UOIT
CHALLENGE INNOVATE CONNECT

SOFE 4610U:

Internet of Things

Lab #4: Installing and testing Mosquitto MQTT on the Raspberry Pi

Objectives

- Install Mosquitto MQTT on Raspberry Pi
- Test Mosquitto by sending test data between two clients
- Use Python code to test and send messages using MQTT

Important Notes

- Work in groups of **two or three** students
- All reports must be submitted as a PDF on blackboard, if source code is included submit your report as PDF and source files as an archive (e.g. zip, tar.gz)
- Save the submission as <lab_number>_<first student's id> (e.g. lab1_100123456.pdf)
- If you cannot submit the document on blackboard then please contact the TA with your submission: **xxxx@uoit.net**

What is MQTT?

MQTT is a publish-subscribe based “light weight” messaging protocol for use on top of the TCP/IP protocol, such as the Wi-Fi packets that we are using in this lab.
The publish-subscribe messaging pattern requires a message broker. The broker is responsible for distributing messages to interested clients based on the topic of a message.

In this lab, we will use the sensorian, which is connected to Raspberry Pi. The aim of this lab is to install and test MQTT Mosquitto to send and receive data over the network. Additionally, we will write a python code to read data from sensorian kit to send it through MQTT protocol.

Lab Activity

Software Setup

1. aptitude is a text-based interface to the Debian GNU/Linux package system. It allows the user to view the list of packages and to perform package management tasks such as installing, upgrading, and removing packages. It allows to run commands from a visual interface or from the command-line

   - aptitude did not want to work in sensorian: It tried to connect to 192.168.137.3:3128 instead of mirrordirector.raspbian.org.
   - The solution was to remove the proxy that is configured in /etc/apt/apt.conf.
   - The easiest way to do this is to run

   ```
   $sudo mv /etc/apt/apt.conf /etc/apt/apt.conf.old
   ```

2. `$sudo aptitude update  #(Update the system packages)`
3. Unfortunately, the Raspberry Pi normal “apt-get” archives do not contain the latest version of the Mosquitto software. So, the first thing is to open a terminal window to your Raspberry Pi and do the following

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-wheezy.list
cd
sudo apt-get update
```

4. Next, we can install the three parts of Mosquitto proper.

   mosquitto – the MQTT broker (or in other words, a server)
   mosquitto-clients – command line clients, very useful in debugging
   python-mosquitto – the Python language bindings

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

5. Starting/ stop/ get status 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
```

6. This code 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.

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

7. Testing the Mosquitto server

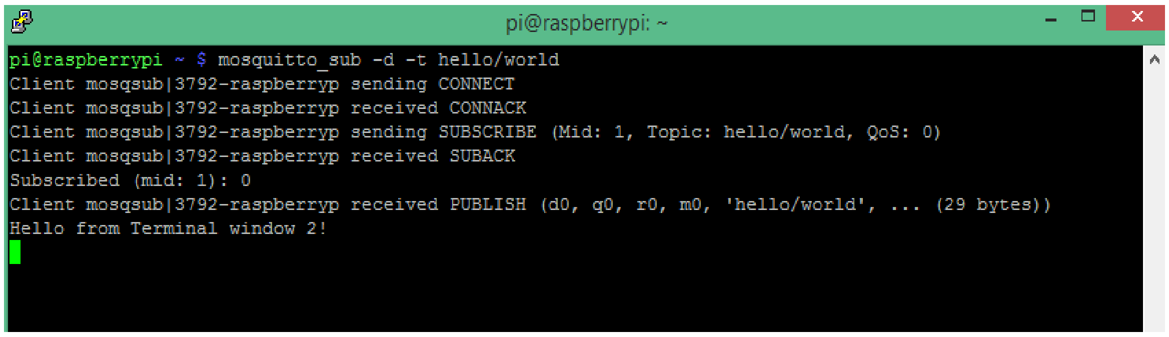
Open up two more terminal windows. In Terminal window 1 type:

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

In Terminal window 2 type:

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

When you have done the second statement, you should see this in the Terminal 1 window as shown below.



```
pi@raspberrypi ~ $ mosquitto_sub -d -t hello/world
Client mosqsub|3792-raspberryp sending CONNECT
Client mosqsub|3792-raspberryp received CONNACK
Client mosqsub|3792-raspberryp sending SUBSCRIBE (Mid: 1, Topic: hello/world, QoS: 0)
Client mosqsub|3792-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|3792-raspberryp received PUBLISH (d0, q0, r0, m0, 'hello/world', ... (29 bytes))
Hello from Terminal window 2!
```

Now you are running the Mosquitto broker successfully.

