

<https://www.halvorsen.blog>



MQTT

MQTT with Python

A Communication Protocol popular in Internet of Things Applications

Hans-Petter Halvorsen

Contents

- [MQTT Overview](#)
- [MQTT Brokers](#)
 - [HiveMQ Cloud](#)
- [MQTT Clients](#)
 - [MQTT X](#)
- [Python](#)
 - [MQTT Python Library](#)
 - [HiveMQ Cloud and Python Examples](#)
- [ThingSpeak](#)
 - [ThingSpeak and MQTT X Client](#)
 - [ThingSpeak and Python](#)

<https://www.halvorsen.blog>



MQTT

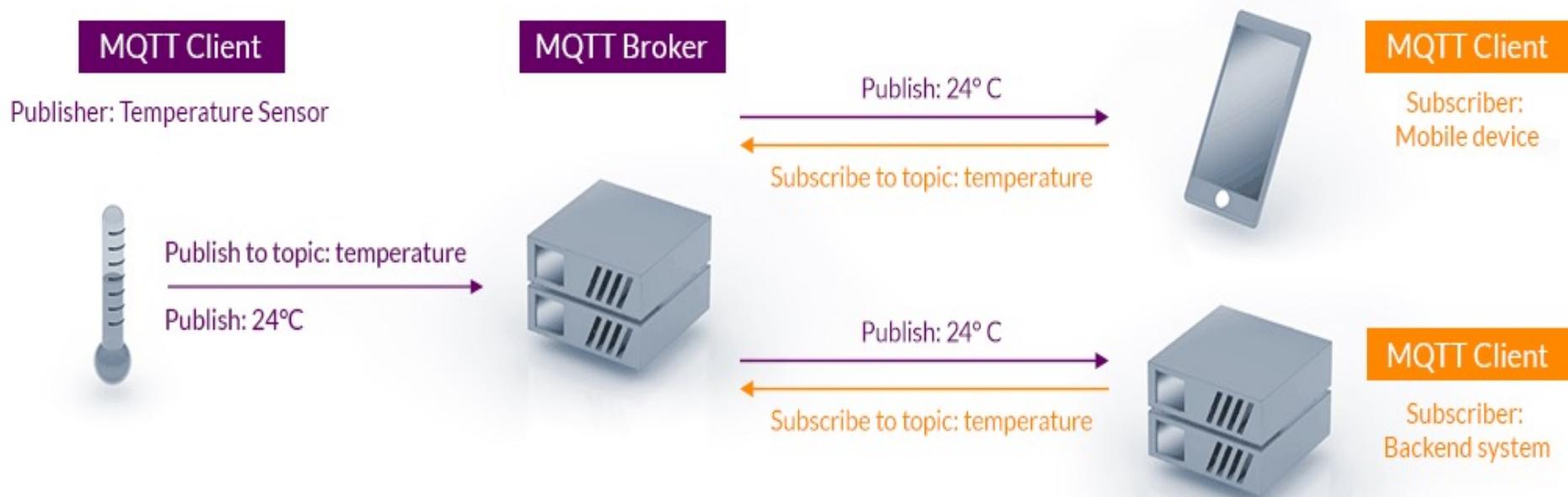
Hans-Petter Halvorsen

[Table of Contents](#)

MQTT

- MQTT is a Communication Protocol popular in Internet of Things (IoT) Applications
- <https://mqtt.org>
- You can use or implement MQTT in all the most popular Programming environments
- MQTT can be used on all the popular platforms like Windows, macOS, Linux, Arduino, Raspberry Pi
- You can use an existing API, or you can implement and use the MQTT protocol from scratch
- We will Python in this Tutorial

