SOFE 4610: Design and Analysis of IoT

Lab #4: Installing and Testing Mosquitto MQTT on the RPi

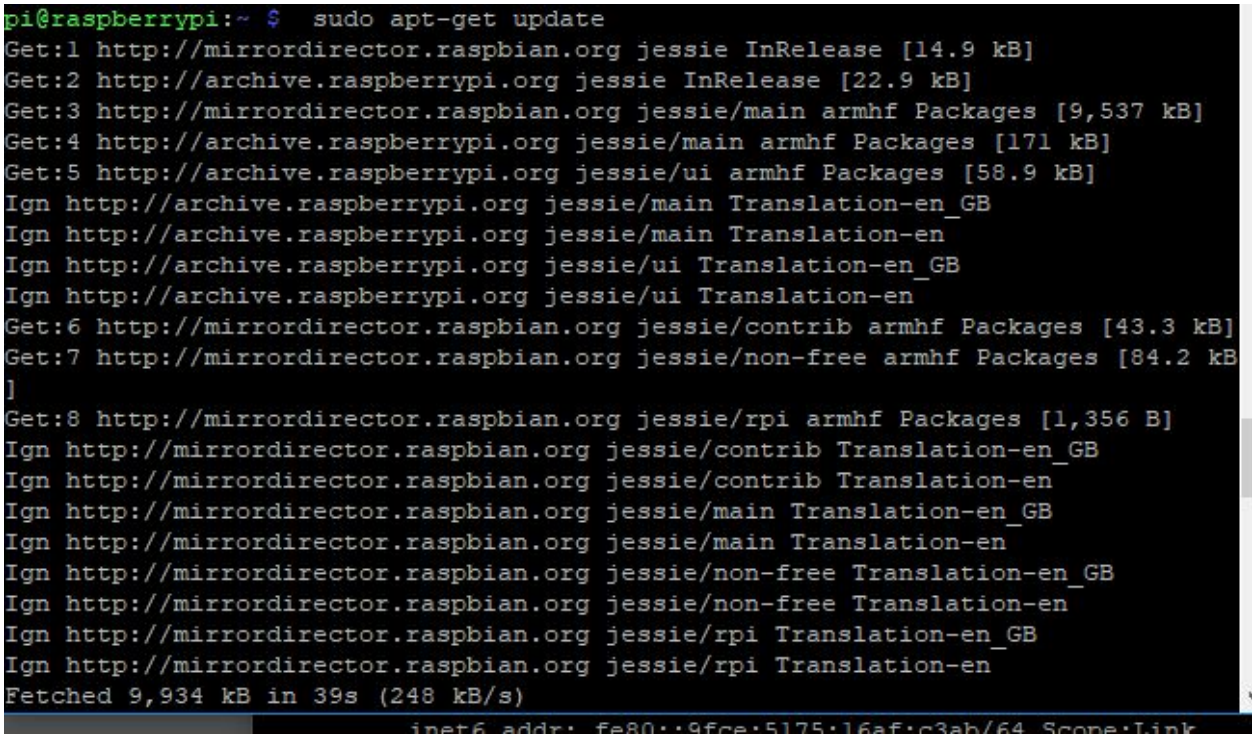
Devante Wilson and Youssef Osman
100554361 and 100715637

Introduction

The proceedings of this lab report detail the installation of MQTT on Raspberry Pi. Additionally, our group will test Mosquitto by sending test data between two clients. Furthermore, we will use Python code to test and send messages using MQTT.

