Lab no 04 – MQTT & Raspberry Pi

This lab introduces a simple application on Raspberry Pi using MQTT protocol.
In this Lab, you will implement MQTT protocol and control LED connected to Raspberry Pi.

Parts: -

1. Install Mosquitto broker.
2. Implement security via Access control method in the broker (client username, client password).
3. Install the Paho MQTT client on different nodes (Publisher, Subscriber).
4. Case Study- Control LED.

Required Resources

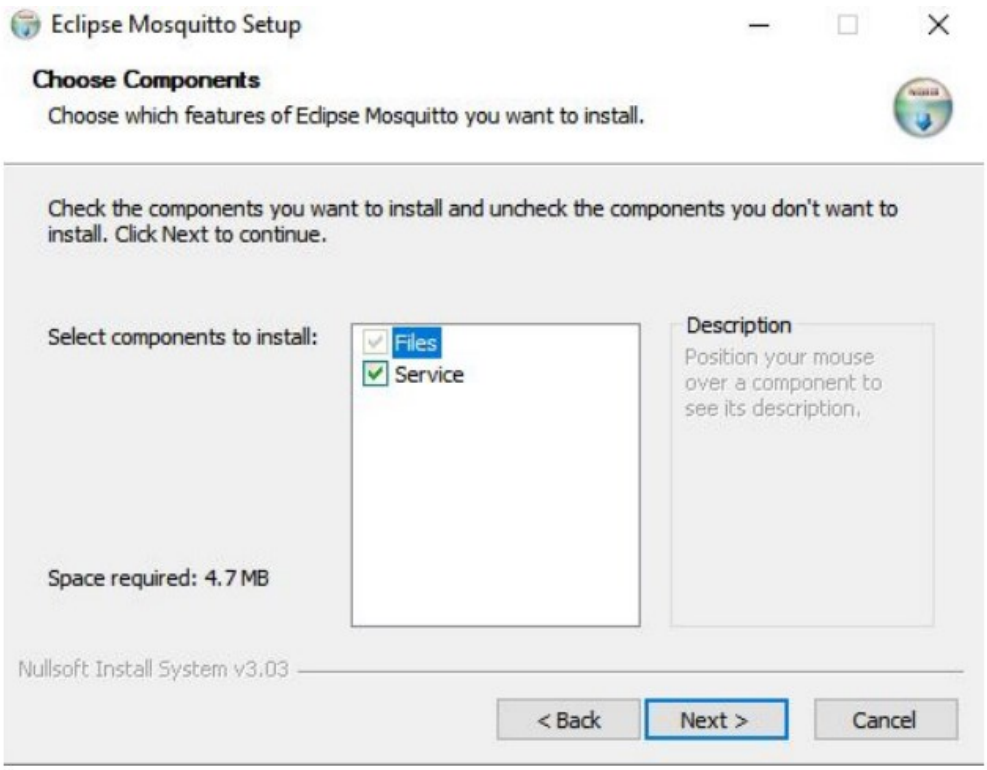
- 2 Raspberry Pi kits.
- SD card.
- LED.
- Resistor 330 ohm.
- Breadboard.
- Jumpers.

Part 1: Install Mosquitto broker.

Mosquitto is an application for the MQTT protocol. It can be used as a broker, subscriber, and publisher. Here is the steps to

- **Download** Eclipse Mosquitto from https://Mosquitto.org/download/
- **Create** new folder “Mosquitto” on director D.
- **Change** directory path to D:/Mosquitto.
- **Install** Eclipse Mosquitto.

Note: If you install it with the “Service” checkbox checked, it will start automatically with Windows and occupy the default port 1883.



- **Create** a text file “test.conf” under the Mosquitto folder (D:/Mosquitto).
- **Copy** the following lines and save the file.

Note these lines to perform an unauthenticated access.

```
Listener 1883                # allows connections on all interfaces.
allow_anonymous true         # allow anyone not authenticated to connect.
```

- **Lunch** Mosquitto broker, **Open** cmd window and write the following commands.

```
D:
Cd Mosquitto
Mosquitto  -v -c test.conf
```

To Publish

- **Open** a second cmd window, and write the following commands

```
D:
Cd Mosquitto
Mosquitto _pub -h localhost -t topic/test -m “hello I am publisher from cmd”
```

To Subscribe

- **Open** third cmd to subscribe and enter the commands

```
D:
Cd Mosquitto
Mosquitto _sub -h localhost -t topic/test
```

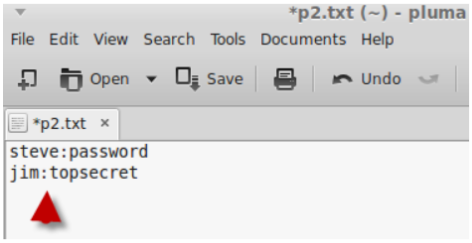
Part 2: Implement Security via Access Control on the Broker side.

