

Course: EGDF20 Diploma in Electronic and Computer Engineering

Module: EGE356 IoT System Architecture & Technology

Lab 6: Data Exchange via Edge Gateway using MQTT

Objectives:

1. Understanding MQTT Subscribe/Publish
2. Endpoint to Gateway Data Transfer with MQTT
3. Endpoint to Gateway Data Transfer to Cloud (Assignment 2 – Part 1)

Part 1: Understanding MQTT Subscribe/Publish

1. Download and install the MQTT software – mosquitto.

The software may be downloaded from BrightSpace or <https://mosquitto.org/download/>

Note that installation guide in this lab is only provided for Windows.

Windows

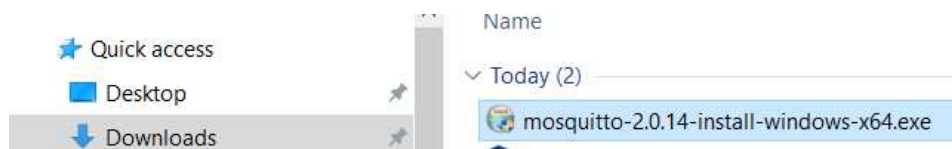
- mosquitto-2.0.14-install-windows-x64.exe (64-bit build, Windows Vista and up, built with Visual Studio Community 2019)
- mosquitto-2.0.14-install-windows-x32.exe (32-bit build, Windows Vista and up, built with Visual Studio Community 2019)

Older installers can be found at <https://mosquitto.org/files/binary/>.

See also README-windows.md after installing.

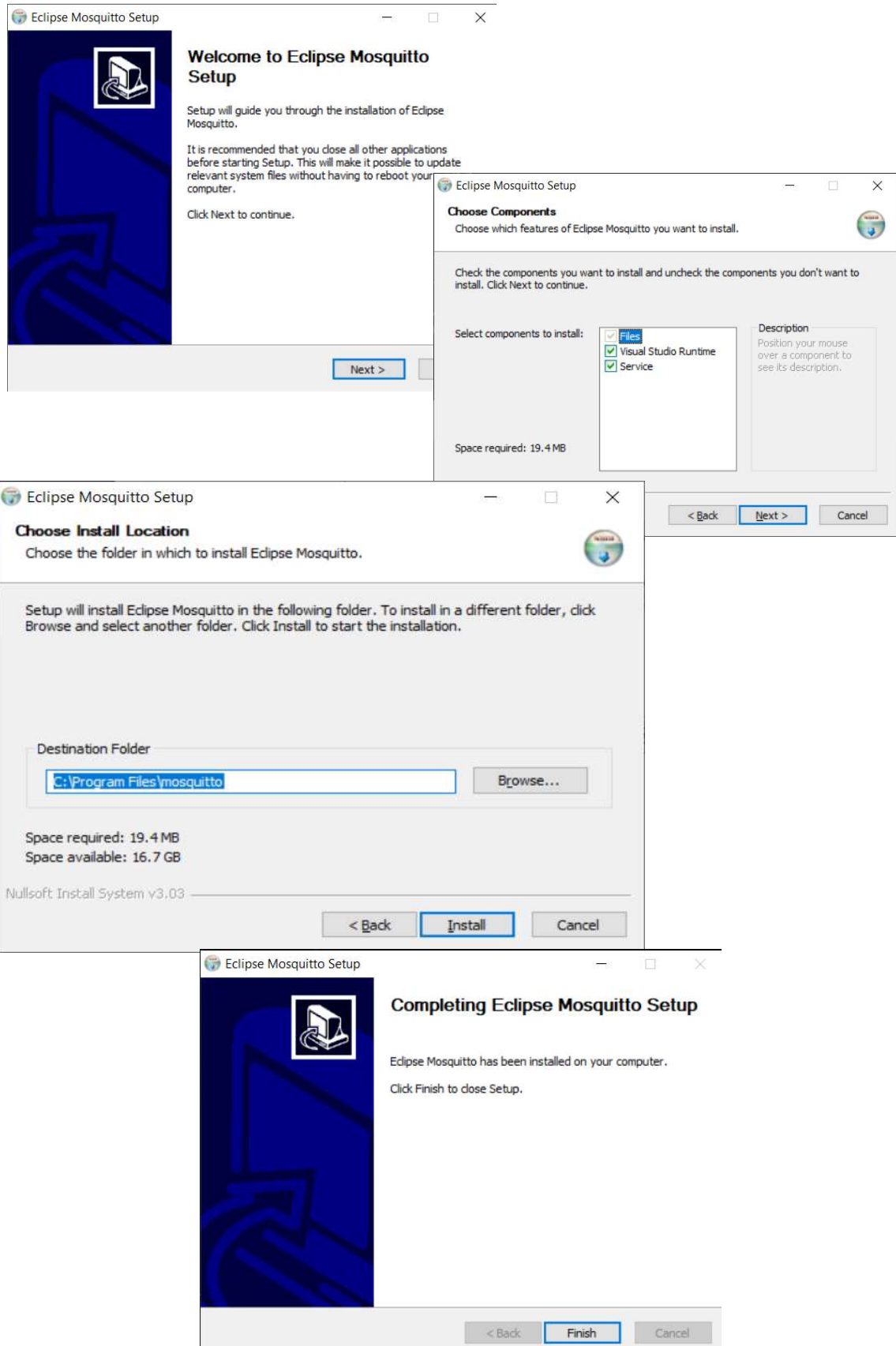
Mac

Mosquitto can be installed from the homebrew project. See brew.sh and then use `brew install mosquitto`



Once downloaded, double click on the install executable file to start the installation process.