Lab Tasks

Setup

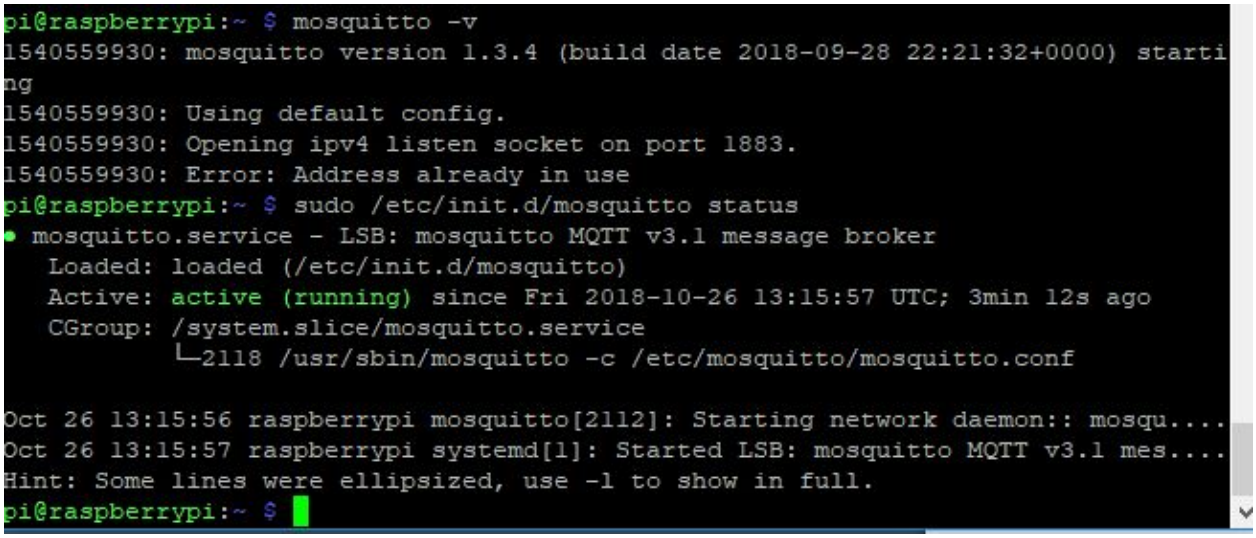


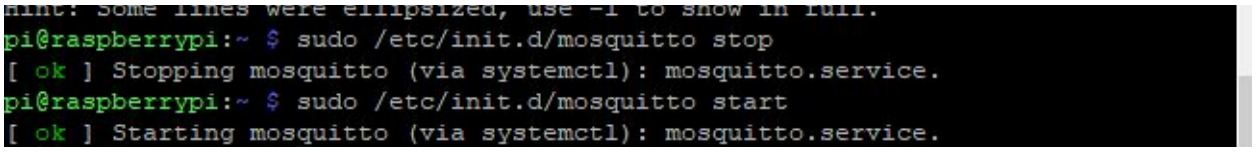
We installed the three parts of Mosquitto:

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

To start, stop, and get status of the Mosquitto server:

```
mosquitto -v      #[should be >= version 1.4.14]
sudo /etc/init.d/mosquitto status
sudo /etc/init.d/mosquitto stop
sudo /etc/init.d/mosquitto start
```





Next, we ran the following command:

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

```
pi@raspberrypi:~ $ sudo pip install paho-mqtt
Downloading/unpacking paho-mqtt
  Downloading paho-mqtt-1.4.0.tar.gz (88kB): 88kB downloaded
  Running setup.py (path:/tmp/pip-build-vNaEys/paho-mqtt/setup.py) egg_info for
package paho-mqtt

Installing collected packages: paho-mqtt
  Running setup.py install for paho-mqtt

Successfully installed paho-mqtt
Cleaning up...
pi@raspberrypi:~ $
```

To test the Mosquitto server, the following was typed into a window terminal:

```
mosquitto_sub -d -t hello/world
```

```
pi@raspberrypi:~ $ mosquitto_sub -d -t hello/world
Client mosqsub/2318-raspberryp sending CONNECT
Client mosqsub/2318-raspberryp received CONNACK
Client mosqsub/2318-raspberryp sending SUBSCRIBE (Mid: 1, Topic: hello/world, Qo
S: 0)
Client mosqsub/2318-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/2318-raspberryp received PUBLISH (d0, q0, r0, m0, 'hello/world',
... (29 bytes))
Hello from Terminal window 2!
```

In window 2, the following command was used:

```
mosquitto_pub -d -t hello/world -m "Hello from Terminal window 2!"
```

```
pi@raspberrypi:~ $ mosquitto_pub -d -t hello/world -m "Hello from Terminal windo
w 2!"
Client mosqpub/2319-raspberryp sending CONNECT
Client mosqpub/2319-raspberryp received CONNACK
Client mosqpub/2319-raspberryp sending PUBLISH (d0, q0, r0, m1, 'hello/world', .
.. (29 bytes))
Client mosqpub/2319-raspberryp sending DISCONNECT
pi@raspberrypi:~ $
```

