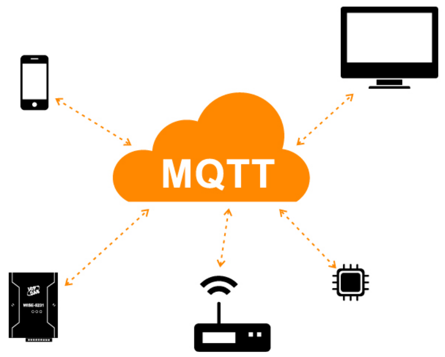
LoRaWAN®



ChirpStack

Node-RED

ArduinoJson

mosquitto

{JSON}
JavaScript Object Notation

# Innovation And Embedded System (539305)

ภาคการศึกษาที่ 2 ปีการศึกษา 2564

อาจารย์ ดร.มาโนทย์ มาปะโท

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีสุรนารี

Suranaree University of Technology

# Course Outline 539305 2(0-6-9)

| Catagory | Item | Topic | Week | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | |
| | | | 0.1 | 0.2 | 1.1 | 1.2 | 2.1 | 2.2 | 3.1 | 3.2 | 4.1 | 4.2 | 5.1 | 5.2 | 6.1 | 6.2 | 7.1 | 7.2 | 8.1 | 8.2 | 9.1 | 9.2 | 10.1 | 10.2 | 11.1 | 11.2 |
| Introduction | 1 | Introduction (Outline, Score) | | | | | C | - | | | | | | | | | | | | | | | | | | |
| | 2 | Project | | | | | C | - | | | | | | | | | | | | | | | | | | |
| MQTT | 3 | MQTT Broker | | | | | | | C | L | | | | | | | | | | | | | | | | |
| | 4 | MQTT pub sub client | | | | | | | C | L | | | | | | | | | | | | | | | | |
| JSON | 5 | JSON | | | | | | | | | C | L | | | | | | | | | | | | | | |
| | 6 | Arduino JSON | | | | | | | | | C | L | | | | | | | | | | | | | | |
| Python MQTT | 7 | Python MQTT | | | | | | | | | | | C | L | | | | | | | | | | | | |
| | 8 | Pub sub | | | | | | | | | | | C | L | | | | | | | | | | | | |
| | 9 | Python MQTT Application | | | | | | | | | | | C | L | | | | | | | | | | | | |
| Python-MQTT Project | 10 | Work | | | | | | | | | | | | | M | - | | P | | | | | | | | |
| | 11 | Present | | | | | | | | | | | | | M | - | | P | | | | | | | | |
| LoRAWAN | 12 | LoRa | | | | | | | | | | | | | | | | | C | C | | | | | | |
| | 13 | LoRaWAN | | | | | | | | | | | | | | | | | C | C | | | | | | |
| | 14 | LoRaWAN practical | | | | | | | | | | | | | | | | | C | C | | | | | | |
| Node-red | 15 | Node-red | | | | | | | | | | | | | | | | | | | C | L | | | | |
| | 16 | JS | | | | | | | | | | | | | | | | | | | C | L | | | | |
| | 17 | Node-red LoRaWAN application | | | | | | | | | | | | | | | | | | | C | L | | | | |
| Node-red Project | 18 | Start | | | | | | | | | | | | | | | | | | | | | F | - | | P |
| | 19 | Update | | | | | | | | | | | | | | | | | | | | | F | - | | P |
| | 20 | Final | | | | | | | | | | | | | | | | | | | | | F | - | | P |

