



SOFE 4610U:  
Internet of Things

# Lab #4: Leveraging MQTT communication for IoT applications

## Table of Contents

<b>1. Objectives</b> .....	2
<b>2. Important Notes</b> .....	3
<b>3. Concepts</b> .....	3
3.1 MQTT .....	3
3.2 Mosquitto .....	3
3.3 paho-mqtt .....	3
<b>4. Software Setup</b> .....	3
<b>5. Lab Tasks</b> .....	4
5.1: Utilize bluepy library to communicate .....	5
5.2: Send messages between the raspberry Pi and Ubuntu OS .....	6
5.3: Communicate between Raspberry Pi and BLE HC05 .....	8
<b>6. Deliverables</b> .....	9

## 1. Objectives

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

## 2. Important Notes

- Work in groups of **Five** 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: Leon Wu <leon.wu@ontariotechu.ca>

## 3. Concepts

### 3.1 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.

<https://mqtt.org/>

### 3.2 Mosquitto

Mosquitto is an Open Source MQTT server. A public, hosted test server is also available.

<https://www.mosquitto.org/>

### 3.3 paho-mqtt

The paho-mqtt is a Python client library, which implements versions 5.0, 3.1.1, and 3.1 of the MQTT protocol. 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.

<https://pypi.org/project/paho-mqtt/>

In this lab, we will use the CC2665 SensorTag 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 the SensorTag kit to send it through MQTT protocol.

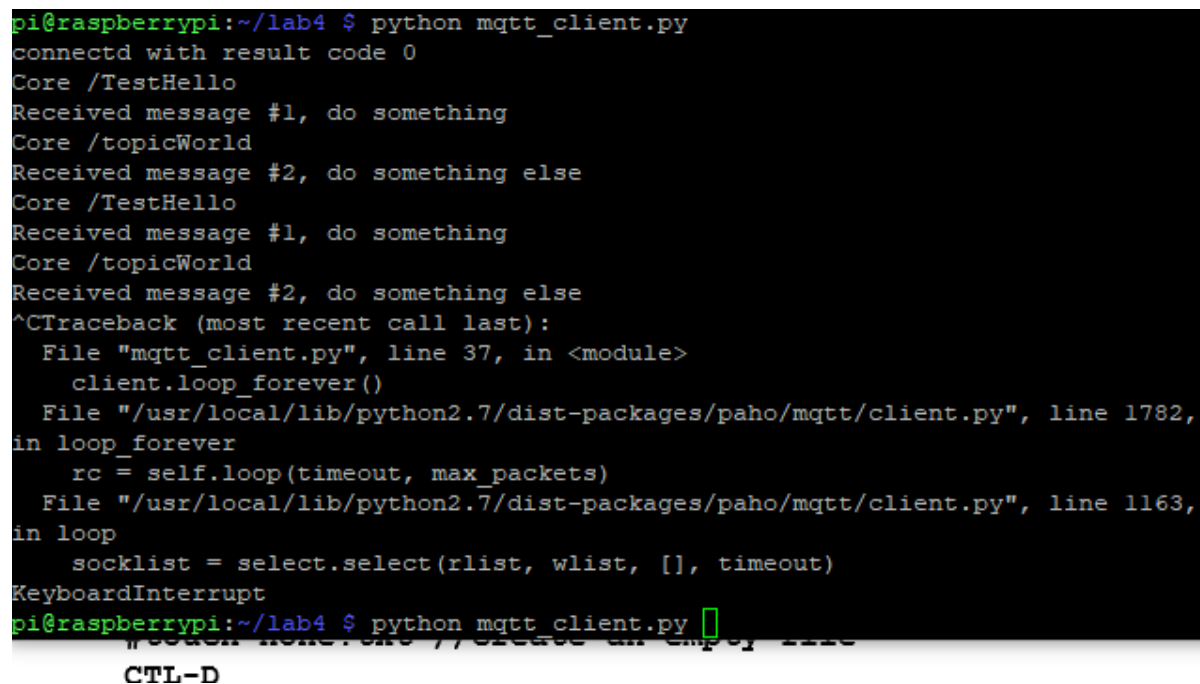
## 4. Software Setup

MQTT is a lightweight messaging protocol designed for low-cost and low-power embedded systems, such as Raspberry Pi platform we are using for this lab.