Lab Task #1

To send messages between the raspberry Pi, we ensured a common IP address between the Ubuntu virtual machine and the Windows OS. 3

```
youssef@youssef-virtual-machine:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:b6:eb:e8
          inet addr:192.168.19.128  Bcast:192.168.19.255  Mask:255.255.255.0
          inet6 addr: fe80::3e90:60cc:b5e7:e20/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6027 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1923 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8094287 (8.0 MB)  TX bytes:132356 (132.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:216 errors:0 dropped:0 overruns:0 frame:0
          TX packets:216 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16645 (16.6 KB)  TX bytes:16645 (16.6 KB)
```

```
IPv4 address:                    10.150.3.134/18
```

We then used the PC as the subscriber and the Raspberry Pi as both the broker and publisher. To accomplish this, *Mosquitto* and *Mosquitto-clients* were installed in Ubuntu using the commands:

sudo apt-get install mosquitto mosquitto-clients
mosquitto_sub -d -t /test -h 192.168.0.21

```
youssef@youssef-virtual-machine:~$ mosquitto_sub -d -t /test -h 192.168.137.2
Client mosqsub/3203-youssef-vi sending CONNECT
Client mosqsub/3203-youssef-vi received CONNACK
Client mosqsub/3203-youssef-vi sending SUBSCRIBE (Mid: 1, Topic: /test, QoS: 0)
Client mosqsub/3203-youssef-vi received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/3203-youssef-vi sending PINGREQ
Client mosqsub/3203-youssef-vi received PINGRESP
Client mosqsub/3203-youssef-vi sending PINGREQ
Client mosqsub/3203-youssef-vi received PINGRESP
Client mosqsub/3203-youssef-vi received PUBLISH (d0, q0, r0, m0, '/test', ... (2
2 bytes))
This is a test message
```

In the raspberry Pi, the following command was used to send the test message.

```
mosquitto_pub -d -t /test -m “This is a test message”
```

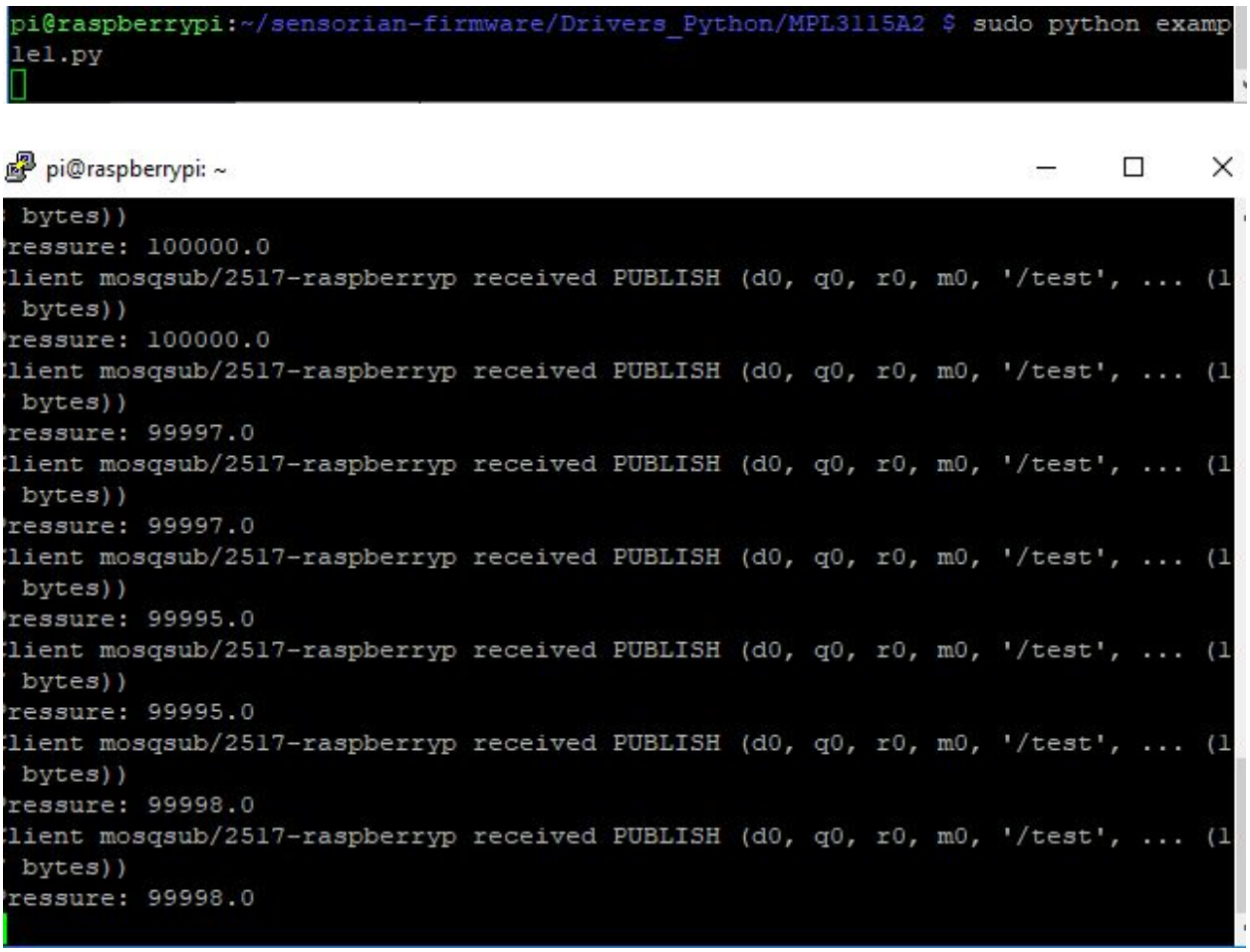
```
pi@raspberrypi:~ $ mosquitto_pub -d -t /test -m "This is a test message"
'Client mosqpub/2367-raspberryp sending CONNECT
Client mosqpub/2367-raspberryp received CONNACK
Client mosqpub/2367-raspberryp sending PUBLISH (d0, q0, r0, m1, '/test', ... (22
 bytes))
Client mosqpub/2367-raspberryp sending DISCONNECT
pi@raspberrypi:~ $ '
```