C = Class
L = Lab
M=Midterm

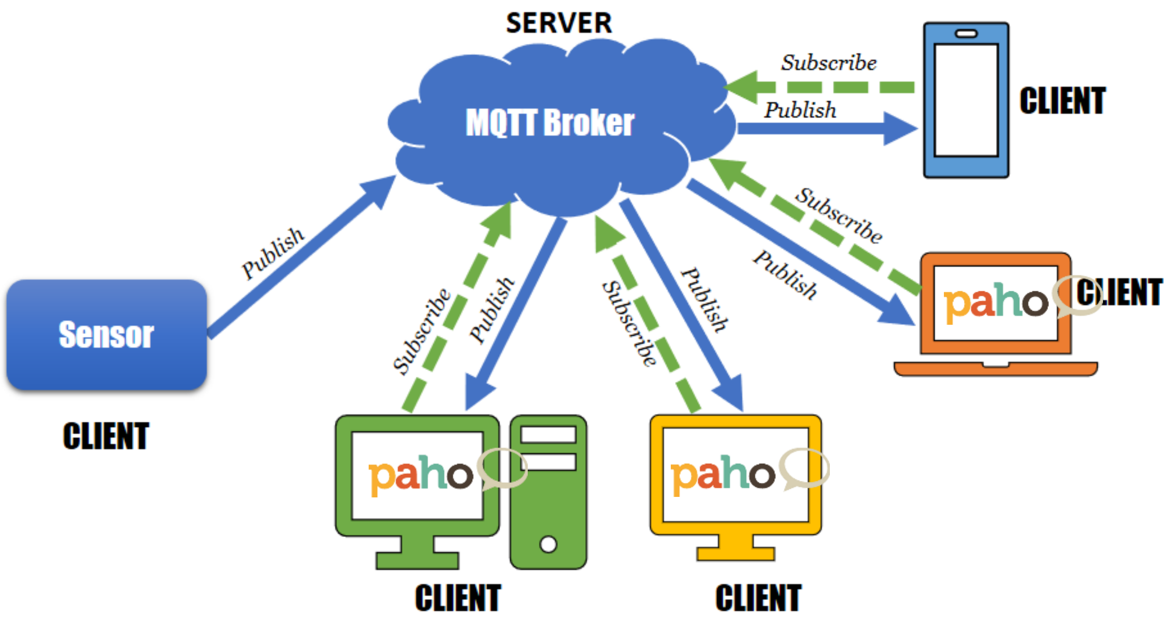
P = Present
F= Final

## Outline Week04

- ❑ Python paho-mqtt
- ❑ Data simulation
- ❑ MQTT+ pyQT GUI
  - Monitoring
  - Hardware control

## Python MQTT client (paho-mqtt)

- ❑ paho-mqtt is  Eclipse Paho MQTT Python client library, which implements versions 5.0, 3.1.1, and 3.1 of the MQTT protocol.
- ❑ paho-mqtt 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.
- ❑ It supports Python 2.7.9+ or 3.6+.