Impt: Click Yes to allow the mosquitto install file unknown publisher to make changes to your device.



2. Launch command prompt and go to C:\Program Files\mosquitto folder, execute the mosquitto -h command to check the mosquitto version and note the MQTT broker version available.

```
(base) C:\Program Files\mosquitto>mosquitto -h
mosquitto version 2.0.14
mosquitto is an MQTT v5.0/v3.1.1/v3.1 broker.

Usage: mosquitto [-c config_file] [-d] [-h] [-p port]

-c : specify the broker config file.
-d : put the broker into the background after starting.
-h : display this help.
-p : start the broker listening on the specified port.
    Not recommended in conjunction with the -c option.
-v : verbose mode - enable all logging types. This overrides
    any logging options given in the config file.
```

3. Boot up the Rpi. If the e-paper of the RPi displays the SSID on boot up, use the information provided as displayed. See below for sample display.



If however, if the e-paper does not display the SSID, once the Rpi is booted up as an AP, connect to the Rpi via USBserial cable and putty.exe. Once connected, execute the following commands:

```
iwconfig
```

Note that the wlan1 interface is used as AP. Connect the LabPC or laptop to the respective SSID as shown (see highlighted portion). Note that the SSID will be different from that shown below.

```
wlan1 IEEE 802.11bg ESSID:"SCSC_IOTX" Nickname:"<WIFI@REALTEK>"
Mode:Master Frequency:2.437 GHz Access Point: 34:C9:F0:90:42:E1
Bit Rate:54 Mb/s Sensitivity:0/0
Retry:off RTS thr:off Fragment thr:off
Power Management:off
Link Quality=56/100 Signal level=80/100 Noise level=0/100
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:0 Missed beacon:0
```

- Once connected to the Rpi AP, run the mosquitto subscribe command to connect to the mosquitto broker and listen for messages on the specific topic. This is referred to as subscribing to a topic. Note that the topic to be subscribing to is "test". To execute the command, refer to item 2.

```
cd c:\Program Files\mosquitto

mosquitto_sub -h "192.168.23.1" -t "test"
```

- Launch the subscriber command prompt. The command prompt shows the following command: `mosquitto_sub -h "192.168.23.1" -t "test"`. The command is used to subscribe to a message to the subscriber.

```
cd c:\Program Files\mosquitto

mosquitto_pub -h "192.168.23.1" -t "test" -m "hello-world"
```

```
C:\Program Files\mosquitto>mosquitto_pub -h "192.168.23.1" -t "test" -m "hello-world"
```

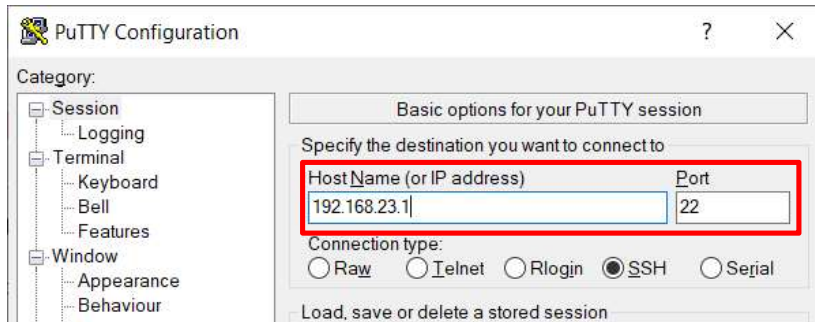
- Note that the message will be received by the subscriber, subscribing to the respective topic

```
(base) C:\Program Files\mosquitto>mosquitto_sub -h "192.168.23.1" -t "test"
hello-world
```

The message is displayed by the subscriber command.

Part 2: Endpoint to Gateway Data Transfer with MQTT

1. Using the existing putty session or reconnect to a new putty session using SSH to Rpi IP address.



Once logged in to console terminal, execute the following commands to activate the python virtual environment so as to start up the Django WebServer.

```
source ege356_config.sh
```

```
pi@egegaw-S01:~ $ source ege356_config.sh
(ege356-labs) pi@egegaw-S01:~/ege356 $
```

Python virtual
environment

Execute the shell script to start the Django WebServer.

```
sh ege356_lab4.sh
```