The Mosquitto MQTT broker can be configured to perform client authentication. Using a valid username and password to authenticate client before a connection is permitted.

Configure the Mosquitto broker:

- **Create** a password file. **Create** a text file and write the username and passwords.

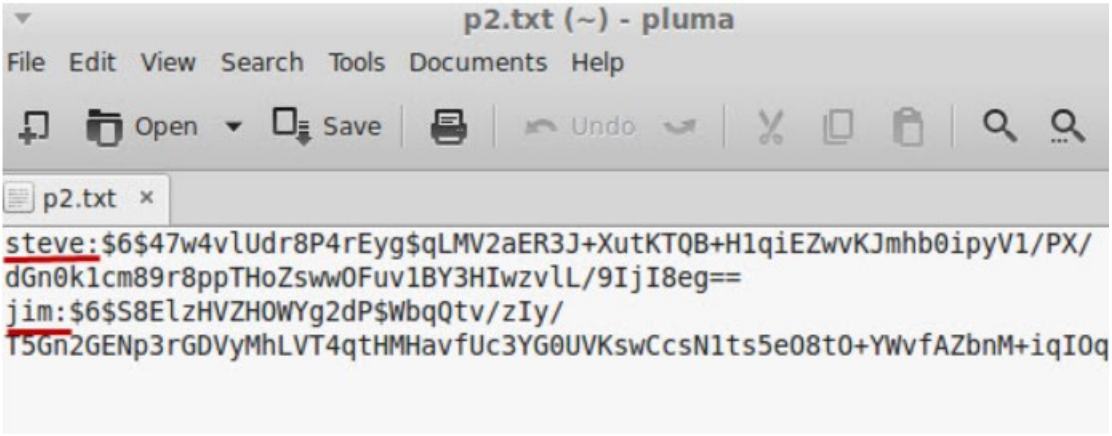
  One username and password per line, the username and password separated by a colon as shown below.



- **Convert** the password file which encrypts the passwords, Go to a command line and type:

```
Mosquitto _passwd -U passwordfile
```

Now if you open the password file again you should see encrypted password, as shown in the below figure.



Force password use.

- **Open** "test.conf" and edit the following commands

```
allow_anonymous false
password_file c:\Mosquitto \passwords.txt   #path of password file
```

Note: Now, publish or subscribe must use username and password, as shown below.

```
D:
Cd Mosquitto
Mosquitto _pub -h localhost -t topic/test -m “hello I am publisher from cmd” -u client1 -P 123456789
Mosquitto _sub -h localhost -t topic/test -u client1 -P 123456789
```

Part 3 Install Paho MQTT client on different nodes (Publisher, Subscriber).

- **Open terminal and write the following commands.**

```
pip install paho-mqtt
cd paho.mqtt.python
python setup.py install
```

Publisher (Raspberry pi - First Node)

Publisher set the following parameters:

```
publish (topic, payload=None, qos=0, retain=False)
```

- **Topic:** the topic that the message should be published on
- **Payload:** the actual message to send
- **QoS:** the quality-of-service level to use (0,1,2)
- **Retain**: if set to True, the message will be set as the "last known good"/retained message for the topic.

```
import paho.mqtt.client as mqtt
import time

mqttBroker = "192.168.1.10"                    #mqtt broker ip
client = mqtt.Client()
client.connect(mqttBroker)                     # mqtt connection
while True:
   client.publish("topic/test", “publish from raspberry pi”)  # publish function
```

\

Subscriber (Raspberry Pi Second Node)

Subscriber set the following parameters:

subscribe(topic, qos=0)

- **Topic**: a string specifying the subscription topic to subscribe to.
- **QoS**: the desired quality of service level for the subscription. Defaults to 0.

```
import paho.mqtt.client as mqtt
import time

def on_message(client, userdata, message):
    print("Received message: ", str(message.payload.decode("utf-8")))

mqttBroker = "192.168.1.10"
client = mqtt.Client()
client.connect(mqttBroker)

client.loop_start()
client.subscribe("topic/test")
client.on_message = on_message      # Called when a message has been received on a
#topic that the client subscribed
```

Part 4 Case study - Control LED

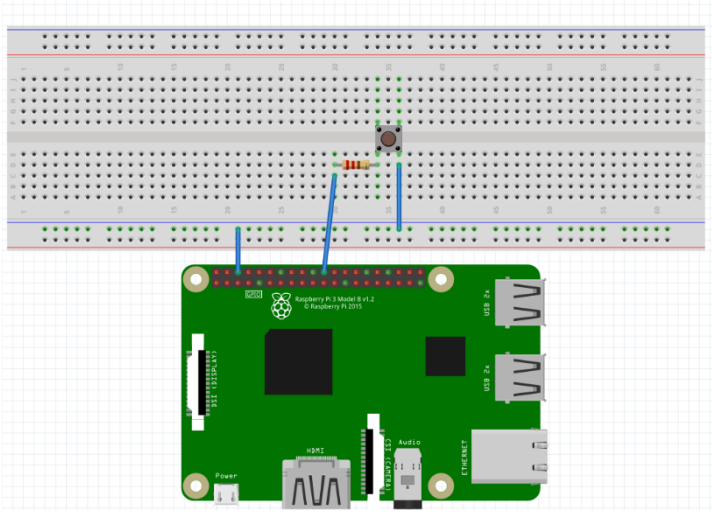
Our Case study is designed as follow:

- Laptop as broker.
- Raspberry Pi Node One as publisher. Where it controls LED by 0 or 1 at topic “/led”
- Raspberry Pi Node Two as subscriber to topic “/led”.

Hardware Connection

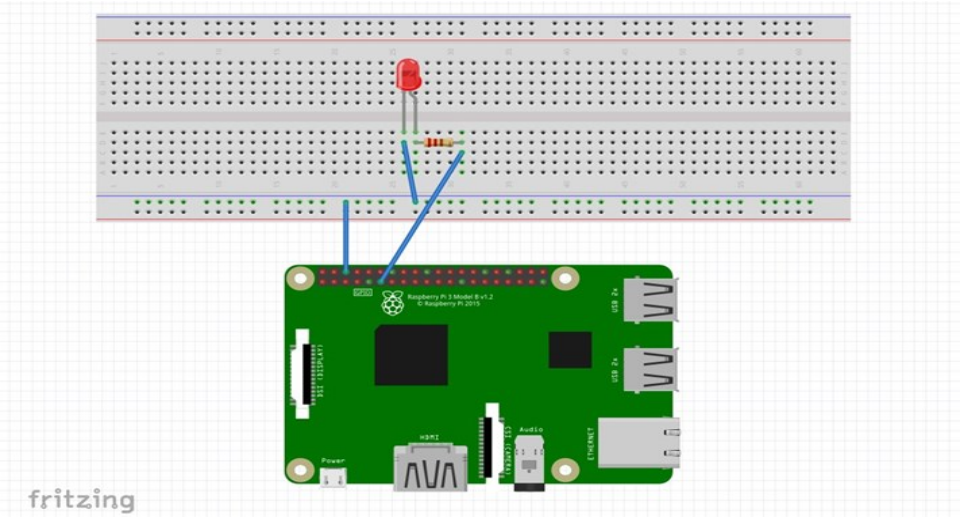
- **Raspberry Pi Node One (Publisher)**

The circuit schematic below shows the connections of the push button to pin GPIO 25.



- **Raspberry Pi Node two (Subscriber)**

The circuit schematic below shows the connections of the LED to pin GPIO 17.



Software Code

- **Publisher Python code**

```
import paho.mqtt.client as mqtt
import time
import RPi.GPIO as GPIO
LED=25
mqttBroker = "192.168.1.10"   # broker Ip
client = mqtt.Client("publisherclient")
client.username_pw_set("client1", "123456789") # username and password
client.connect(mqttBroker)
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED,GPIO.IN,pull_up_down=GPIO.PUD_DOWN))
while True:
   ledvalue=GPIO.input(LED)
   if(ledvalue==1):
      client.publish("topic/led",1)
      print("Just published " + str(1))
      time.sleep(2)
   else:
       client.publish("topic/led",0)
       print("Just published " + str(0))
       time.sleep(2)
```

- **Subscriber Python code**

```
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
import time
LEDin=17
def on_message(client, userdata, message):
   if (str(message.payload.decode("utf-8"))=='1'):
         GPIO.output(LEDin,GPIO.HIGH)
         print("Received message: ", str(message.payload.decode("utf-8")))
         time.sleep(2)
   else:
         GPIO.output(LEDin,GPIO.LOW)
```

```
        print("Received message: ", str(message.payload.decode("utf-8")))
        time.sleep(2)
mqttBroker = "192.168.1.10"
client = mqtt.Client("subscriberclient")
client.username_pw_set("client1", "123456789")
client.connect(mqttBroker)
client.subscribe("topic/led")
GPIO.setmode(GPIO.BCM)
GPIO.setup(LEDin,GPIO.OUT)
client.on_message = on_message
client.loop_forever()
```

Testcase Scenario

1. Connect the Raspberry Nodes, as shown in above figures.
2. Upload the software code to the Raspberry Nodes using **Thonny, Python IDE**.
3. Open the MQTT Mosquitto broker on windows, as shown in part 1.
4. Start publisher-code, as shown in part 4 (software code). Publish discrete values to “topic/led” using push button to control LED. (Raspberry Pi node one acts as publisher)
5. Start subscriber-code, as shown in part 4 (software code) to receive published discrete value on “topic/led”. (Raspberry Pi node two acts as subscriber)