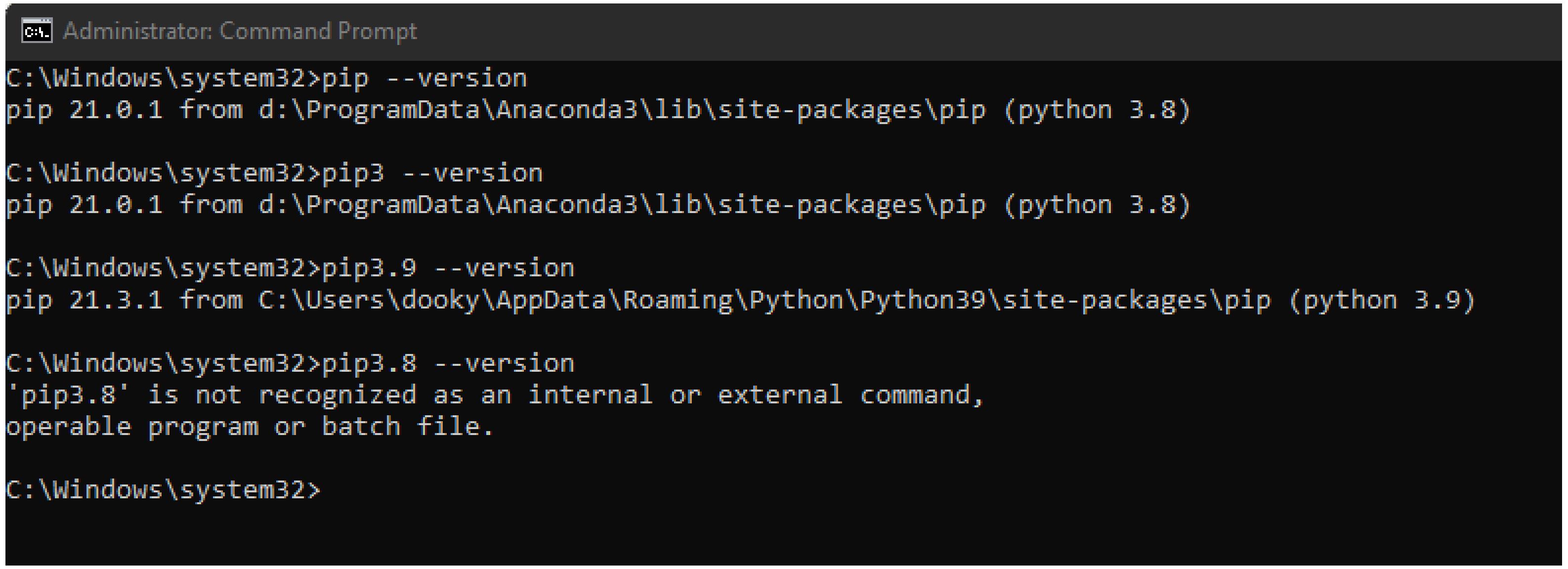


## paho-mqtt

- Installation

```
pip install paho-mqtt
```

- ในกรณีที่มีการติดตั้ง python หลายเวอร์ชั่น  หรือบน environment ที่ต่างกัน เช่น 3.9 แบบปกติ หรือ 3.9 บน Anaconda ให้ทำการตรวจสอบตำแหน่งที่ระบบจะทำการติดตั้งก่อน โดยใช้คำสั่ง pip3 --version หรือ pip3.9 --version
- หรือหากทราบว่ามี  python เวอร์ชั่นอื่น ก็ให้ตรวจสอบไปตาม Version ที่ทราบว่ามี
- หากไม่มีเวอร์ชั่นนั้นๆ จะขึ้นข้อความ is not recognized as an internal or external command,
- จากตัวอย่างด้านล่างจะเห็นว่าหากต้องการติดตั้งไปที่ตำแหน่งที่ติดตั้งเวอร์ชั่น 3.9  ต้องใช้คำสั่ง pip3.9 install paho-mqtt



```
C:\Windows\system32>pip --version
pip 21.0.1 from d:\ProgramData\Anaconda3\lib\site-packages\pip (python 3.8)

C:\Windows\system32>pip3 --version
pip 21.0.1 from d:\ProgramData\Anaconda3\lib\site-packages\pip (python 3.8)

C:\Windows\system32>pip3.9 --version
pip 21.3.1 from C:\Users\dooky\AppData\Roaming\Python\Python39\site-packages\pip (python 3.9)

C:\Windows\system32>pip3.8 --version
'pip3.8' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\system32>
```

http://www.steves-internet-guide.com/into-mqtt-python-client/

## ตัวอย่างรายละเอียดของคำสั่ง https://pypi.org/project/paho-mqtt/

**Client(client_id="", clean_session=True, userdata=None, protocol=MQTTv311, transport="tcp")**

**client_id**
the unique client id string used when connecting to the broker. If client_id is zero length or None, then one will be randomly generated. In this case the clean_session parameter must be True.

**clean_session**
a boolean that determines the client type. If True, the broker will remove all information about this client when it disconnects. If False, the client is a durable client and subscription information and queued messages will be retained when the client disconnects.

**userdata**
user defined data of any type that is passed as the userdata parameter to callbacks. It may be updated at a later point with the user_data_set() function.

**protocol**
the version of the MQTT protocol to use for this client. Can be either MQTTv31, MQTTv311 or MQTTv5
transport
set to “websockets” to send MQTT over WebSockets. Leave at the default of “tcp” to use raw TCP.

## คำสั่งที่ใช้งานบ่อย

```
❑ client(client_id="", clean_session=True, userdata=None, protocol=MQTTv311, transport="tcp")
```

client_id ให้ใส่ชื่อของ client ถ้าไม่ใส่จะใช้วิธีการสุ่มอัตโนมัติ
ตัวอย่างการใช้งาน

```
import paho.mqtt.client as mqtt
mqttc = mqtt.Client(“manot01")
```

```
❑ connect(host, port=1883, keepalive=60, bind_address="")
```

**Host**: the hostname or IP address of the remote broker
**Port**: the network port of the server host to connect to. Defaults to 1883.

```
broker_address="electsut.trueddns.com"
broker_port=27860
client = mqtt.Client(“manot01") client.connect(broker_address,broker_port)
```

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

## คำสั่งที่ใช้งานบ่อย

❑ `reconnect()`

Reconnect to a broker using the previously provided details. You must have called connect*() before calling this function.

❑ `disconnect()`

- Disconnect from the broker cleanly. Using disconnect() will not result in a will message being sent by the broker.
- Disconnect will not wait for all queued message to be sent, to ensure all messages are delivered, wait_for_publish() from MQTTMessageInfo should be used. See publish() for details.