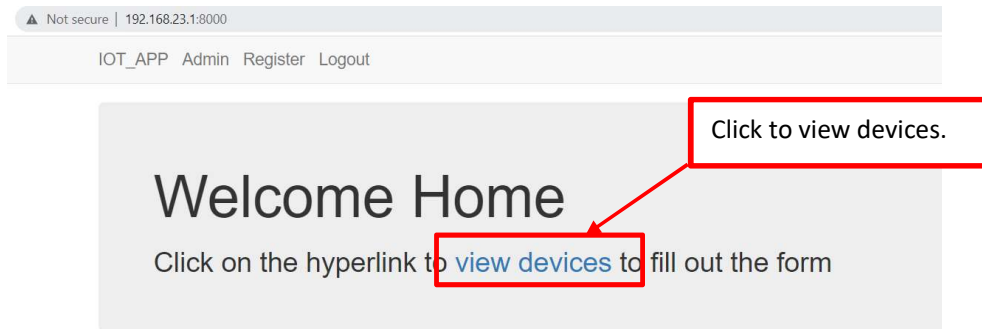
Execute shell script

```
(ege356-labs) pi@egegaw-S01:~/ege356 $ sh ege356_lab4.sh
/home/pi/ege356
Watching for file changes with StatReloader
Performing system checks...

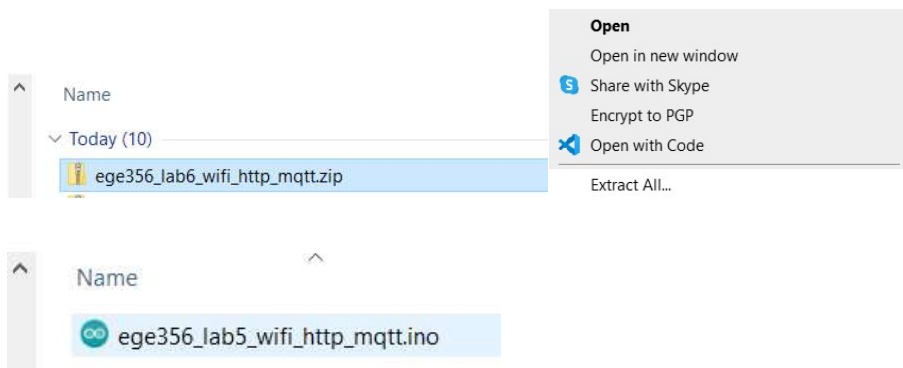
System check identified no issues (0 silenced).
May 25, 2022 - 09:02:52
Django version 3.2.10, using settings 'deviceweb service.settings'
Starting development server at http://192.168.23.1:8000/
Quit the server with CONTROL-C.
```

WebServer successfully
started

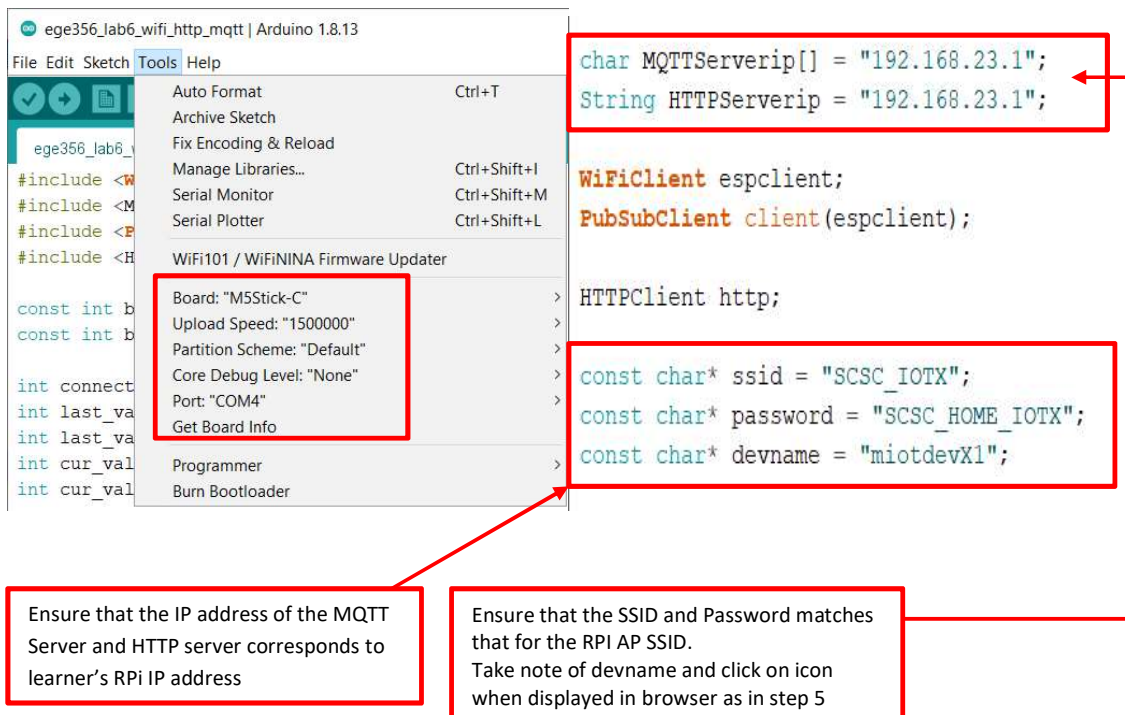
- Check to ensure that the webserver is up by accessing the web application via the web-browser. Login to the web application to view the devices.



- Download the zip file from BrightSpace. Extract the zip file as shown.



