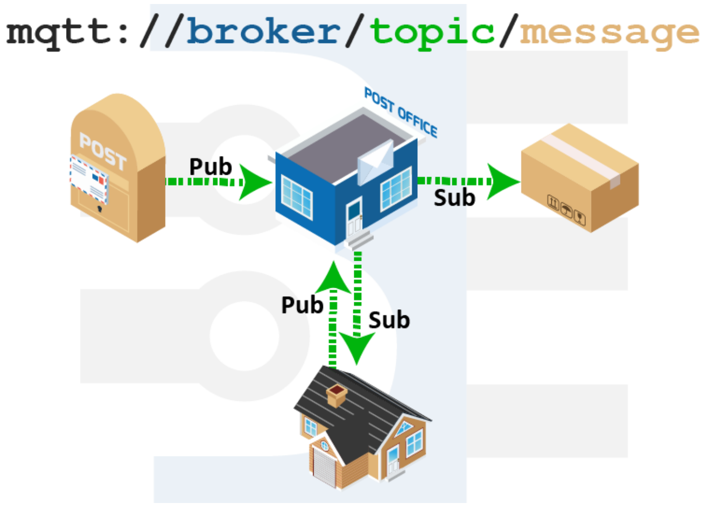
Message Queuing Telemetry Transport (MQTT)



The broker is the Post Office or Server.
Clients can publish messages, to the post office. Other clients can subscribe to topics, .or receive letters, from the post office.
Clients can both publish and subscribe.

Message Broker
A messaging broker system is a publish/subscribe protocol

Broker: The broker is a server. accepts messages from clients and then delivers to any interested clients. Messages belong to a topic.

Client: A device either publishes messages or subscribes to a topic, or both.

Topic: A namespace for messages on the broker. Clients subscribe and publish to a topic.

Publish: A client sending a message to the broker, using a topic name.

Subscribe: A client tells the broker which topics interest it. Once subscribed, the broker sends messages published to that topic.
A client can subscribe to multiple topics.

Unsubscribe: Tells the broker to stop sending messages on a topic.

MQTT message is tiny. A packet is like 2 bytes.

Send a message
temp: 22.5 on a topic heating/thermostat/living-room.

Subscribers listen to a topic.
heating/thermostat/living-room
would receive the message temp: 22.5.

Hosting an MQTT Broker

natively on a computer
inside a virtual machine
virtual machine in the cloud

dedicated cloud-based broker
single-board computer

mosquitto MQTT broker

```
apt install mosquitto mosquitto-clients
```

We can use the Raspberry Pi as MQTT client and broker. We can also use a computer for experiments

Enable the mosquitto broker to auto-start after reboot

```
systemctl enable mosquitto
sudo systemctl status mosquitto
```

Subscribe to the MQTT Topic Locally
A topic is a string that looks like a file system path. a/b/c/…

There are no restrictions on the number of elements. special characters: / + and # are wildcards

subscribe to the test/message topic:

```
mosquitto_sub -h localhost -t "test/message"
```

Publish to the MQTT Topic Locally
your current terminal is occupied listening to the topic.

Open another terminal
publish message to the test/message topic.
mosquitto_pub -h localhost -t "test/message" -m "Hello, MTTQ"

Identify the Raspberry Pi on the Network

The MQTT client need to know where it is on the network.

The MQTT client needs a hostname or an IP address to receive or publish a message.

hostname
ifconfig | grep inet | grep cast

Subscribe to test/message from remote client
mosquitto_sub -h 192.168.1.25 -t "test/message"

Publish a Test Message Remotely
mosquitto_pub -h 192.168.1.25 -t "test/message" -m "Hello from remote"

Simple MQTT client

```
import time
import paho.mqtt.client as paho

#broker="broker.hivemq.com"
broker="iot.eclipse.org"
```

```
#define callback
def on_message(client, userdata, message):
    time.sleep(1)
    print("received message:")
    print(str(message.payload.decode("utf-8")),
timex())

def timex():
    return str(int(time.time()))

#create client object
client= paho.Client("client-001")

#Bind function to callback
client.on_message=on_message

#client1.on_publish = on_publish
print("connecting to broker ",broker, timex())
client.connect(broker)#connect

#start loop to process received messages
client.loop_start()
print("subscribing ", timex())

client.subscribe("house/bulb1") #subscribe
time.sleep(2)

print("publishing ", timex())
client.publish("house/bulb1","on")#publish
time.sleep(5)
```

```
print("disconnecting" ,timex())
client.disconnect() #disconnect
client.loop_stop() #stop loop
```