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

## คำสั่งที่ใช้งานบ่อย

❑ `loop_start()`
❑ `loop_stop(force=False)`

These functions implement a threaded interface to the network loop. Calling loop_start() once, before or after connect*(), runs a thread in the background to call loop() automatically. This frees up the main thread for other work that may be blocking. This call also handles reconnecting to the broker. Call loop_stop() to stop the background thread. The force argument is currently ignored..

```
mqttc.connect("mqtt.eclipseprojects.io")
mqttc.loop_start()

while True:
    temperature = sensor.blocking_read()
    mqttc.publish("paho/temperature", temperature)
```

❑ `loop_forever(timeout=1.0, max_packets=1, retry_first_connection=False)`

- This is a blocking form of the network loop and will not return until the client calls disconnect(). It automatically handles reconnecting.
- Except for the first connection attempt when using connect_async, use retry_first_connection=True to make it retry the first connection. Warning: This might lead to situations where the client keeps connecting to an non existing host without failing.
- The timeout and max_packets arguments are obsolete and should be left unset.

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

## คำสั่งที่ใช้งานบ่อย

❑ `publish(topic, payload=None, qos=0, retain=False)`

This causes a message to be sent to the broker and subsequently from the broker to any clients subscribing to matching topics. It takes the following arguments:
**topic** the topic that the message should be published on
**payload** the actual message to send. If not given, or set to None a zero length message will be used. Passing an int or float will result in the payload being converted to a string representing that number. If you wish to send a true int/float, use struct.pack() to create the payload you require
**qos** the quality of service level to use
**retain** if set to True, the message will be set as the “last known good”/retained message for the topic.

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

## คำสั่งที่ใช้งานบ่อย

- ❑ `subscribe(topic, qos=0)`

- ❑ Subscribe the client to one or more topics.

Method

- ❑ **Simple string and integer**

e.g. subscribe("my/topic", 2)
**topic** a string specifying the subscription topic to subscribe to.
**qos** the desired quality of service level for the subscription. Defaults to 0.

- ❑ **List of string and integer tuples**

e.g. subscribe([("my/topic", 0), ("another/topic", 2)])

This allows multiple topic subscriptions in a single SUBSCRIPTION command, which is more efficient than using multiple calls to subscribe().
**topic** a list of tuple of format (topic, qos). Both topic and qos must be present in all of the tuples.
**qos** not used.

- ❑ `unsubscribe(topic)`

- ❑ Unsubscribe the client from one or more topics.

- ❑ **topic** a single string, or list of strings that are the subscription topics to unsubscribe from.**qos** not used.

## Callback

- ❑ Callback เป็นฟังก์ชันซึ่งจะถูกรันอัตโนมัติเมื่อมีการใช้งานคำสั่งต่างๆ
- ❑ สามารถบรรจุคำสั่งที่จะให้ทำงานเมื่อมีการใช้งานคำสั่งต่างๆได้
- ❑ ตัวอย่าง **on_message**

- ❑ **on_message**(client, userdata, message)

- ❑ Called when a message has been received on a topic that the client subscribes to and the message does not match an existing topic filter callback.
- ❑ Use message_callback_add() to define a callback that will be called for specific topic filters. on_message will serve as fallback when none matched.
- ❑ clien tthe client instance for this callback
- ❑ userdata the private user data as set in Client() or user_data_set()
- ❑ message an instance of MQTTMessage. This is a class with members topic, payload, qos, retain.

```python
def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)
```

