CSCi 4907
Introduction to IoT and Edge Computing Applications

Prof. Kartik Bulusu, CS Dept.

Week 11 [04/11/2024]	• Recorded Guest lecture by **Chris Rodley,** Chris Rodley - CEO and Founder Snap Information Technologies Ltd www.snapcore.co	• MQTT using Paho-MQTT • In-class Raspberry Pi Lab – Publish and Subscribe Messages using Paho MQTT

```
git clone git@github.com:gwu-csci3907/Spring2024.git

git clone https://github.com/gwu-csci3907/Spring2024.git
```

GW School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Spring 2024

Photo: Kartik Bulusu

Explore MQTT Basics
Message Queuing Telemetry Transport

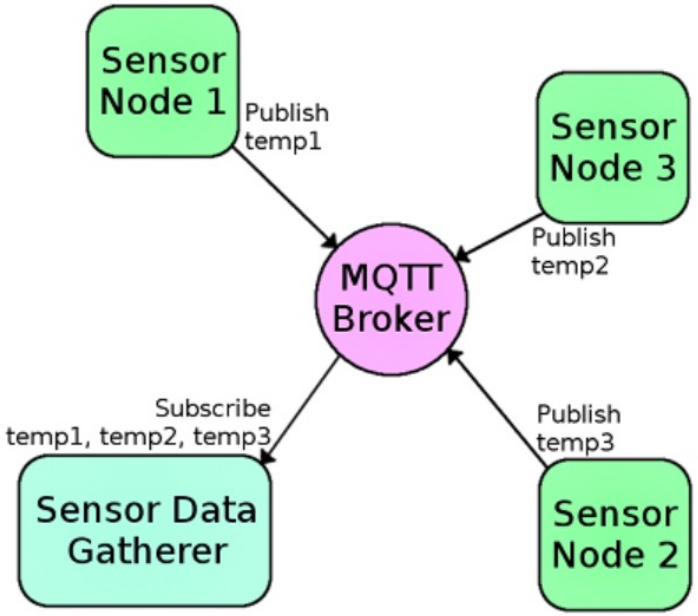
Goal: To understand how publishing and subscribing works practically

School of Engineering
& Applied Science
THE GEORGE WASHINGTON UNIVERSITY

GW

Prof. Kartik Bulusu, CS Dept.

MQTT paradigm



Hardware

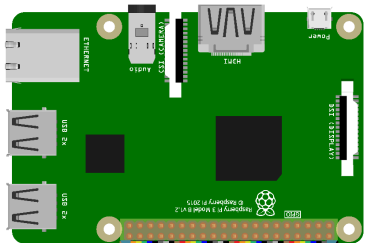
Broker

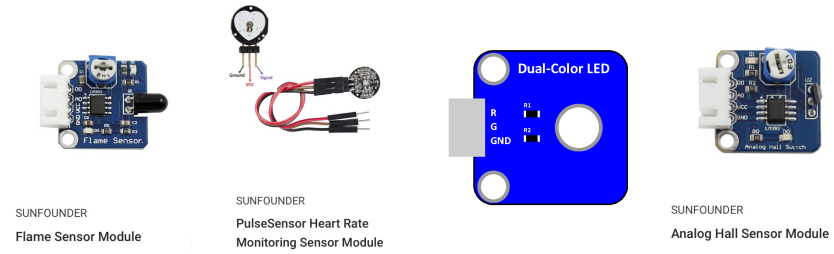
- The broker is the server
- It distributes the information to the interested devices connected to the server.

Client

- The device that connects to broker to send or receive information.

Source: https://learn.sparkfun.com/tutorials/introduction-to-mqtt#the-basics





SUNFOUNDER
Flame Sensor Module

SUNFOUNDER
PulseSensor Heart Rate Monitoring Sensor Module

SUNFOUNDER
Analog Hall Sensor Module

Messaging

Topic

- The name that the message is about.
- Clients publish, subscribe, or do both to a topic.

Subscribe

- Clients tell the broker which topic(s) they're interested in.

