Chapitre 1

Communication entre deux Raspberry pi moyennant le protocole MQTT

1.1 Introduction

Le MQTT est un protocole qui envoie spécifiquement des données à partir de dispositifs de l'Internet des objets et est pris en charge par la plupart des microcontrôleurs et des systèmes. Pour utiliser la communication MQTT Raspberry Pi, peu de choses sont nécessaires, c'est pourquoi ce type de transmission est très intéressant. Dans ce projet, nous installons un courtier Raspberry Pi MQTT, à la suite duquel nous recevons et envoyons des données. Nous pouvons soit utiliser plusieurs Raspberry Pi pour les tests, soit utiliser un seul appareil.

1.2 Qu'est-ce que MQTT ?

Selon la page officielle de MQTT :
"MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium."
Il s'agit donc d'un protocole de réseau standardisé avec lequel des messages/commandes courts peuvent être transmis. Le grand avantage est que les adaptateurs wifi intégrés (par exemple dans le Raspberry Pi) sont utilisés pour la connexion internet. Plus d'accessoires et de câblage complexes ne sont pas nécessaires ! Il est ainsi possible d'envoyer les données via le réseau local ou l'internet.
En détail, le transfert se compose de trois éléments différents :

- **L'éditeur :** envoie des messages.
- **Courtier :** Transmet les messages aux abonnés enregistrés.
- **Abonné :** Reçoit les messages par l'intermédiaire du courtier.