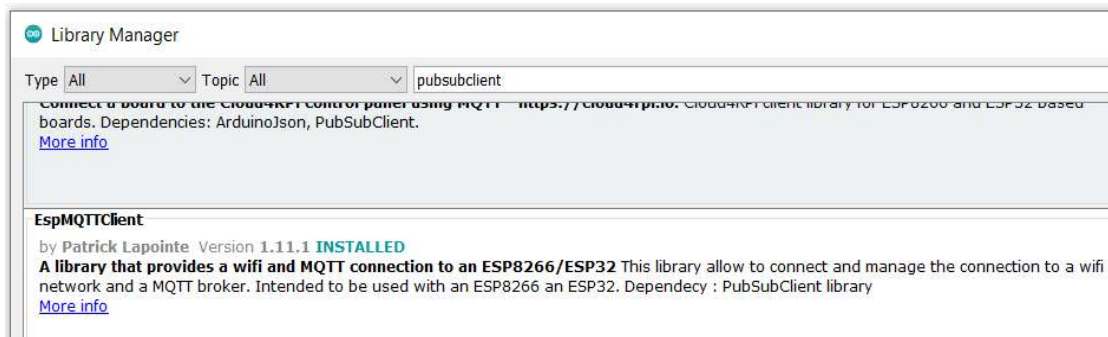
- Connect up the M5StickC device, launch the Arduino file, and load the program to M5StickC device.



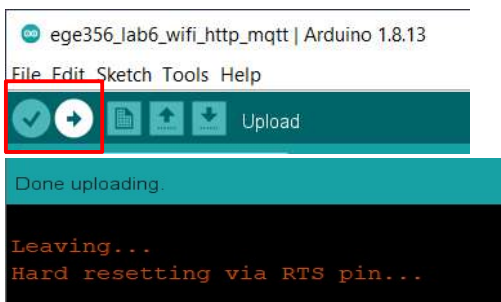
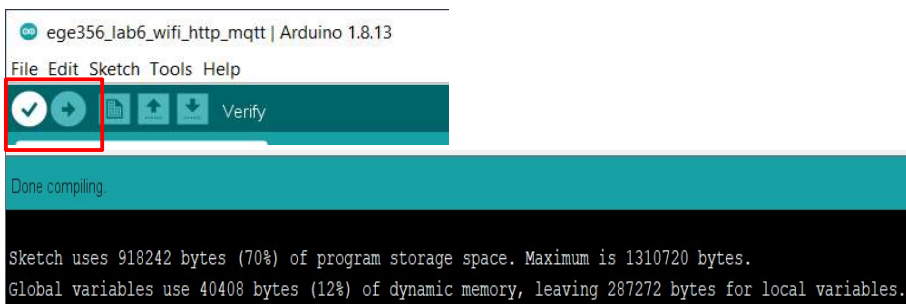
- Check “Managed Libraries” to ensure that “PubSubClient” is installed.



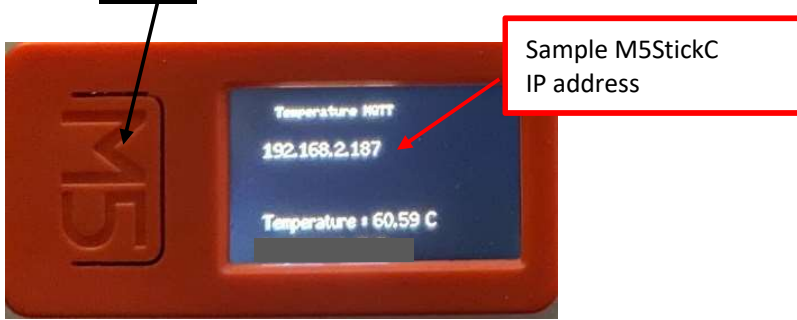
If required, install EspMQTTClient as well.



- Compile (Verify) and Load the code to the M5StickC device.



7. Click on Button A to connect to the AP Wifi network.



8. Note that once connected, the M5StickC device also sends a MQTT message “hello world” to the topic “test”. Note that the Web Application has been also programmed to receive MQTT messages and subscribes to the topic “test”. The MQTT server that the Web Application connects to is 192.168.23.1.

```
pi@egeg-S01: ~/ege356
{'_id': ObjectId('628fb32bc6969804b0b4cfd'), 'name': 'miotdevX1', 'datetime': d
atetime.datetime(2022, 5, 26, 17, 4, 43, 59000), 'status': 'updated'}
[26/May/2022 17:07:55] "GET /viewdevices/ HTTP/1.1" 200 7911
Message Received: test,hello world

Message Received: test,hello world

Message Received: test,hello world
```

9. At the Web Application Devices Page, click on miotdevX1 device icon to be redirected to the MQTT.

Devices Page

Fri May 27 2022 01:20:12 GMT+0800 (Singapore Standard Time)

Devices		
mttestiotX34 March 24, 2022, 3:49 a.m.	miotdevX0 April 9, 2022, 12:08 p.m.	iotdevX0 April 9, 2022, 12:34 p.m.
miotdevX1 April 12, 2022, 7:33 a.m.	miotdev_rouge April 20, 2022, 8:04 a.m.	miotdevX1T April 21, 2022, 2:37 p.m.
miotdev_rouge2 April 22, 2022, 12:09 a.m.	iotdevX May 17, 2022, 5:20 a.m.	