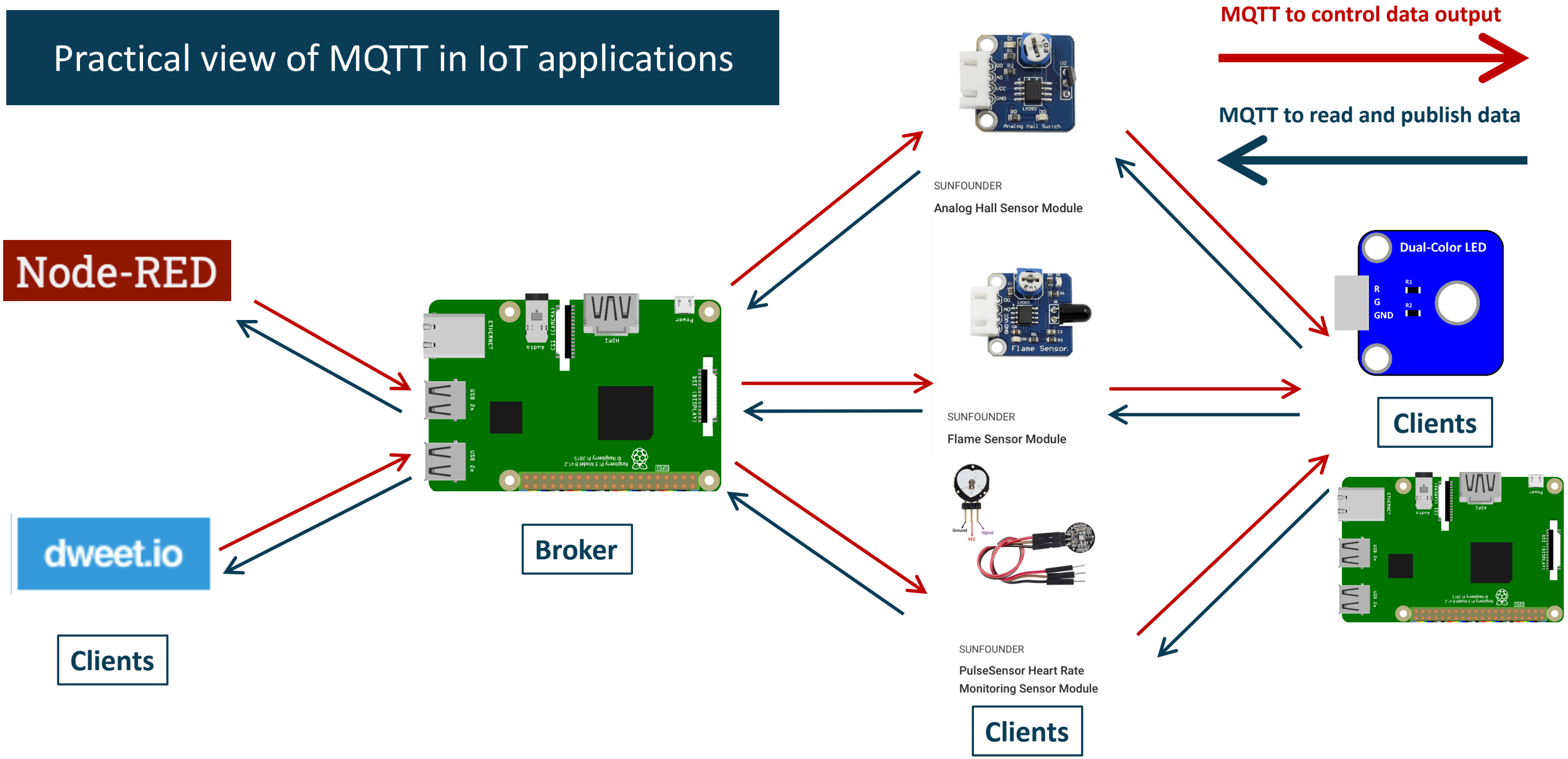
Publish

- Clients that send information to the broker to distribute to interested clients based on the topic name.

QoS

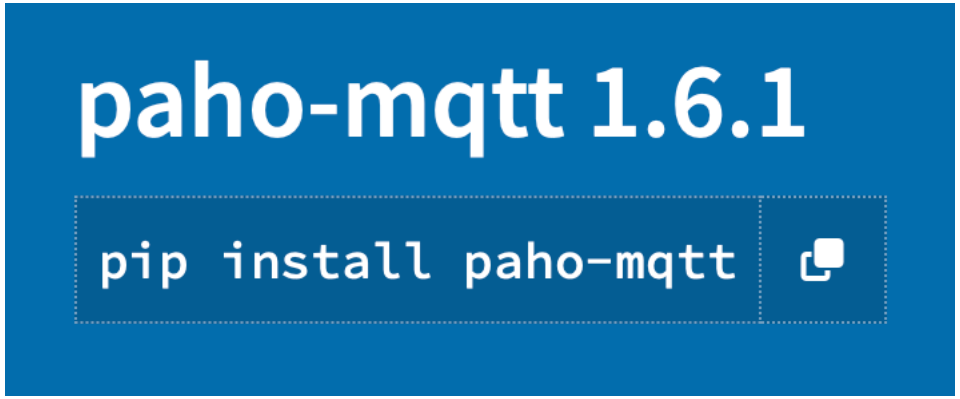
- Quality of Service to the broker
- Integer value ranging from. 0-2.