Lab Task #2

For this task, we needed to write some python code to connect to the Sensorian and read sensors data. The data from Sensorian can be published through MQTT from the Raspberry Pi to another client.

Below is the procedure which reads from the Sensorian being published to the */test* topic:

```
#!/usr/bin/python
import time
import os, sys
import MPL3115A2 as altibar
from ctypes import *
import paho.mqtt.publish as publish

sensor = CDLL("./libMPL.so")                          #load t$
sensor.I2C_Initialize(altibar.MPL3115A2_ADDRESS)      #initialize I2C and BCM$


def main(argv):

    AltiBar = altibar.MPL3115A2()
    AltiBar.ActiveMode()                              #puts sensor in active mode
    AltiBar.BarometerMode()                           #puts sensor in active mode
    topic = "/test"
    while True:
        press = AltiBar.ReadBarometricPressure()
        send = "Pressure:  " + str(press)
        publish.single(topic, send, hostname="192.168.0.19") #this is the IP address on pi
        time.sleep(1)

main(sys.argv[1:])
```

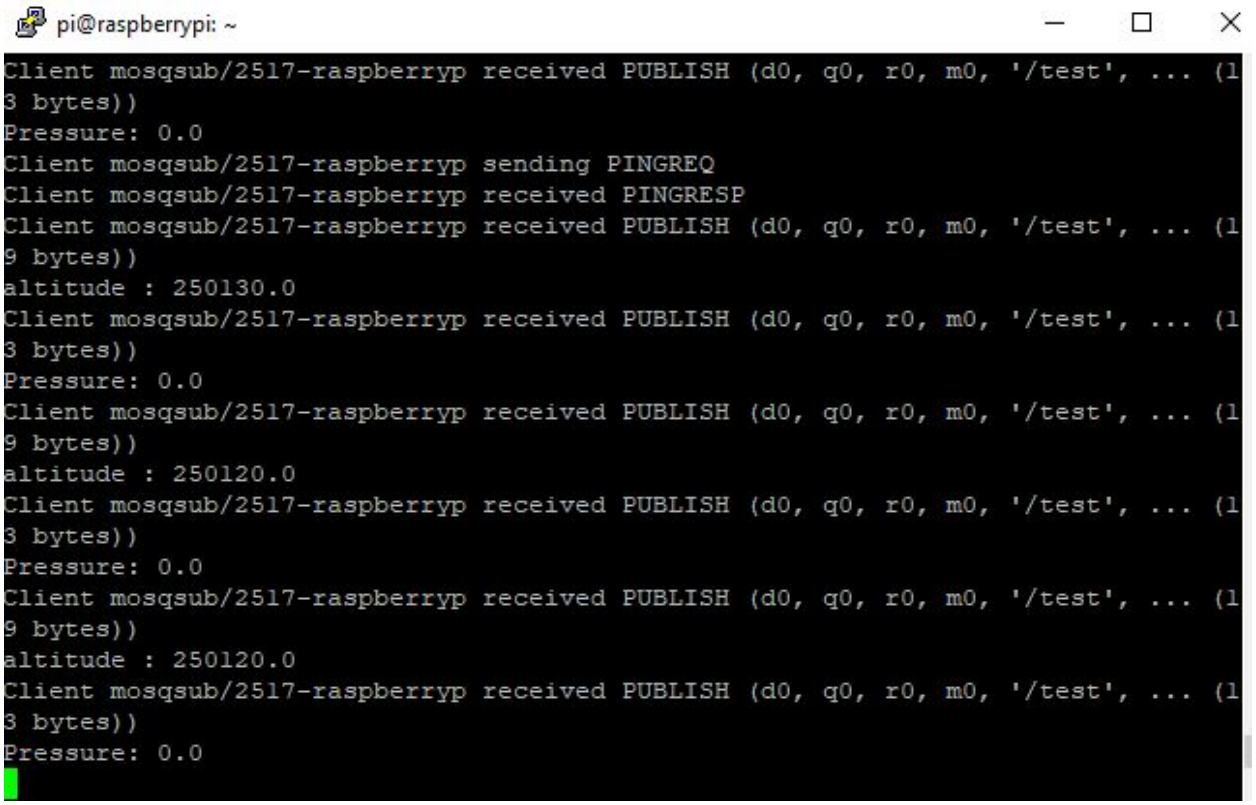


The next step was to modify the code to read values from (temperature, altitude, and pressure) sensors at the same time.

```
#!/usr/bin/python
import time
import os, sys
import MPL3115A2 as altibar
from ctypes import *
import paho.mqtt.publish as publish

sensor = CDLL("./libMPL.so")                          #load t$
sensor.I2C_Initialize(altibar.MPL3115A2_ADDRESS)      #initialize I2C and BCM$


def main(argv):

topic = "/test"
      while True:
         AltiBar.ActiveMode()                         #puts sensor in active mode