← → ↻ ⚠ Not secure | 192.168.23.1:8000/messageview/

MQTT Data Page

Fri May 27 2022 01:22:51 GMT+0800 (Singapore Standard Time)

0.0 degrees C

Normal

Activate

Deactivate

10. Now click on Button B. Observe that the MQTT Data Page will display the updated temperature data from M5StickC.



Button B

← → ↻ ⚠ Not secure | 192.168.23.1:8000/messageview/

MQTT Data Page

Fri May 27 2022 10:43:42 GMT+0800 (Singapore Standard Time)

53.00 degrees C

Normal

Activate

Deactivate

```
pi@egeg-w-S01: ~/ege356
Message Received: test/sens1,Normal!
Message Received: test,hello world
Message Received: test/temp1,53.02
message temp1: 53.02
Message Received: test/sens1,Normal!
xdata:<function xdata at 0xb57cba08>
ydata:<function ydata at 0xb57cba50>
getxdata:<function xdata at 0xb57cba08>
getydata:<function ydata at 0xb57cba50>
print data1:53.00
print data2:Normal
[27/May/2022 02:43:41] "GET /messageview
```

11. The **python module** that enabled the mqtt communication between M5StickC device and Rpi is provided by **mqttdev.py**. Download this file from BrightSpace and answer the following questions with reference to **mqttdev.py**.

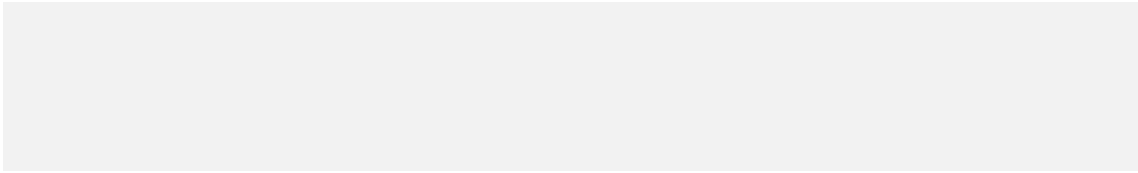
Open the mqttdev.py file with any IDE (Atom, Visual Code, etc) or simply use notepad.

Which function executes the connection to the MQTT server? Specify the MQTT Server IP Address and the TCP Port used.

What is the purpose of client.on_connect?

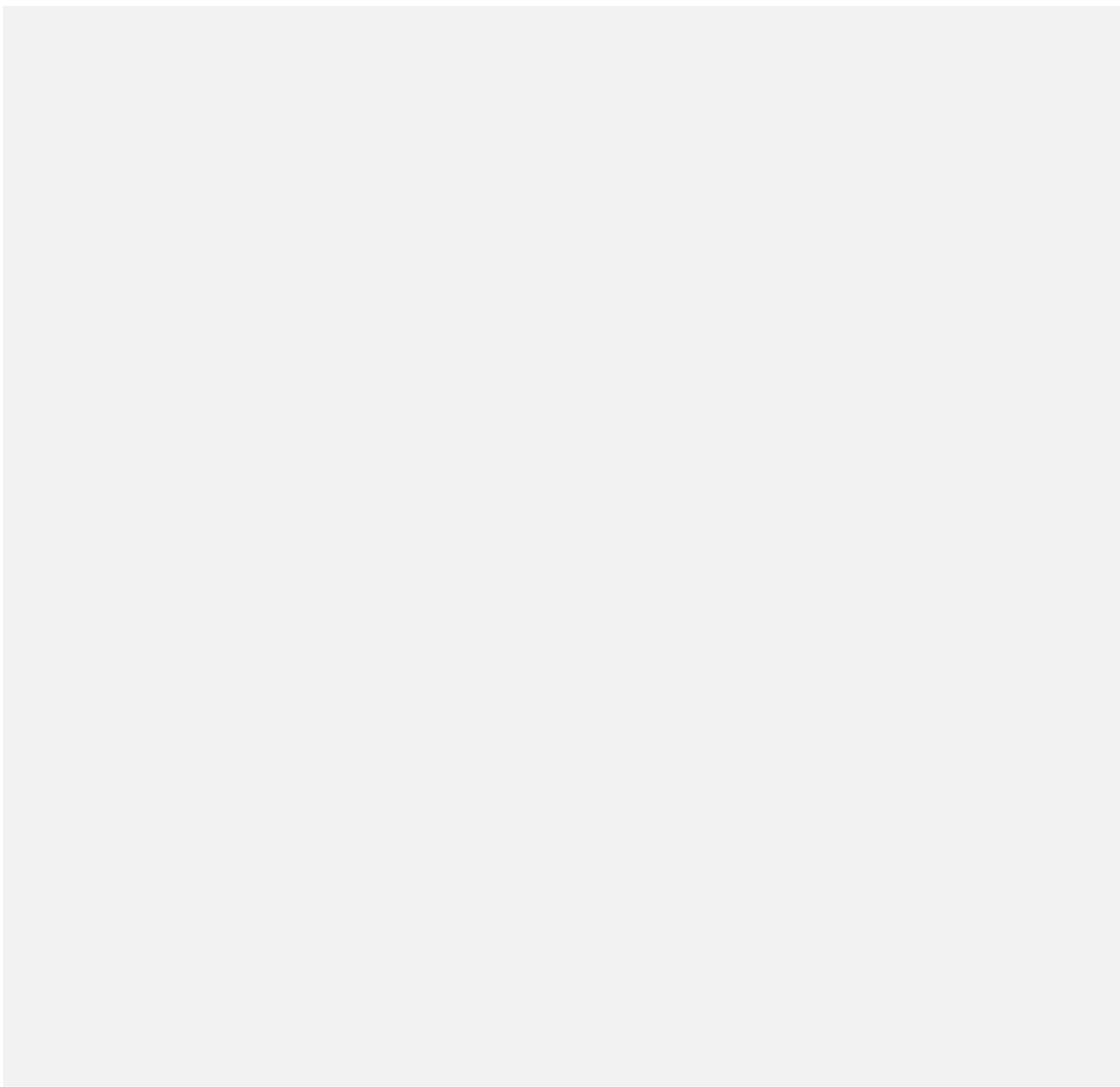
What is the purpose of client.on_message?