In this experiment, we will use Python language to program. While other languages, such as C language, provides speed, Python is arguably the ultimate proto-typing language for its easy-to-use libraries with PIP.

It could be challenging to get an MQTT client to work in C++ because some libraries only work on Windows systems, and other libraries have complex installation and linking options.

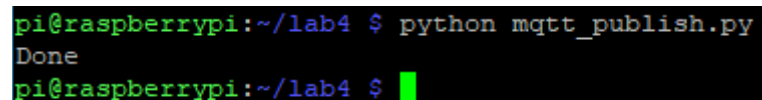
1. In this experiment, we use PIP command to install the library on Raspberry pi.
  - 1.1. `$sudo pip3 install paho-mqtt` // (for python3) on "buster" pi – invoke python3
  - 1.2. `$sudo pip install paho-mqtt` (for python2.7)
2. Introduce how paho-mqtt library for the client and server python code
  - 2.1. `$python3 mqtt_client.py`



```
pi@raspberrypi:~/lab4 $ python mqtt_client.py
connectd with result code 0
Core /TestHello
Received message #1, do something
Core /topicWorld
Received message #2, do something else
Core /TestHello
Received message #1, do something
Core /topicWorld
Received message #2, do something else
^C
Traceback (most recent call last):
  File "mqtt_client.py", line 37, in <module>
    client.loop_forever()
  File "/usr/local/lib/python2.7/dist-packages/paho/mqtt/client.py", line 1782,
in loop_forever
    rc = self.loop(timeout, max_packets)
  File "/usr/local/lib/python2.7/dist-packages/paho/mqtt/client.py", line 1163,
in loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt
pi@raspberrypi:~/lab4 $ python mqtt_client.py
```

CTL-D

- 2.2. `$python3 mqtt_publish.py`



```
pi@raspberrypi:~/lab4 $ python mqtt_publish.py
Done
pi@raspberrypi:~/lab4 $
```

\* <https://learn.adafruit.com/diy-esp8266-home-security-with-lua-and-mqtt/configuring-mqtt-on-the-raspberry-pi>

## 5. Lab Tasks

### 5.1: Utilize bluepy library to communicate

```
$ sudo apt install -y python3 python3-pip python3-dev
```

```
$ pip3 install bluepy
```

```
pi@raspberrypi:~ $ pip3 install bluepy
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting bluepy
  Downloading https://www.piwheels.org/simple/bluepy/bluepy-1.3.0-cp37-cp37m-linux_armv7l.whl (560kB)
    100% |#####| 563kB 658kB/s
Installing collected packages: bluepy
  The scripts blescan, sensortag and thingy52 are installed in '/home/pi/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed bluepy-1.3.0
```

Then, go to directory `/home/pi/.local/bin`, study `btle.py` provided by the bluepy package for communicating with Sensortag using Python language

```
$ cd ~/.local/lib/python3.7/site-packages/bluepy
```

```
$ python3 btle.py xx:xx:xx:xx:xx
```

