ensure a secure connection and access only by authorized clients we want to setup a username and password. First we need to create a password file. Replace “username” with your desired username. You will then be asked to input a password:

sudo mosquitto\_passwd -c /etc/mosquitto/passwd username

In order to enable remote access we need to edit the mosquito.conf. A listener port will be defined and the password file will be added to the configuration:

sudo nano /etc/mosquitto/mosquitto.conf

Add the following lines:

# Enable authentication   
allow\_anonymous false   
password\_file /etc/mosquitto/passwd   
  
# Configure standard MQTT port   
listener 1883

After the install we can reboot the system using

sudo reboot

#### Setup avahi deamon

As the IP address of our MQTT Broker may change we want to be able to access it using a name instead of an IP address when trying to connect to it.   
In order to do so we will install a local DNS or mDNS service calles Avahi.  
Runn following commands to set it up:

Installation of avahi:

sudo apt update

sudo apt install avahi-daemon

Change hostname to the one desired, you can also use the hostname given in the setup configuration earlier.:

sudo raspi-config -> “System Options” -> “Hostname”

You can now access device ssh or the mosquito MQTT broker using:

ssh [user@hostname.local](mailto:user@hostname.local)

### Testing

In order to test the MQTT Broker we need to install the Mosquitto MQTT Tools on a separate Computer or device. In this case we installed Mosquitto on a Windows PC in the same network as the RPI.

Use a command line tool and go to the folder of you mosquito installation. On Windows usually:

c:\Program Files (x86)\mosquitto>

User following command to subscribe to a test topic on our MQTT Broker.   
You can use -h “hostname.local” or the IP address of the RaspberryPi. -t “test” is the test topic. Username and password are the ones chosen earlier when setting up the Mosquitto MQTT Broker.

mosquitto\_sub.exe -h hostname.local -t test -u "username" -P "password"

This command will run a MQTT subscriber waiting for data to be published:

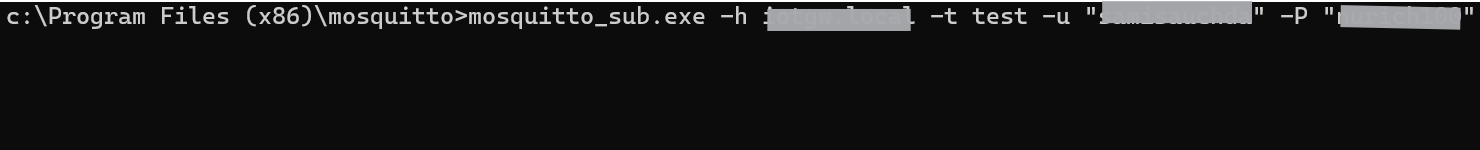


Figure 4 command line running mosquitto subscriber on windows

In a second command line (also inside the mosquito installation folder) we will run the publish command using the same settings. -m defines the message to be send:

mosquitto\_pub.exe -h hostname.local -t test -m “Test” -u "username" -P "password"

This will publish a message to the MQTT Broker. The subscriper will receive the message

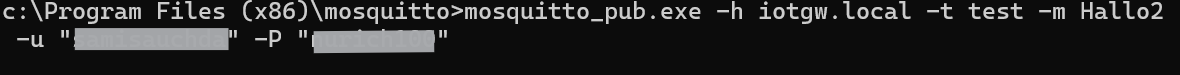


Figure 5 message "Hallo2" published on MQTT Broker

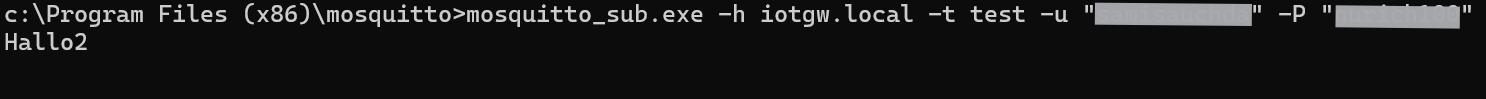


Figure 6 Subscriber received the test message

## MQTT Client (Sensor)

### Setup

As our MQTT Sensor we will use a ESP32-S3-DevKitC-1 (Figure 7) and a DHT11 humidity and temperature sensor (Figure 8). We will connect the DHT11 VCC pin to a 3V3 pin on the ESP32, the GND pin of the sensor to any ground pin on the SP32 and the Data pin of the DHT11 to GPIO41 on the ESP32.

A diagram of a computer chip

AI-generated content may be incorrect.

Figure 7 ESP32-S3 Pinout [1]



Figure 8 DHT11 Sensor pinout [2]

The software we want to deploy on the ESP32 can be downloaded from the github repository [[here]](https://github.com/samisauchda/TechnikumITT_DHT_MQTT/tree/master) and is also part of the task submission.

Download the code and add it to a project with PlattformIO. This will handle the download of all libraries needed. You can see the opened code in Figure 9.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 9 ESP32 softweare opened in PlattformIO on VS Code

Using PlattformIO you can upload the code to the ESP32. A Instruction on how to use PlatformIO will not be included here.