What is the purpose of *client.loop_forever()*?



12. Refer to Arduino program for M5StickC.

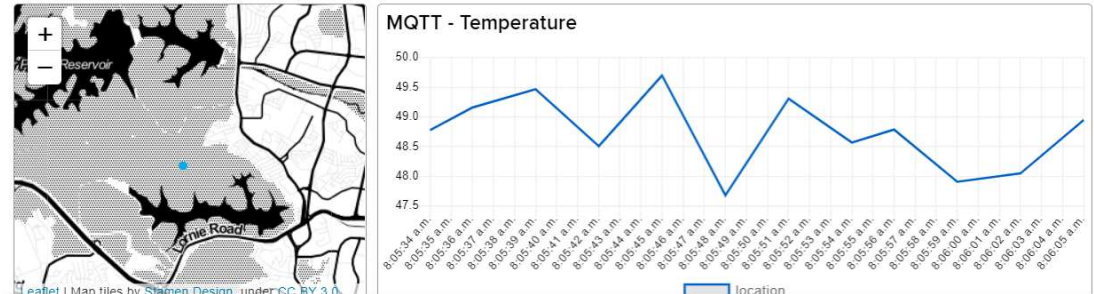
With regards to MQTT Server Connection, Subscription and Publishing data, what happens when button A is pressed? Write down the codes relevant to MQTT connection.



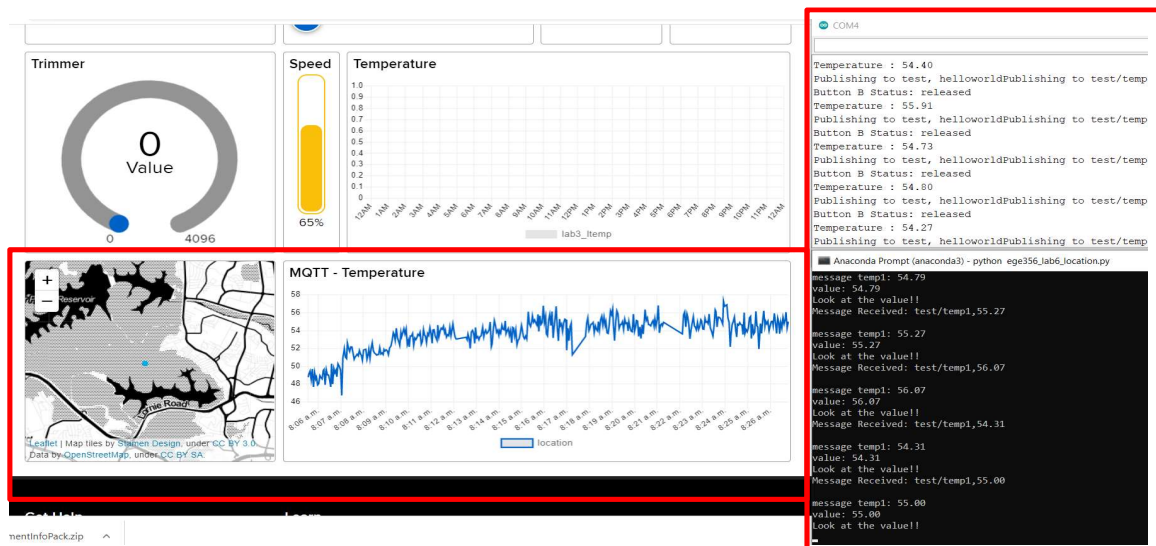
13. Summary: MQTT clients that subscribe to data are connected to the MQTT server and they stay in the listening loop waiting to receive data from the topic that they have subscribed to. Once they receive the data, the data can then be read and processed.

Part 3: Endpoint to Gateway Data Transfer to Cloud (Assignment 2 – Part 1)

- In this Assignment, learners are required to create the “location” feed. This feed is used by the Map block and Line Chart block as shown below. The MQTT Temperature will display the temperature captured from M5StickC device via MQTT and the Map will display a location in Singapore.