## Callback

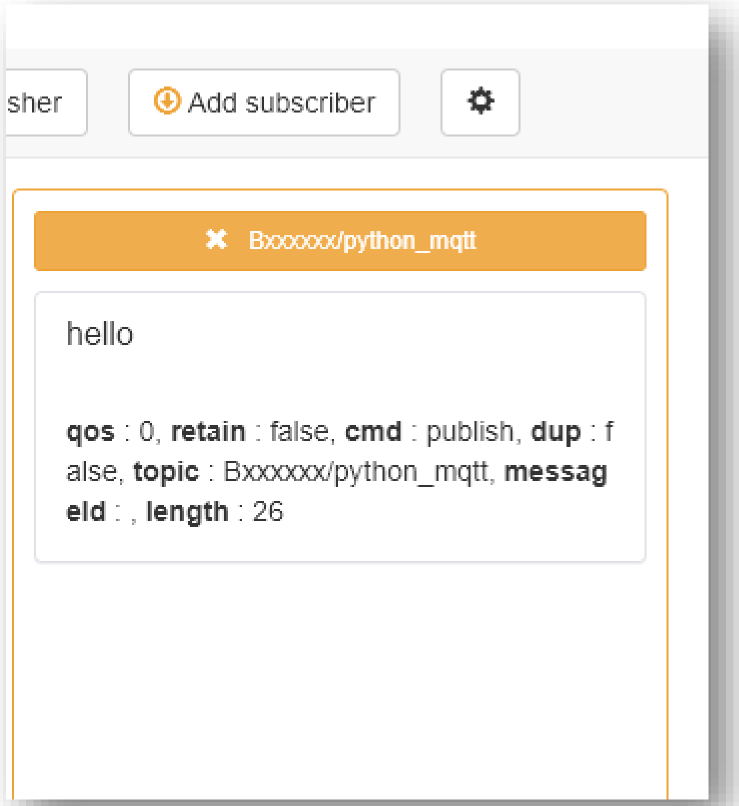
- ❑ คำสั่ง Callback มีให้เลือกใช้งานอีกจำนวนมากสามารถศึกษาเพิ่มเติมได้ที่ https://pypi.org/project/paho-mqtt/

- ❑ **on_message(client, userdata, message)**
- ❑ on_connect(client, userdata, flags, rc)
- ❑ on_disconnect(client, userdata, rc)
- ❑ on_publish(client, userdata, mid)
- ❑ on_subscribe(client, userdata, mid, granted_qos)
- ❑ on_unsubscribe(client, userdata, mid)
- ❑ on_socket_open(client, userdata, sock)
- ❑ on_socket_open(client, userdata, sock)
- ❑ on_socket_register_write(client, userdata, sock)
- ❑ on_socket_unregister_write(client, userdata, sock)

## ตัวอย่างการใช้งาน

จงเขียนโปรแกรมเพื่อเชื่อมต่อไปยัง MQTT broker ที่ใช้ในรายวิชานี้ โดยกำหนดชื่อ client เป็นรหัสนักศึกษา จากนั้นให้ publish ข้อความว่า hello  ผ่านไปยัง topic “Bxxxxxx/python_mqtt ( ให้ใช้รหัสนักศึกษาของตนเอง)

```python
import paho.mqtt.client as mqtt
broker_address="electsut.trueddns.com"
broker_port=27860
client = mqtt.Client("Bxxxxxx")
client.connect(broker_address,broker_port)
client.publish("Bxxxxxx/python_mqtt","hello")
```



hello

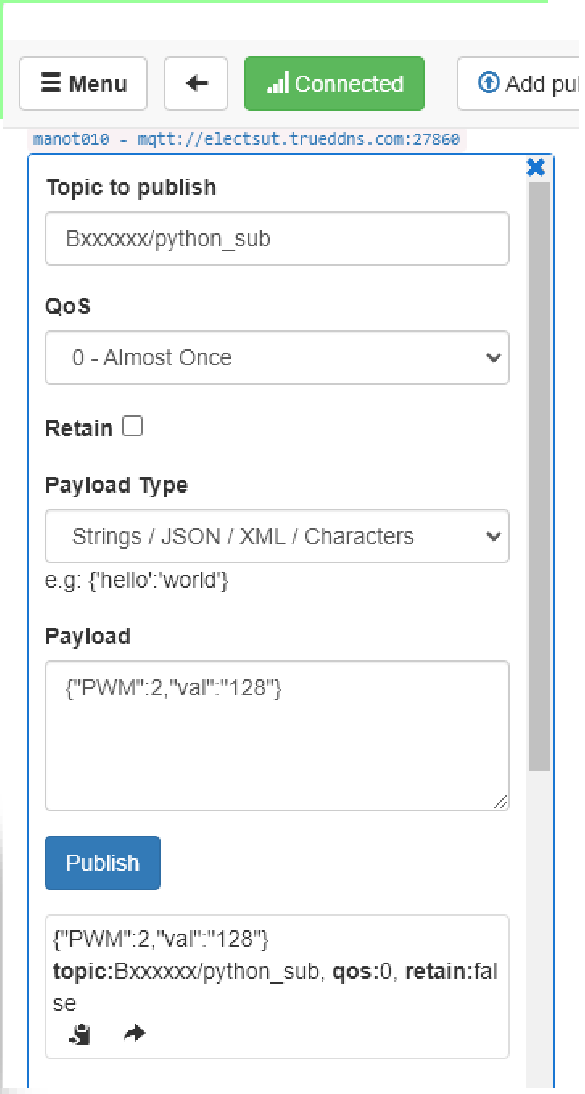
**qos** : 0, **retain** : false, **cmd** : publish, **dup** : f
alse, **topic** : Bxxxxxx/python_mqtt, **messag**
**eId** : , **length** : 26

