Lab session week 8: IoT Programming with MQTT Protocol

Aim

The aim of this tutorial is for students to be able to confidently use MQTT protocol and understand the Publisher-Subscriber architecture.

Important Information

- Please use your own laptop.
- If your laptop does not have a USB type A socket, you will need to bring an adaptor.
- Be gentle with the hardware.
- You can work with a physical Raspberry Pi board or a virtual desktop to complete this task.
- Do not connect your laptop or a physical Raspberry Pi via Ethernet to the Swinburne network.
- If you are using a physical Raspberry Pi, you will need to find a way to connect it to your laptop without using the Swinburne network.

Background

There are several libraries in Arduino to use MQTT protocol. However, those libraries require a shield on the Arduino board to have network connectivity (WiFi or Ethernet cable). Given that we do not have such shield, we will use the same approach that we have been using in previous labs. Thus, we will connect the Arduino board via serial communication to the Raspberry Pi edge server, and we will have another Raspberry Pi working as a cloud server (physical or virtual). Figure 1 represents the intended network diagram of this tutorial. You are free to explore MQTT Arduino libraries if you have an Arduino shield that provides network connectivity to your device.

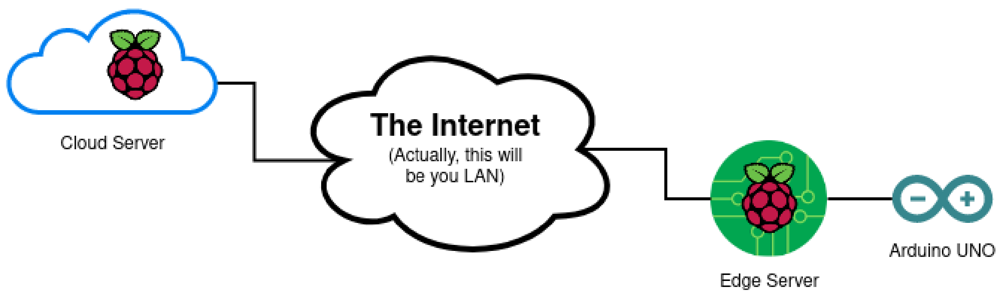


Figure 1: Network Diagram of this Tutorial

Task 1: Getting ready the virtual machines

The easiest way to setup the cloud server is by cloning the virtual machine that we have been using as edge server. If you have been using a physical Raspberry Pi, you can have virtual machine following the steps of Tutorial week 3 and 4. Alternatively, you can use a second physical Raspberry Pi.

Full clone the existing virtual machine by following the steps below:

1. Right click on the existing virtual machine
2. Clone…
3. Change the name to something like: Raspbian Cloud
4. Next
5. Clone type: Full clone
6. Clone

Once it is cloned, we need to unable serial port communication for the Raspbian Cloud server:

1. Right click on Raspbian Cloud
2. Settings…
3. Serial Ports
4. UNCHECK Enable Serial Port
5. OK

Make sure the virtual machine that we have been using in previous labs is identified with "Edge" on its name, so you don't mess up later.

Now you can start both virtual machines and check the IP address of the machines (see Figure 2). In my case:

- Edge IP address: `192.168.1.181`
- Cloud IP address: `192.168.1.196`

Task 2: MQTT Installation

Cloud server

We will install the MQTT broker and client in the cloud server with the following command:

```
sudo apt-get install mosquitto mosquitto-clients
```

We also will need to start the broker (MQTT server):

```
sudo systemctl start mosquitto.service
```

We need to enable the broker (MQTT server), so it starts always after booting up the server:

```
sudo systemctl enable mosquitto.service
```