### Code Explanation

The software uses the WifiManager Library [[here]](https://github.com/tzapu/WiFiManager). Upon boot the ESP32 will open up a access point. You can connect to this access point. When connected you will be directed to a captive portal (see Figure 10). Here you can enter the WiFi SSID and password of the network you cant the sensor to connect to. In addition you can enter all needed MQTT parameters for our MQTT Broker. The default values for the MQTT settings are:

mqtt\_broker = "iotgw.local"   
mqtt\_port = "1883”   
mqtt\_client\_id = "DHTSensor"   
mqtt\_username = "test\_user"   
mqtt\_password = "test\_password"   
mqtt\_topic\_temp = "DHTSensor/temperature"   
mqtt\_topic\_humid = "DHTSensor/humidity"

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 10 WiFiManager portal for SSID password and MQTT parameters

After clicking save theESP32 will connect to your WiFi Network and start publishing data to the MQTT Broker.  
If you entered wrong credentials or MQTT parameters you might have to erase the ESP32s flash and re upload the program using PlattformIO.

The code uses the MQTT Library [[here]](https://github.com/256dpi/arduino-mqtt) to publish the Sensor data to the MQTT Broker. The data is published as a string and I chose to publish temperature and humidity to separate topics.

The DHT Library [[here]](https://github.com/adafruit/DHT-sensor-library) is used to read the sensor data.

Upon boot the ESP will try to connect to WiFi. It will either connect to the given credentials or open up its own AP as seen in the section above. After that it will try to connect to the MQTT Broker.

In the main loop it will check for a WiFi connection and a connection to the MQTT server approximately every 100ms. Every 10 seconds the data from the sensor will be read and published to the MQTT broker.

### Testing

Set up the client by uploading the code to the Esp32, connecting the sensor and configuring the WiFi credentials and MQTT parameters. Afterwards you can test if the MQTT Broker receives the messages by using Mosquitto subscriber on a different computer as shown in an earlier chapter.

## MQTT Client (Receiver)

### Setup

The MQTT Client (Receiver) is a python script that can be run on any Linux or windows machine in the same network as the MQTT broker. It subscribes to the two topics of the sensor and saves the receives data to a .csv file.  
The code can be found in the submission and on the github repository [[here]](https://github.com/samisauchda/TechnikumITT_DHT_MQTT/tree/master).

To run the script you need to install the paho-mqtt package. This can be installed using pip install paho-mqtt.

### Code explanation

The code first starts with the import of the libraries needed:  
A black rectangle with white text

AI-generated content may be incorrect.

Then follow variables for configuration of the MQTT connection and the output file:  
A screen shot of a computer

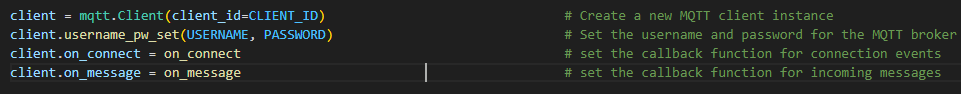
AI-generated content may be incorrect.

A function to handle MQTT connections:  
A screen shot of a computer code

AI-generated content may be incorrect.

A function to handle incoming MQTT messages. This function receives the data and saves it to .csv:  
A computer screen shot of text

AI-generated content may be incorrect.

This part creates a new MQTT client instance. It also assigns the callback functions:  


And finally the code part that starts the actual mqtt connection and loop that keeps the python file running:  
A computer screen with text on it

AI-generated content may be incorrect.

### Testing

Test the client by running the python script:

python mqttCSVexport.py

The console output should look something like this:

A computer screen with white text

AI-generated content may be incorrect.

# Table of Figures

[Figure 1 raspberry pi imager tool 3](#_Toc196312024)

[Figure 2 general configuration tab in raspberry pi imager 4](#_Toc196312025)

[Figure 3 services configuration tab in raspberry pi imager 4](#_Toc196312026)

[Figure 4 command line running mosquitto subscriber on windows 7](#_Toc196312027)

[Figure 5 message "Hallo2" published on MQTT Broker 7](#_Toc196312028)

[Figure 6 Subscriber received the test message 7](#_Toc196312029)

[Figure 7 ESP32-S3 Pinout [1] 8](#_Toc196312030)

[Figure 8 DHT11 Sensor pinout [2] 8](#_Toc196312031)

[Figure 9 ESP32 softweare opened in PlattformIO on VS Code 9](#_Toc196312032)

[Figure 10 WiFiManager portal for SSID password and MQTT parameters 10](#_Toc196312033)

# References

|  |  |
| --- | --- |
| [1] | Espressif, 2025. [Online]. Available: https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2024/09/ESP32-S3-pinout.jpg?quality=100&strip=all&ssl=1. |
| [2] | Components101. [Online]. Available: https://components101.com/sensors/dht11-temperature-sensor. |