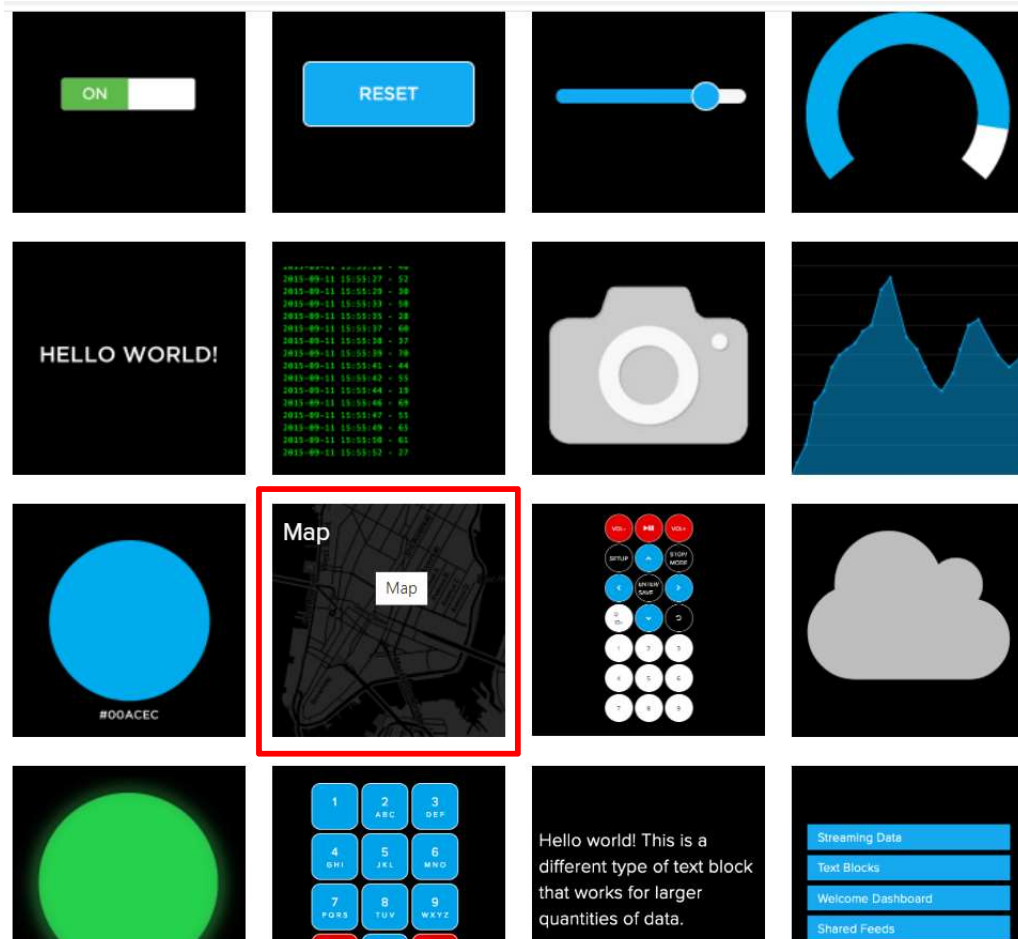


- This Assignment has **2 tasks** to be completed as described below:
 - Task 1** - Download and modify the ege356_Lab6_location.py file to send a fixed GPS location data to AdafruitIO, and a random data ranging between 40-70 to the Line Chart labelled MQTT-Temperature. (40 marks)
Execute the python script and capture a screenshot of the data as shown.
 - Task 2** - Create a copy of the file ege356_Lab6_location.py and rename it as ege356_assignment2part1_admin#.py. (60 marks)
 - Modify the file so that it is able to capture temperature from M5StickC via MQTT and then send the data to Adafruit IO Cloud. Note that to do this, both the laptop/PC running the python script and the M5StickC need to be connected to the RPi Access Point wifi network. Both the M5StickC and Laptop/PC are connected to the same MQTT server. The laptop/PC should send the data to Adafruit IO Cloud each time it receives the data via MQTT. The data sent to the Adafruit IO Cloud should be displayed using the MQTT – Temperature Chart as shown below.
 - The static GPS Location should be sent to Adafruit IO Cloud once each time the python script runs.

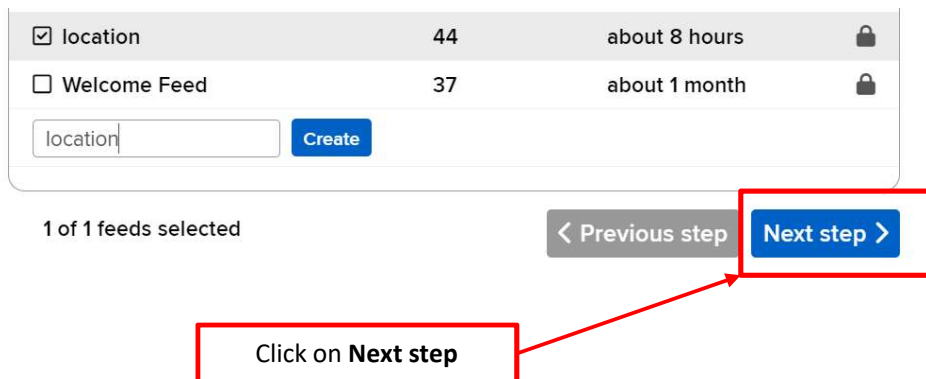


3. Follow the instructions (items 3 – 6) below to complete **Task 1**:

Login to the Adafruit IO account and create the Map Block and the Line Chart Block. To create the Map Block, select the Map as highlighted in the diagram below.



Create the “location” feed, and select the “location” feed to be used with the Map Block.



Block settings



In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

Block Preview

Hours of History (0 for realtime)

Light Mode Map Type

Dark Mode Map Type

Map A map can track the locations of your feed data.

< Previous step

Create block

Click on **Create block**

Perform similar steps to add the Line Chart and link it with the same "location" feed.