Practical view of MQTT in IoT applications



Source:
https://www.eclipse.org/paho/
https://pypi.org/project/paho-mqtt/

Eclipse paho - Another open source MQTT broker



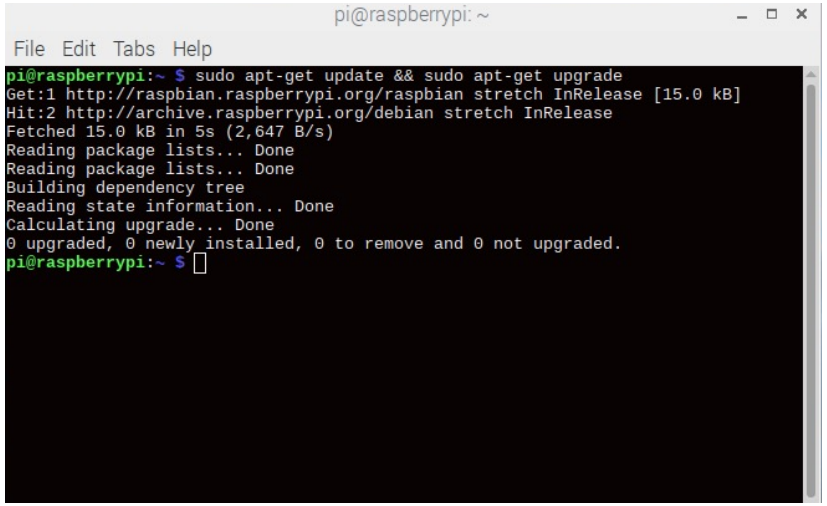
Eclipse-paho provides a client class which enable applications to connect to an MQTT broker to publish messages, and to subscribe to topics and receive published messages.

It also provides some helper functions to make publishing one off messages to an MQTT server very straightforward.

Step-1: Install paho-mqtt & psutil libraries

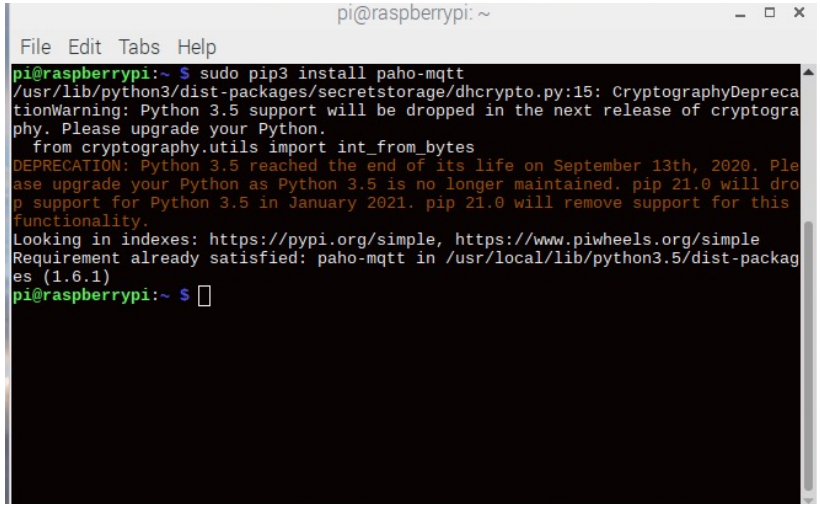
Source:
https://towardsdatascience.com/iot-made-easy-esp-micropython-mqtt-thingspeak-ce05eea27814
https://nothans.com/thingspeak-tutorials/update-a-thingspeak-channel-using-mqtt-on-a-raspberry-pi
https://pypi.org/project/paho-mqtt/
https://pypi.org/project/psutil/

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




```
sudo pip –upgrade install psutil
```