## Publish/subscribe

จงเขียนโปรแกรมเพื่อเชื่อมต่อไปยัง MQTT broker ที่ใช้ในรายวิชานี้ โดยกำหนดชื่อ client เป็นรหัสนักศึกษา
- ให้ publish ข้อความว่า hello ผ่านไปยัง topic “Bxxxxxx/python_mqtt ( ให้ใช้รหัสนักศึกษาของตนเอง)
- ให้ subscribe ไปยัง topic “Bxxxxxx/python_sub ( ให้ใช้รหัสนักศึกษาของตนเอง)
- จากนั้นให้ทดลองใช้ MQTTbox publish ข้อความ “good morning” กลับมา

```python
import paho.mqtt.client as mqtt
broker_address="electsut.trueddns.com"
broker_port=27860
client = mqtt.Client("Bxxxxxx")
client.connect(broker_address,broker_port)

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

client.on_message=on_message #attach function to callback
client.publish("Bxxxxxx/python_mqtt","hello")
client.subscribe("Bxxxxxx/python_sub")
client.loop_start() #start the loop
time.sleep(10) # wait
client.loop_stop() #stop the loop
```

```
message received  {"PWM":2,"val":"128"}
message topic= Bxxxxxx/python_sub
message qos= 0
message retain flag= 0
```



manot010 - mqtt://electsut.trueddns.com:27860

Topic to publish: Bxxxxxx/python_sub
QoS: 0 - Almost Once
Retain ☐
Payload Type: Strings / JSON / XML / Characters
e.g: {'hello':'world'}
Payload: {"PWM":2,"val":"128"}
Publish

{"PWM":2,"val":"128"}
**topic**:Bxxxxxx/python_sub, **qos**:0, **retain**:false

## การใช้งาน client.loop_forever()

สามารถใช้งาน client.loop_forever() เพื่อให้โปรแกรมรันโดยไม่ออกจากโปรแกรมได้
หากต้องการหยุดต้องใช้คำสั่ง client.disconnect()

```python
import paho.mqtt.client as mqtt
broker_address="electsut.trueddns.com"
broker_port=27860
client = mqtt.Client("Bxxxxxx")
client.connect(broker_address,broker_port)

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)
    myPL=str(message.payload.decode("utf-8"))
    if myPL=="stop_loop":
        client.disconnect()

client.on_message=on_message #attach function to callback
client.publish("Bxxxxxx/python_mqtt","hello")
client.subscribe("Bxxxxxx/python_sub")
client.loop_forever()
```

## Python JSON

Python has a built-in package called json, which can be used to work with JSON data.
`Convert from JSON to Python: json.loads(x)`
`Convert from Python to JSON: json.dumps(x)`

```
# Example
# Convert from JSON to Python:

import json
# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["age"])
```