MQTT



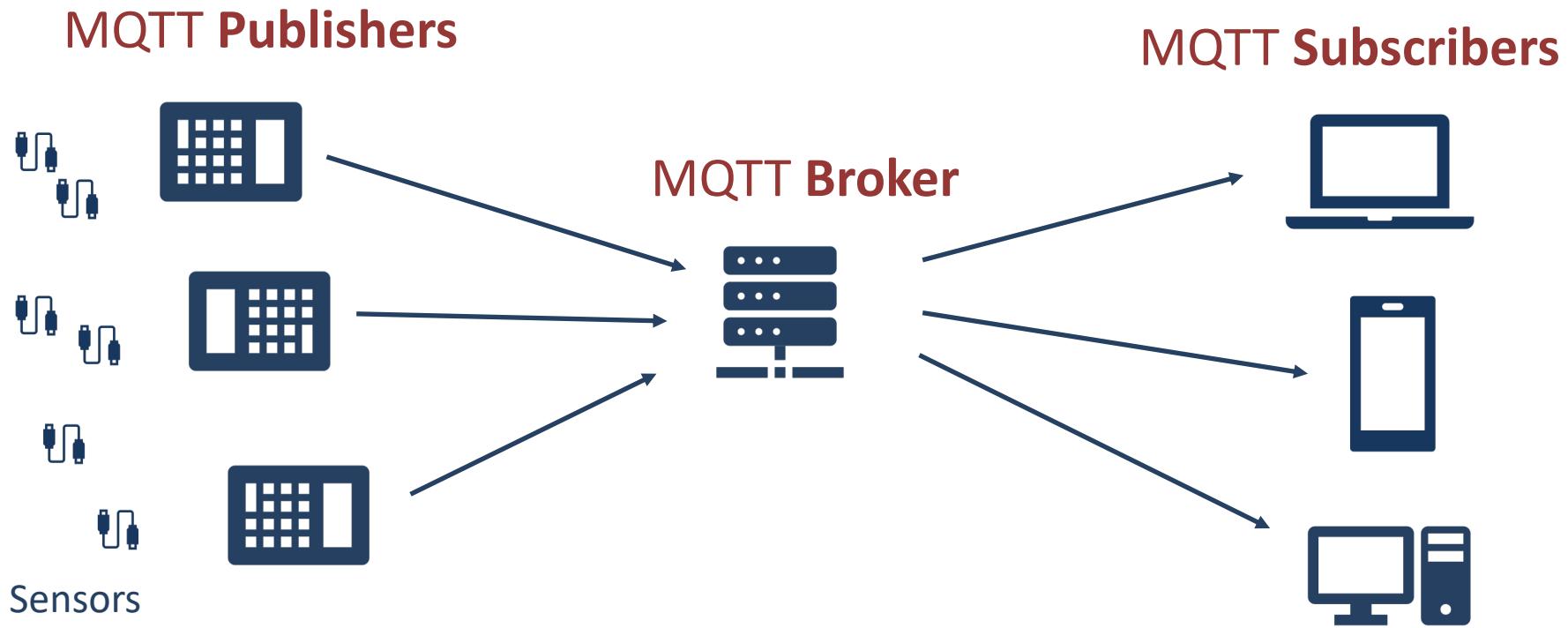
MQTT

- Message Queueing Telemetry Transport (MQTT) is an IoT connectivity protocol
- MQTT is used in applications with thousands of sensors
- MQTT is efficient in terms of bandwidth, battery, and resources
- **MQTT uses a publish/subscribe model**
- MQTT can be implemented using standard HTTP calls
- M2M (machine to machine) Communication

Internet of Things (IoT) and MQTT

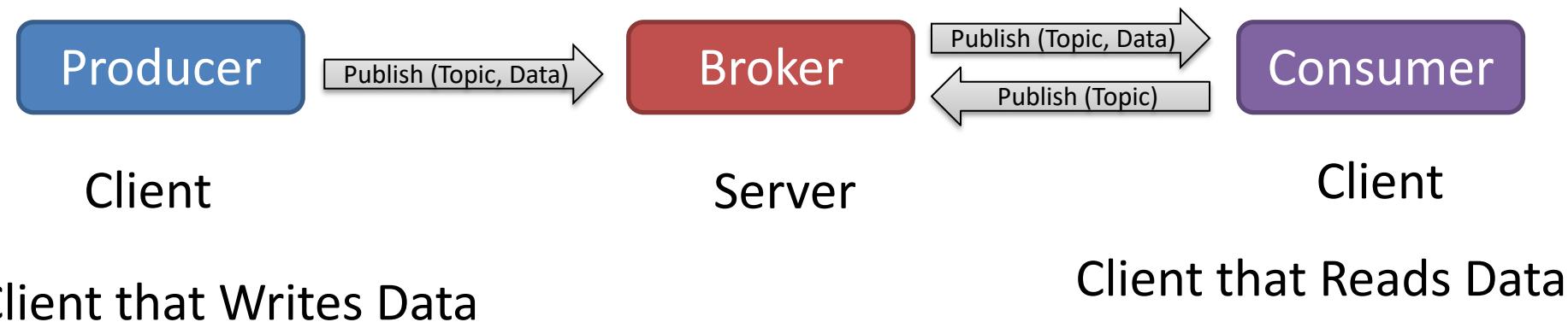
- Internet of Things (IoT): To get data to and from devices on a network.
- MQTT is a lightweight protocol that makes this easier