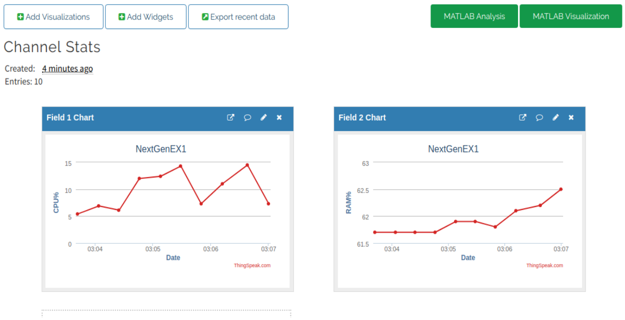
ThingSpeak

Create an account and channel in thingspeak.com . Get your api key and channel id.

```
import requests
import time
import random

channelID = "783907"
apiKey = "CD6R8LT5KK5Q"

#  MQTT Connection Methods
def sender():
  for x in range(20):
    v1=random.randint(0, 10)
    v2=random.randint(10, 20)
    payload = {'api_key': apiKey, 'field1':
str(x), 'field2': str(v2)}
    print (x,v1, v2)
    r =
requests.post('https://api.thingspeak.com/update',
params=payload)
    print ("Result: ", x, r.text)
    time.sleep(5)
sender()
```



Read Data

```
import requests
import time

READ_API_KEY='PJ14ZHK38H4OB'
CHANNEL_ID= '78307'
URL='http://api.thingspeak.com/channels/'+CHANNEL_ID+
'/feeds/last.json?api_key='+ READ_API_KEY

while True:
    resp = requests.get(URL)
    data = resp.json()
    a = data['created_at']
    b = data['field1']
    c = data['field2']
    print (a , "    " , b + "    " , c )
    time.sleep(5)
```

Sending Data with headers

```
headers = {'Content-type': 'application/json'}
```

```
r = requests.request("POST",
'https://api.thingspeak.com/update', data=payload,
headers=headers);
print (response);
time.sleep(10);
```

Using paho MQTT

```
import paho.mqtt.publish as publish
import psutil
import time

# ThingSpeak Channel ID
channelID = "783907"

# The Write API Key for the channel
apiKey = "CD6R4P9Z8LT5KK5Q"

#  MQTT Connection Methods
# use the default MQTT port of 1883
useUnsecuredTCP = False

# use MQTT over an unsecured websocket on port 80.
# Try this if port 1883 is blocked on your network.
useUnsecuredWebsockets = False

#use MQTT over a secure websocket on port 443.
# use more system resources,
# but the connection will be secured by SSL.
useSSLWebsockets = True

###   End of user configuration   ###

def timex():
    return str(int(time.time()))
```

```
# ThinSpeak MQTT service
mqttHost = "mqtt.thingspeak.com"

# Set up the connection parameters
if useUnsecuredTCP:
    tTransport = "tcp"
    tPort = 1883
    tTLS = None

if useUnsecuredWebsockets:
    tTransport = "websockets"
    tPort = 80
    tTLS = None

if useSSLWebsockets:
    import ssl
    tTransport = "websockets"
    tTLS =
{'ca_certs':"/etc/ssl/certs/ca-certificates.crt",'tls_v
ersion':ssl.PROTOCOL_TLSv1}
    tPort = 443

# Create the topic string
topic = "channels/" + channelID + "/publish/" + apiKey

# Run loop every 20 seconds
#  and published to a channel using MQTT.

while(True):
    print("starting ", timex())
    # get the system performance data
    cpuP = psutil.cpu_percent(interval=20)
    ramP = psutil.virtual_memory().percent
    print (" CPU =",cpuP,"   RAM =",ramP)
    # build the payload string
    tPayload = "field1=" + str(cpuP) + "&field2=" + 
str(ramP)

    # publish this data to the topic
    try:
        publish.single(topic, payload=tPayload, 
hostname=mqttHost, port=tPort, tls=tTLS, 
transport=tTransport)

    except (KeyboardInterrupt):
        break
    except:
        print ("There was an error publishing data.")
```

Flask IoT

https://www.hackster.io/ahmedibrrahim/smart-home-automation-iot-using-raspberry-pi-and-python-47fb62

https://www.google.com/amp/s/thingsmatic.com/2016/06/24/a-self-hosted-mqtt-environment-for-internet-of-things-part-1/amp/

https://projectiot123.com/2019/02/04/raspberry-pi-gui-based-home-automation-using-python/

Arduino
https://github.com/mathworks/thingspeak-arduino/tree/master/examples/ESP32

ESP32

https://appcodelabs.com/introduction-to-iot-how-to-build-a-simple-mqtt-subscriber-using-esp8266