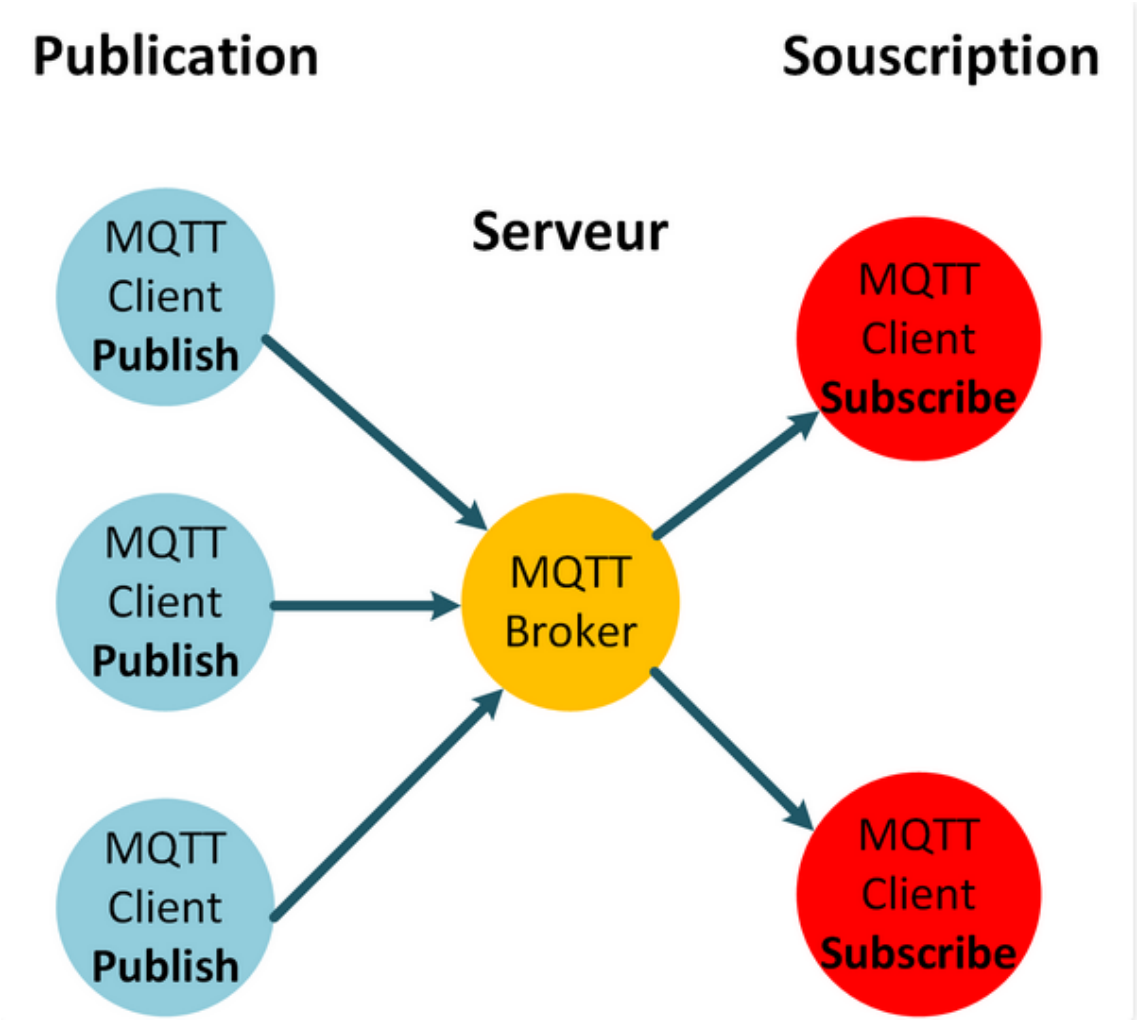

FIGURE 1.1 – Application du protocole MQTT

1.3 Serveur MQTT Raspberry Pi – Mosquitto

Il existe plusieurs applications qui peuvent être utilisées pour envoyer et recevoir par le biais du MQTT, mais la plus simple sur le Raspberry Pi est probablement Mosquitto. Nous allons d'abord l'installer sur les deux Raspberry Pi.
sudo apt-get install -y mosquitto mosquitto-clients

Après l'installation, un serveur Mosquitto est lancé automatiquement. On ouvre un abonné dans le canal « test_channel » qui attend des messages.
mosquitto_sub -h localhost -v -t test_channel
Le canal est ici comme une fréquence, sur laquelle on écoute. Par exemple, différentes données peuvent être envoyées sur différents canaux (par exemple, la température, l'humidité, la luminosité, etc.).

Afin de transférer simplement les données, on peut soit utiliser le même Raspberry

Pi (ouvrir un nouveau terminal / connexion SSH), soit envoyer les données à partir d'un autre Pi. Si nous utilisons le même Raspberry Pi, l'utilisation est facilement possible. Pour cela, nous envoyons simplement un message test (en tant qu'éditeur) dans le même canal.
mosquitto_pub -h localhost -t test_channel -m "Hello Raspberry Pi"
Sinon, on spécifie l'adresse IP interne (par exemple 192.168.1.5) du destinataire au lieu de « localhost »(dans la ligne de commande). Du côté du destinataire, le message doit apparaître.

1.4 Raspberry Pi MQTT échange de données avec Python

Pour que nous puissions utiliser le tout à partir de scripts, nous voulons le mettre à la disposition de Python. Pour cela, nous installons d'abord une bibliothèque via le gestionnaire de paquets Python (pour Python3, on peut utiliser également pip3) :
sudo pip3 install paho-mqtt
Nous commençons par le récepteur. Pour cela, nous créons un nouveau fichier avec le contenu :

sudo nano mqtt_subscriber.py

```
1
2 import paho.mqtt.client as mqtt
3
4 MQTT_SERVER = "localhost"
5 MQTT_PATH = "test_channel"
6
7 # The callback for when the client receives a CONNACK response from the
    server.
8 def on_connect(client, userdata, flags, rc):
9     print("Connected with result code "+str(rc))
10
11     # Subscribing in on_connect() means that if we lose the connection and
12     # reconnect then subscriptions will be renewed.
13     client.subscribe(MQTT_PATH)
14
15 # The callback for when a PUBLISH message is received from the server.
16 def on_message(client, userdata, msg):
17     print(msg.topic+" "+str(msg.payload))
18     # more callbacks, etc
19
20 client = mqtt.Client()
21 client.on_connect = on_connect
22 client.on_message = on_message
23
24 client.connect(MQTT_SERVER, 1883, 60)
25
26 # Blocking call that processes network traffic, dispatches callbacks and
27 # handles reconnecting.
28 # Other loop*() functions are available that give a threaded interface and
    a
```

```
29 # manual interface.
30 client.loop_forever()
```

Après avoir enregistré le fichier, nous l'exécutons :
sudo python mqtt_subscriber.py
On remarque que l'abonné est en attente pour recevoir une donnée.
Le code de l'éditeur ressemble à ceci (on crée un nouveau fichier et on l'exécute, en gardant à l'esprit l'adresse IP correcte) :

```
1 import paho.mqtt.publish as publish
2
3 MQTT_SERVER = "192.168.1.5"
4 MQTT_PATH = "test_channel"
5
6 publish.single(MQTT_PATH, "Hello World!", hostname=MQTT_SERVER)
```

import paho.mqtt.publish as publish
MQTT_SERVER = "192.168.1.5" MQTT_PATH = "test_channel"
publish.single(MQTT_PATH, "Hello World!", hostname=MQTT_SERVER)