\* where xx are the mac address you obtained from previous lab

```

pi@raspberrypi:~/local/lib/python3.7/site-packages/bluepy $ python3 btle.py 00:
0C:BF:02:7A:3D
Connecting to: 00:0C:BF:02:7A:3D, address type: public
Service <uuid=Generic Access handleStart=1 handleEnd=5> :
  Characteristic <Device Name>, hnd=0x2, supports READ
  -> b'HC-02'
  Characteristic <Appearance>, hnd=0x4, supports READ
  -> b'\x00\x00'
Service <uuid=Generic Attribute handleStart=6 handleEnd=6> :
Service <uuid=Device Information handleStart=16 handleEnd=32> :
  Characteristic <Manufacturer Name String>, hnd=0x11, supports READ
  -> b'Realtek'
  Characteristic <Model Number String>, hnd=0x13, supports READ
  -> b'8761ATV'
  Characteristic <Serial Number String>, hnd=0x15, supports READ
  -> b'00-01'
  Characteristic <Hardware Revision String>, hnd=0x17, supports READ
  -> b'00-01'
  Characteristic <Firmware Revision String>, hnd=0x19, supports READ
  -> b'Nov 30 2017 10:35:37'
  Characteristic <Software Revision String>, hnd=0x1b, supports READ
  -> b'6027'
  Characteristic <System ID>, hnd=0x1d, supports READ
  -> b'00000000-00000000'
  Characteristic <IEEE 11073-20601 Regulatory Certification Data List>, hnd=0x
1f, supports READ
  -> b'IEEE-11073-20601'
Service <uuid=49535343-fe7d-4ae5-8fa9-9fafd205e455 handleStart=33 handleEnd=45>
:
  Characteristic <49535343-6daa-4d02-abf6-19569aca69fe>, hnd=0x22, supports RE
AD WRITE
  -> b'\x00\x0f\x00\x0f\x00\x00\x00\x00\x02'
  Characteristic <49535343-aca3-481c-91ec-d85e28a60318>, hnd=0x24, supports RE
AD WRITE
  -> b''
  Characteristic <49535343-1e4d-4bd9-ba61-23c647249616>, hnd=0x26, supports NO
TIFY
  Characteristic <49535343-8841-43f4-a8d4-ecbe34729bb3>, hnd=0x29, supports WR
ITE NO RESPONSE WRITE
  Characteristic <49535343-aca3-481c-91ec-d85e28a60318>, hnd=0x2b, supports RE
AD WRITE NOTIFY
  -> b''
pi@raspberrypi:~/local/lib/python3.7/site-packages/bluepy $

```

## 5.2: Send messages between the raspberry Pi and Ubuntu OS

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 machine.
3. The purpose 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
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. **[Broker]** Install mosquitto on the Raspberry pi

- a. Get the repo key

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
```

```
sudo apt-key add mosquitto-repo.gpg.key
```

- b. Make the repository available

```
cd /etc/apt/sources.list.d/
```

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list  
sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list  
sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list
```

- c. Update apt and install mosquito

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

- d. Set configuration

```
cd /etc/mosquito  
nano mosquito.conf
```

add 2 lines as following to allow remote connection

```
listener 1883  
allow_anonymous true
```

e. Running mosquito

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

12. [Subscriber] 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
```

13. [Publisher] 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**

### *5.3: Communicate between Raspberry Pi and BLE HC05*

Write the Python code subscriber.py and publisher.py to use mqtt to communicate between Raspberry Pi and HC05, and obtain the sensor data.

1. [Broker] Raspberry Pi, start mosquito, see Task 5.2 step 11.
2. [Publisher] Raspberry Pi, write a publisher.py, which read data from sensor, and publish the data as message content to the Broker with a custom topic, such as 'task3\_topic\_group1'.



3. [Subscriber] Raspberry Pi, write a subscriber.py, which connect to the broker, subscribe the topic, and print the message.

Publisher and subscriber are clients, which dependent on paho-mqtt library.

Hint: import paho.mqtt.client to publisher.py and subscriber.py, create an Client instance, connect to broker. Some examples for beginners:

<https://mntolia.com/mqtt-python-with-paho-mqtt-client/>

## 6. Deliverables

Complete all lab tasks 5.1 & 5.2, write the step by step including screen shots.

For task 5.3, submit the complete code and the screen shots for both the raspberry Pi and Ubuntu terminal.

For task 5.4, please show screen shots from client and server sides and wireshark captures.

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