Edit the ege356_Lab6_location.py.

```
# Import standard python modules
```

```
import time
```

```
import random
```

Add the following code line, for use to create random numbers

```
# Import Adafruit IO REST client.
```

```
from Adafruit_IO import Client, Feed, RequestError
```

```
# Set to your Adafruit IO key.
```

```
# Remember, your key is a secret,
```

```
# so make sure not to publish it when you publish this code!
```

```
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'
```

```
# Set to your Adafruit IO username.
```

```
# (go to https://accounts.adafruit.com to find your username)
```

```
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'
```

```
....
```

```
# We don't have a GPS hooked up, but let's fake it for the example/test:
```

```
# (replace this data with values from a GPS hardware module)
```

```

value = 40
lat = 40.726190
lon = -74.005334
ele = 6 # elevation above sea level (meters)
.....
.....

metadata = { 'lat':lat, 'lon':lon, 'ele':ele, 'created_at':time.asctime(time.gmtime()) }
aio.send_data(location.key,value,metadata)

# limit feed updates to every 3 seconds, avoid IO throttle
loop_delay = 3

while True:
    value=random.randint(40,70)
    print('\nSending Values to location feed...\n')
    print('\tValue: ', value)
    print('\tLat: ', lat)
    print('\tLon: ', lon)
    print('\tEle: ', ele)
    # Send location data to Adafruit IO
    metadata = { 'lat':lat, 'lon':lon, 'ele':ele, 'created_at':time.asctime(time.gmtime()) }
    aio.send_data(location.key,value,metadata)
    # shift all values (for test/demo purposes)
    #value += 1
    #lat -= 0.01
    #lon += -0.02
    #ele += 1

    # Read the location data back from IO
    print('\nData Received by Adafruit IO Feed:\n')
    data = aio.receive(location.key)
    print('\tValue: {0}\n\tLat: {1}\n\tLon: {2}\n\tEle: {3}'
          .format(data.value, data.lat, data.lon, data.ele))
    # wait loop_delay seconds to avoid api throttle
    time.sleep(loop_delay)
  
```

GPS coordinates used for Map

Add the following code lines to send data to "location" feed to update Map location and Temperature value

Assign random generated value between 40 to 70 to "value" variable for each while loop.

Format location metadata and send updated "value" and metadata to "location" feed

Assignment to the following variables are not required, commented out.

4. Install the python package for Adafruit IO Cloud using the command below

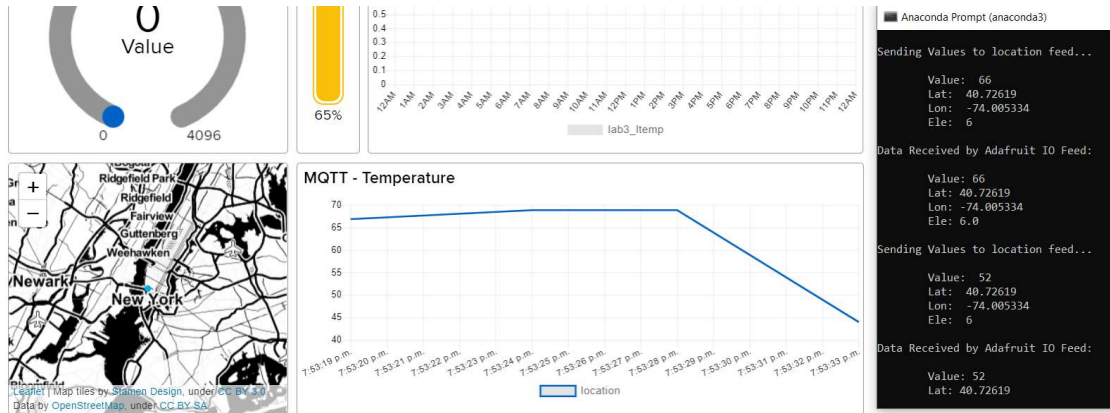
```
pip3 install adafruit-io
```

If necessary, the following packages may be required as well.

```
pip3 install setuptools
pip3 install wheel
```

**** Impt:** Note that depending of python environment and version used, execute the command using pip or pip3 or both.

- Execute the python script using command prompt, or python IDLE or visual code environment.



- Observe that only the temperature value is updated constantly. The Map Location is only updated once.
- Use the information provided and the hints to complete **Task 2**:

For this task, modify the file **ege356_assignment2part1_admin#.py**, which is a copy of task 1's python file (**ege356_Lab6_location.py**). Refer to code lines from the file **mqttdev.py** for codes related to MQTT.

Use the following pip command to install paho-mqtt python package if the laptop/PC does not already have the package installed.

```
pip3 install paho-mqtt
```

**** Impt:** Note that depending of python environment and version used, execute the command using pip or pip3 or both.

Hints to complete this task:

- Use the MQTT client connection functions found in **mqttdev.py**.
- Each time a message is received, one of the MQTT functions is executed. Identify the specific function, and modify this function to send the received data to Adafruit IO Cloud each time MQTT data is received.
- Based on (ii), the while loop is not required, instead rely on the mqtt client loop, each time MQTT data is received, update the data to Adafruit IO Cloud.
- Note that the rate that MQTT data is sent from M5StickC device exceeds that of Adafruit IO and will cause the site to throttle data sent to Adafruit IO Cloud. Ensure that data sent to the cloud is sent only once every 3 seconds.

- v) Modify the GPS location variables with any location coordinates within Singapore. Note that this location code only needs to be sent once.
- vi) Refer to the video provided in BrightSpace for a demo on the expected output display from Adafruit IO, the Arduino serial output and the python print output display.