Lab Tasks

Task 1:

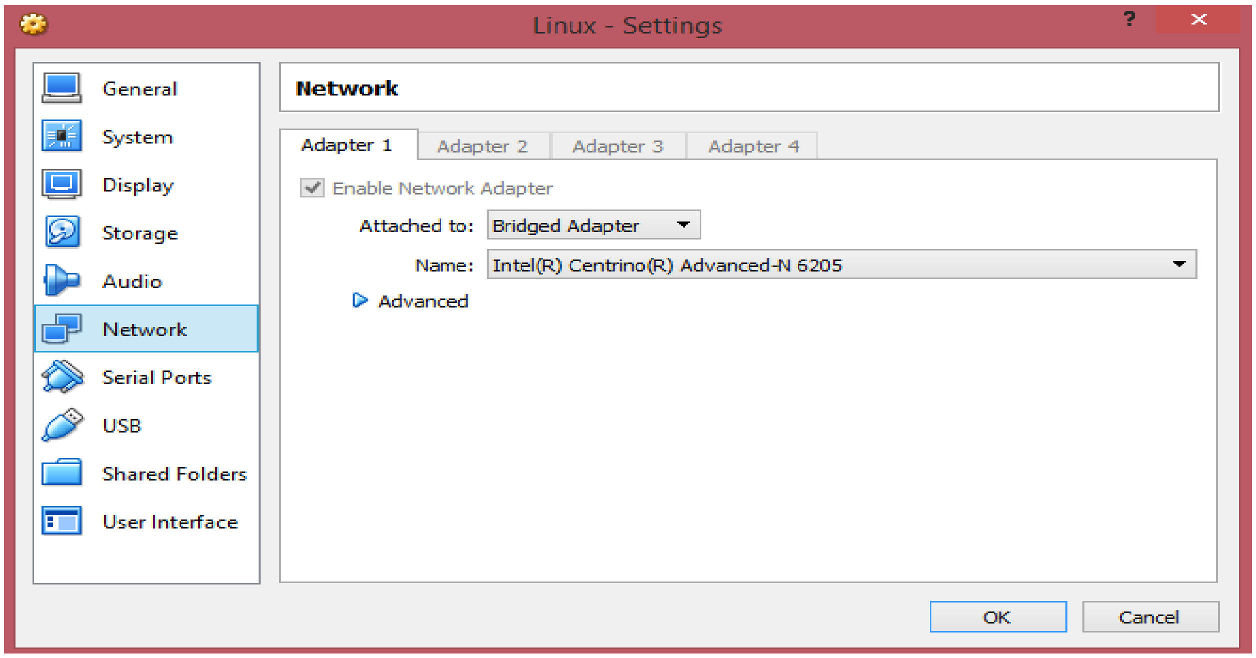
1. Now we want to test and send messages between the Raspberry Pi and a computer using MQTT.
2. To complete this lab, you should install Ubuntu OS on a virtual box.
3. The idea is to send messages between the raspberry Pi and Ubuntu OS.
4. You should be familiar with the IP address of your computer, Ubuntu, and the Pi.
5. The three platform should be in the same IP range/ Domain, to be able to communicate between all of those platforms.
6. To view the IP address of Ubuntu machine, use:

```
$ ifconfig
```

7. To view the IP address of your computer running Windows OS

```
C:\ipconfig
```

8. In case the IP address of Ubuntu machine differs than the windows OS, Use Bridged Mode instead of NAT for the VM in the host Virtual Box GUI



Enable Network Adapter ✓
Attached to: Bridged Adapter
Name: Intel(R) Centrino(R) Advanced-N 6205

9. This assumes that 192.168.0.21 is the ip address of the Raspberry Pi.
10. This example will use the PC as the subscriber and the Raspberry Pi as both the broker and publisher
11. In Ubuntu machine install mosquito and mosquitto-clients

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

mosquitto_sub -d -t /test -h 192.168.0.21
```

12. In the Raspberry Pi, run the following:

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

If everything has been configured correctly, the subscriber should receive the message sent by the publisher

Task 2:

Now, we want to write a python code to connect to the Sensorian and read sensors data. The data getting 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:])
```

1- Create a python file and paste the above code into the file.

```
$ nano  example.py           #(used to create a file called example.py)
```

2- Modify the code to print the reading data from Sensorian on the Respberry Pi terminal.

3- Run the python file
   $ python example.py

4- Copy the screenshots of the output and include them in your report

5- Modify the code to read values from the temperature sensor and send the data if the temperature is over 30 degrees, run the code and check the output

- **Use AltiBar.ReadTemperature() to read values from the temperature sensor**
- **Use altitude = AltiBar.ReadAltitude() to read values from the altitude sensor**

6- Copy the screenshots of the output and include them in your report
7- Modify the code to read values from (temperature, altitude and pressure) sensors at the same time.
8- Send the data through MQTT publisher.
9- Copy the screenshots of the output and include them in your report

Deliverables

Complete all lab tasks, for task 1, write the step by step including screen shots; for task2, submit the complete code and the screen shots for booth the raspberry Pi and Ubuntu terminal.

Please note, all lab report will have title pages, introduction, content of the lab tasks and conclusion.

```
#!/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:])
```