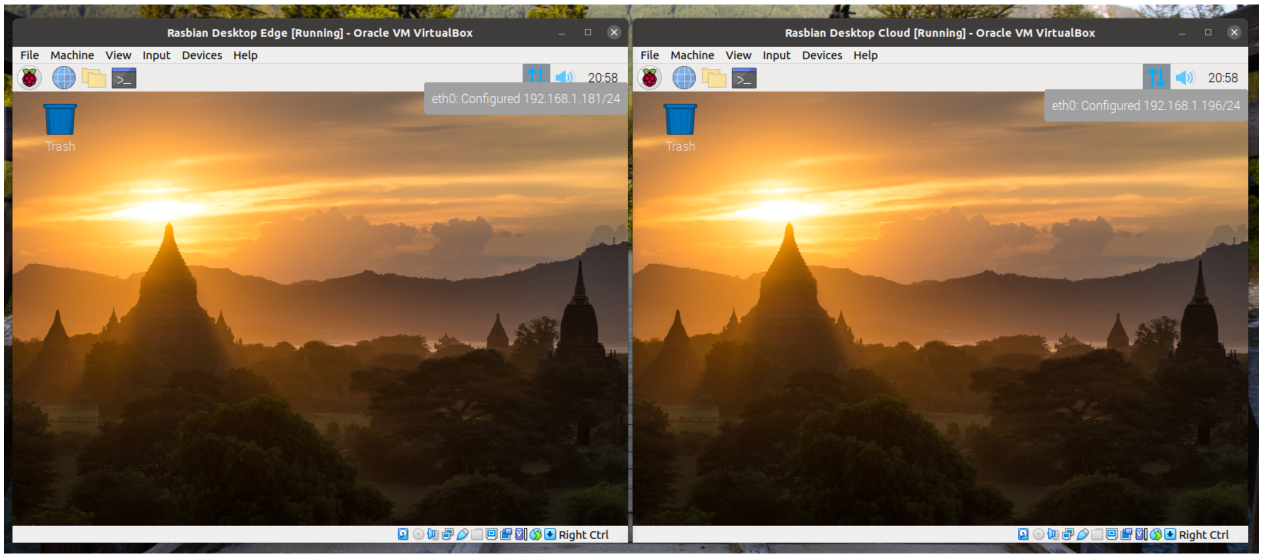

Rasbian Desktop Edge [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
20:58
eth0: Configured 192.168.1.181/24
Trash
Right Ctrl
Rasbian Desktop Cloud [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
20:58
eth0: Configured 192.168.1.196/24
Trash
Right Ctrl

Figure 2: Virtual machines setup

Edge server

We will only install the client on the edge server:

```
sudo apt-get install mosquitto-clients
```

Task 3: MQTT Basic Testing

The following commands provide more information on how to use the publisher and subscriber via command line.

```
mosquitto_pub --help
mosquitto_sub --help
```

The cloud server is subscribed to the topics published by the edge server in the following example. However, the edge server can also be subscribed to the topics published by the cloud server as shown in Figure 3.

Cloud server

In a terminal execute the following command to subscribe to the topic `topic/example`:

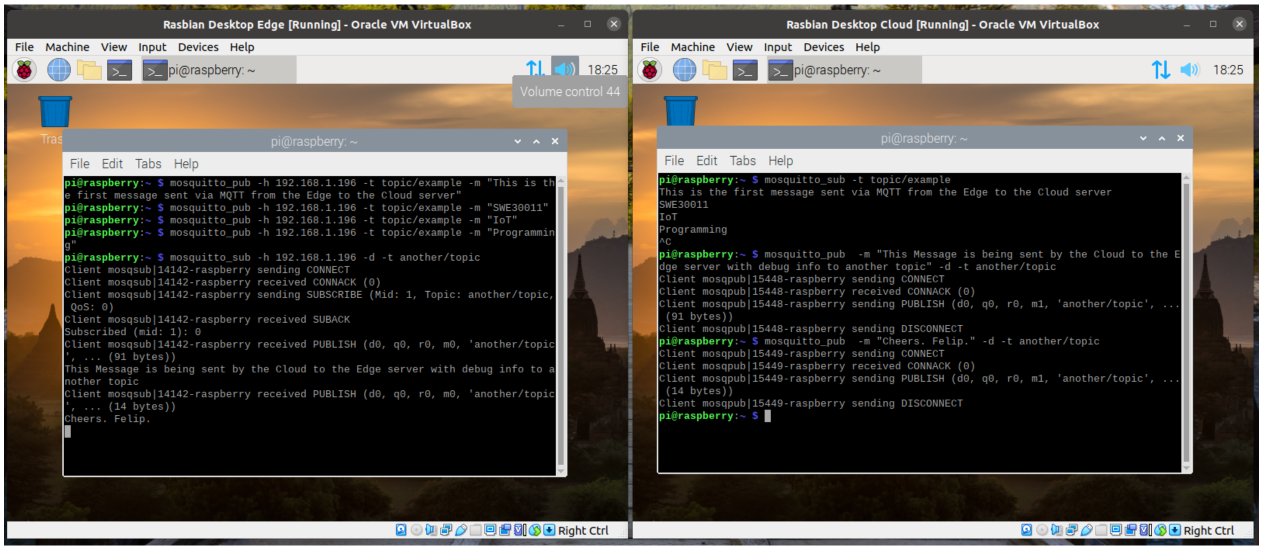
```
mosquitto_sub -t topic/example
```

Edge server

To publish messages use the following command. In this case we need to indicate the IP address of the broker with the `-h` parameter, and the message with the `-m` parameter.

```
mosquitto_pub -h BROKER_IP_ADDRESS -t NAME/TOPIC -m "MESSAGE"
mosquitto_pub -h 192.168.1.196 -t topic/example -m "This is the first message sent via MQTT
                                                   from the Edge to the Cloud server"
```

Feel free to explore the different options when using MQTT. Some examples are shown in Figure 3.


Rasbian Desktop Edge [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
pi@raspberry: ~
18:25
Volume control 44
pi@raspberry: ~
File Edit Tabs Help
Right Ctrl
Rasbian Desktop Cloud [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
pi@raspberry: ~
18:25
pi@raspberry: ~
File Edit Tabs Help
Right Ctrl

```
pi@raspberry:~ $ mosquitto_pub -h 192.168.1.196 -t topic/example -m "This is th
e first message sent via MQTT from the Edge to the Cloud server"
pi@raspberry:~ $ mosquitto_pub -h 192.168.1.196 -t topic/example -m "SWE30011"
pi@raspberry:~ $ mosquitto_pub -h 192.168.1.196 -t topic/example -m "IoT"
pi@raspberry:~ $ mosquitto_pub -h 192.168.1.196 -t topic/example -m "Programmin
g"
pi@raspberry:~ $ mosquitto_sub -h 192.168.1.196 -d -t another/topic
Client mosqsub|14142-raspberry sending CONNECT
Client mosqsub|14142-raspberry received CONNACK (0)
Client mosqsub|14142-raspberry sending SUBSCRIBE (Mid: 1, Topic: another/topic,
 QoS: 0)
Client mosqsub|14142-raspberry received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|14142-raspberry received PUBLISH (d0, q0, r0, m0, 'another/topic
', ... (91 bytes))
This Message is being sent by the Cloud to the Edge server with debug info to a
nother topic
Client mosqsub|14142-raspberry received PUBLISH (d0, q0, r0, m0, 'another/topic
', ... (14 bytes))
Cheers. Felip.
```

```
pi@raspberry:~ $ mosquitto_sub -t topic/example
This is the first message sent via MQTT from the Edge to the Cloud server
SWE30011
IoT
Programming
^C
pi@raspberry:~ $ mosquitto_pub  -m "This Message is being sent by the Cloud to the E
dge server with debug info to another topic" -d -t another/topic
Client mosqpub|15448-raspberry sending CONNECT
Client mosqpub|15448-raspberry received CONNACK (0)
Client mosqpub|15448-raspberry sending PUBLISH (d0, q0, r0, m1, 'another/topic', ...
 (91 bytes))
Client mosqpub|15448-raspberry sending DISCONNECT
pi@raspberry:~ $ mosquitto_pub  -m "Cheers. Felip." -d -t another/topic
Client mosqpub|15449-raspberry sending CONNECT
Client mosqpub|15449-raspberry received CONNACK (0)
Client mosqpub|15449-raspberry sending PUBLISH (d0, q0, r0, m1, 'another/topic', ...
 (14 bytes))
Client mosqpub|15449-raspberry sending DISCONNECT
pi@raspberry:~ $
```