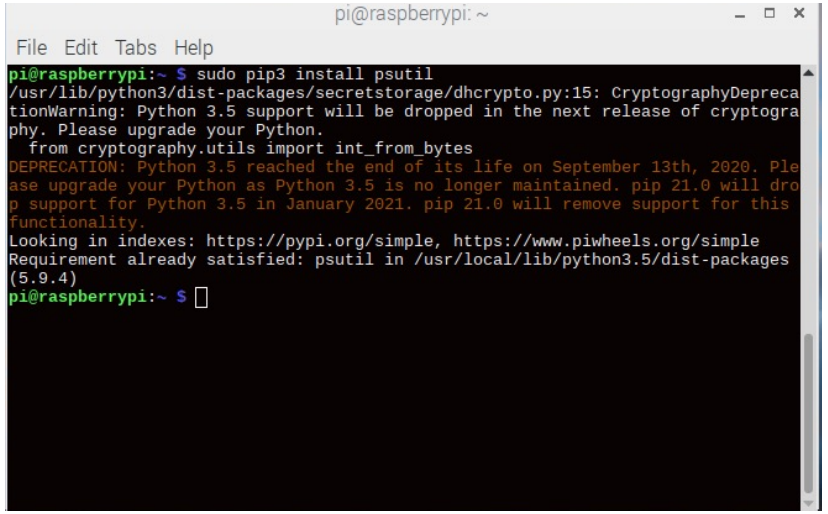
```
sudo pip install paho-mqtt
```



paho-mqtt 2.0.0

```
pip install paho-mqtt
```

```
sudo pip install psutil
```



psutil 5.9.8

```
pip install psutil
```

psutil (process and system utilities) is a cross-platform library for retrieving information on **running processes** and **system utilization** (CPU, memory, disks, network, sensors) in Python

Step-2: Check installations of `paho-mqtt` & `psutil` libraries

```
git clone --depth 1 -b v1.6.1 https://github.com/eclipse/paho.mqtt.python
cd paho.mqtt.python
python3 setup.py install
```

Step-3: git clone the folder titled pahoMQTT_examples

Explore MQTT with Paho
[Graded lab assignment]

Source:
https://www.emqx.com/en/blog/use-mqtt-with-raspberry-pi#raspberry-pi-mqtt-project-preparation

Step-4: Prepare the MQTT broker
Note: we will use the free public MQTT broker at `broker.emqx.io`

```
# test_connect.py
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected success")
    else:
        print(f"Connected fail with code {rc}")

client = mqtt.Client()
client.on_connect = on_connect
client.connect("broker.emqx.io", 1883, 60)
client.loop_forever()
```

The callback function.
It will be triggered when trying to connect to the MQTT broker client is the client instance connected this time

userdata is users' information, usually empty. If it is needed, you can set it through user_data_set function.

flags save the dictionary of broker response flag.

rc is the response code.
Generally, we only need to pay attention to whether the response code is 0.

Source:
https://www.emqx.com/en/blog/use-mqtt-with-raspberry-pi#raspberry-pi-mqtt-project-preparation

Step-4: Prepare the MQTT broker
Note: we will use the free public MQTT broker at `broker.emqx.io`

```
# test_connect.py
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected success")
    else:
        print(f"Connected fail with code {rc}")

client = mqtt.Client()
client.on_connect = on_connect
client.connect("broker.emqx.io", 1883, 60)
client.loop_forever()
```

The callback function.
It will be triggered when trying to connect to the MQTT broker client is the client instance connected this time

userdata is users' information, usually empty. If it is needed, you can set it through user_data_set function.

flags save the dictionary of broker response flag.

rc is the response code.
Generally, we only need to pay attention to whether the response code is 0.

Source:
https://www.emqx.com/en/blog/use-mqtt-with-raspberry-pi#raspberry-pi-mqtt-project-preparation

```
# subscriber.py
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.subscribe("raspberry/topic")

def on_message(client, userdata, msg):
    print(f"{msg.topic} {msg.payload}")

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.will_set('raspberry/status', b'{"status": "Off"}')

client.connect("broker.emqx.io", 1883, 60)

client.loop_forever()
```

Subscribes the topic that gets called during **on_connect**
If reconnect after losing the connection with the broker, it will continue to subscribe to the raspberry/topic topic

The callback function
that will be triggered when receiving messages

When the Raspberry Pi is powered off, or the network is interrupted abnormally, it will send the will message to other clients

Create connection, the three parameters are broker address, broker port number, and keep-alive time respectively

Set the network loop blocking, it will not actively end the program before calling disconnect() or the program crash

Source:
https://www.emqx.com/en/blog/use-mqtt-with-raspberry-pi#raspberry-pi-mqtt-project-preparation

```python
#publisher.py
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")


for i in range(5):
    client.publish('raspberry/topic', payload=i, qos=0, retain=False)
    print(f"send {i} to raspberry/topic")

client = mqtt.Client()
client.on_connect = on_connect
client.connect("broker.emqx.io", 1883, 60)

client.loop_forever()
```