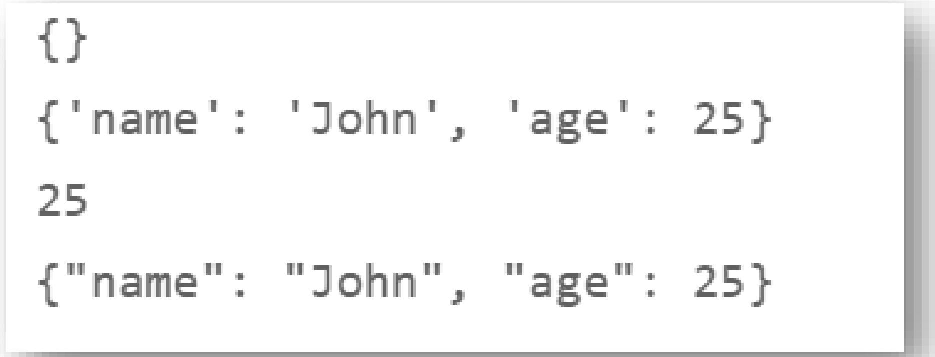
```
# Example
# Convert from Python to JSON:

import json

# a Python object (dict):
x = {
  "name": "John",
  "age": 30,
  "city": "New York"
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

w3schools.com-python_json/

## Python JSON: Assign/change value

สามารถกำหนดค่าและเปลี่ยนแปลงค่าที่อยู่ภายใน JSON ได้

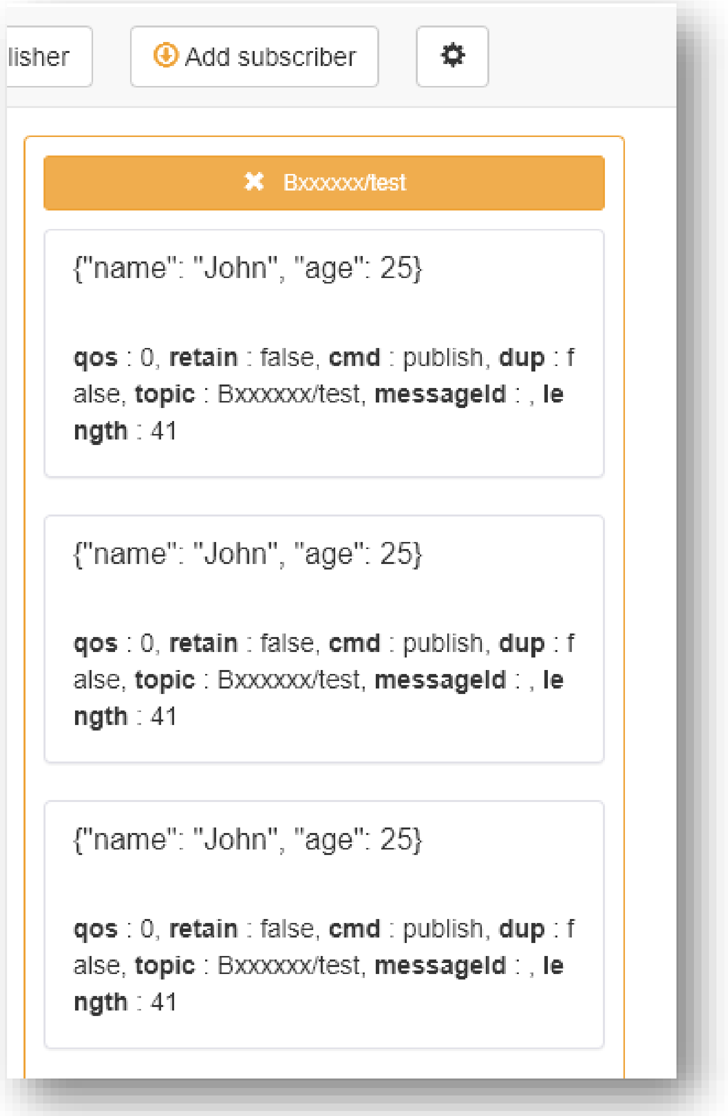
```
import json
x='{}'
s=json.loads(x)

print(s)
s["name"]="John"
s["age"]=25

print(s)

print(s["age"])

y=json.dumps(s)

print(y)
```



```
{}
{'name': 'John', 'age': 25}
25
{"name": "John", "age": 25}
```

## Python JSON test

```
import paho.mqtt.client as mqtt
import json

broker_address="electsut.trueddns.com"
broker_port=27860
client = mqtt.Client("P1")
client.connect(broker_address,broker_port)

x='{}'
s=json.loads(x)
print(s)
s["name"]="John"
s["age"]=25

y=json.dumps(s)

client.publish("Bxxxxxx/test",y)
```



```
{"name": "John", "age": 25}

qos : 0, retain : false, cmd : publish, dup : f
alse, topic : Bxxxxxx/test, messageId : , le
ngth : 41

{"name": "John", "age": 25}

qos : 0, retain : false, cmd : publish, dup : f
alse, topic : Bxxxxxx/test, messageId : , le
ngth : 41

{"name": "John", "age": 25}

qos : 0, retain : false, cmd : publish, dup : f
alse, topic : Bxxxxxx/test, messageId : , le
ngth : 41
```

## Python MQTT Example

# Backup

Arduino sub sensor
Arduino sub LED on-off

Arduino pub Led status
Arduino pub ADC val