Figure 3: MQTT exchange of messages using command line

Task 4: Programming MQTT with paho

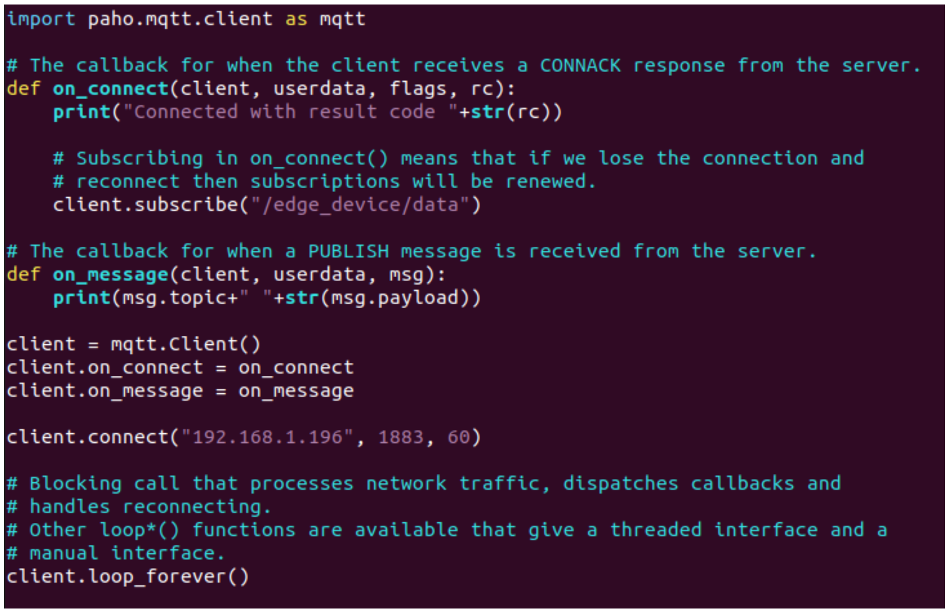
MQTT command line is great for testing purposes; however, it is not very suitable for programming. For instance, if we want to send a message to the Cloud server via MQTT every time our Arduino board sends data via serial communication, we can use an MQTT client library such as `paho-mqtt`.

We will use Eclipse Paho MQTT Python client library. To install it, we will execute the following:

```
sudo apt-get install python-paho-mqtt
```

Subscriber

Figure 4 shows a very simple example that subscribes to the topic `/edge_device/data` of the broker (IP address `192.168.1.196`) and prints out the resulting messages.

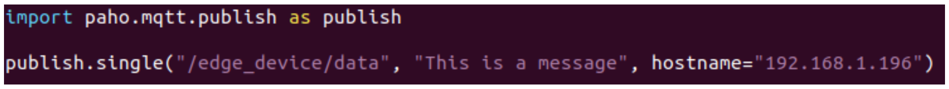


```
import paho.mqtt.client as mqtt

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("/edge_device/data")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

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

client.connect("192.168.1.196", 1883, 60)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```

Figure 4: Basic example of a Subscriber

Publisher

Figure 5 shows a very simple example that publishes a message to the topic `/topic/example` of the broker (IP address `192.168.1.196`)



```
import paho.mqtt.publish as publish

publish.single("/edge_device/data", "This is a message", hostname="192.168.1.196")
```

Figure 5: Basic example of a Publisher

Task 5: Sending Arduino data to the Cloud server via MQTT

1. Program the Arduino board to read data (e.g., read the potentiometer) and send it periodically via serial communication to the edge server.
2. Program the edge server to read the data received via serial communication, and publish it to a topic (e.g., /edge_sensor/data).
3. Subscribe the cloud server to the topic, and store all the data being received in a database on the cloud server.
4. Add the timestamp in the MQTT message and store it in the database.

Resources

Arduino

- Arduino Language Reference: https://www.arduino.cc/reference/en/
- Arduino Foundations: https://www.arduino.cc/en/Tutorial/Foundations
- Arduino Built-In Examples: https://www.arduino.cc/en/Tutorial/BuiltInExamples

Raspberry Pi

- https://www.raspberrypi.org/

GNU/Linux

- Linux Journey is a site dedicated to making learning Linux fun and easy. https://linuxjourney.com/
- Introduction to Linux: A Hands on Guide. https://tldp.org/LDP/intro-linux/intro-linux.pdf
- Introduction to Linux (LFS101), the Linux Foundation training course: https://training.linuxfoundation.org/training/introduction-to-linux/

MQTT

- MQTT: The Standard for IoT Messaging. https://mqtt.org/
- Eclipse Mosquitto: An open source MQTT broker. https://www.mosquitto.org/
- Eclipse Paho MQTT Python client library. https://pypi.org/project/paho-mqtt/

Python

- Python for Beginners (Programmers). https://www.python.org/about/gettingstarted/