         AltiBar.BarometerMode()                       #puts sensor in active mode
         pressure= AltiBar.ReadBarometricPressure()
         send = "Pressure:  " + str(pressure)
         publish.single(topic, send, hostname="192.168.0.19") #this is the IP address on pi
         time.sleep(0.5)

         AltiBar.ActiveMode()                         #puts sensor in active mode
         time.sleep(0.5)
         AltiBar.TemperatureMode()
         temperature = AltiBar.ReadTemperature()
         time.sleep(0.5)
         send = "temperature :  " + str(temperature)
         publish.single(topic, send, hostname="192.168.0.19") #this is the IP address on pi
         time.sleep(1)

         AltiBar.ActiveMode()                         #puts sensor in active mode
         time.sleep(0.5)
         AltiBar.AltimeterMode()
         time.sleep(0.5)
         altitude =  AltiBar.ReadAltitude()
         time.sleep(0.5)
         send = "altitude  :  " + str(altitude)
         publish.single(topic, send, hostname="192.168.0.19") #this is the IP address on pi
         time.sleep(1)

main(sys.argv[1:])
```

The data was send through the MQTT publisher.

```
pi@raspberrypi:~/sensorian-firmware/Drivers_Python/MPL3115A2 $ nano example5.py
pi@raspberrypi:~/sensorian-firmware/Drivers_Python/MPL3115A2 $ sudo python examp
le5.py
```



Conclusion

After performing all the tasks during the lab session, our group learned how to go through the installation of MQTT on Raspberry Pi. Additionally, we tested Mosquitto by sending test data between two clients. Furthermore, we gained experience with using Python to modify code to test and send messages using MQTT.