MQTT Scenario



Publish/Subscribe Model

Typically, we have what we call **Producers** (Publishers), and we have **Consumers**, which can be both Publishers and Subscribers.



MQTT Terms

- MQTT Broker
 - Server
- MQTT Publishers
 - Clients that Write/Publish Data
- MQTT Subscribers
 - Clients that Read/Subscribe to Data

MQTT Topics

- Data in MQTT are Published to Topics
- Topics are made up of one or more topic levels, separated by a forward slash

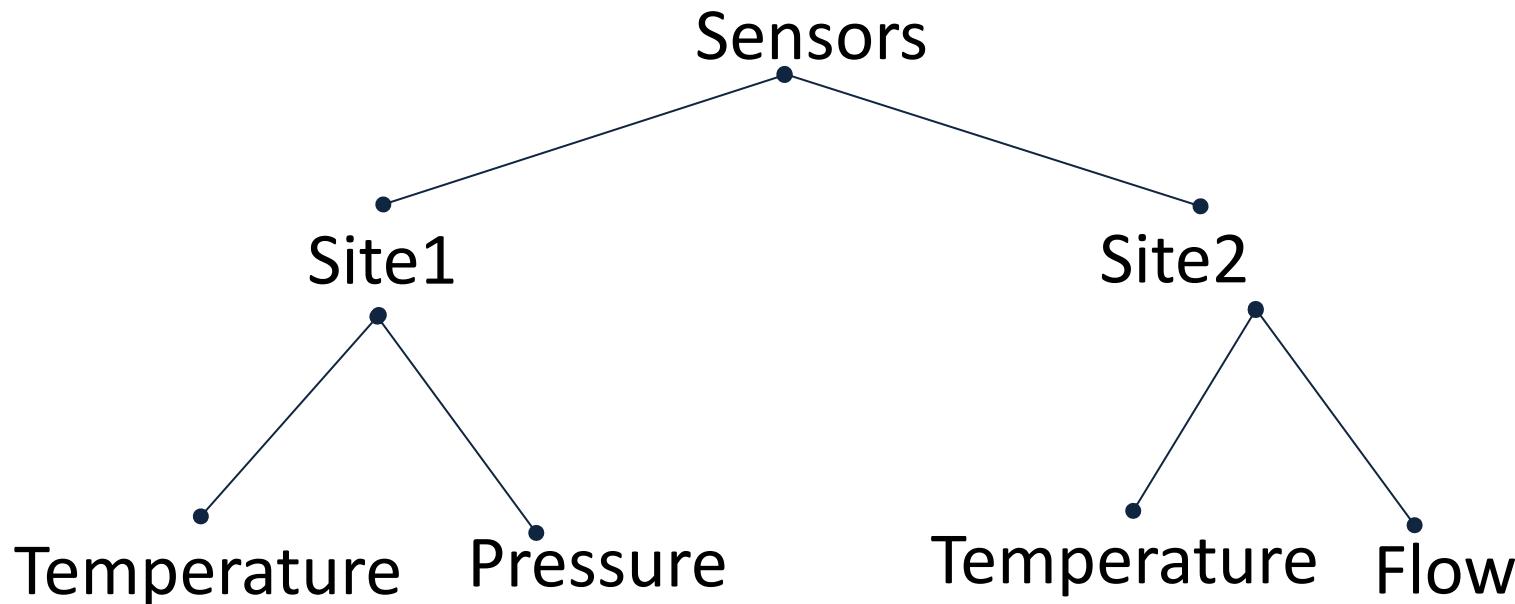
Example:

Sensor/Temperature/TMP36

- Topics are used to organize the data
- Topics are case sensitive
- Topics don't have to be pre-registered at the broker

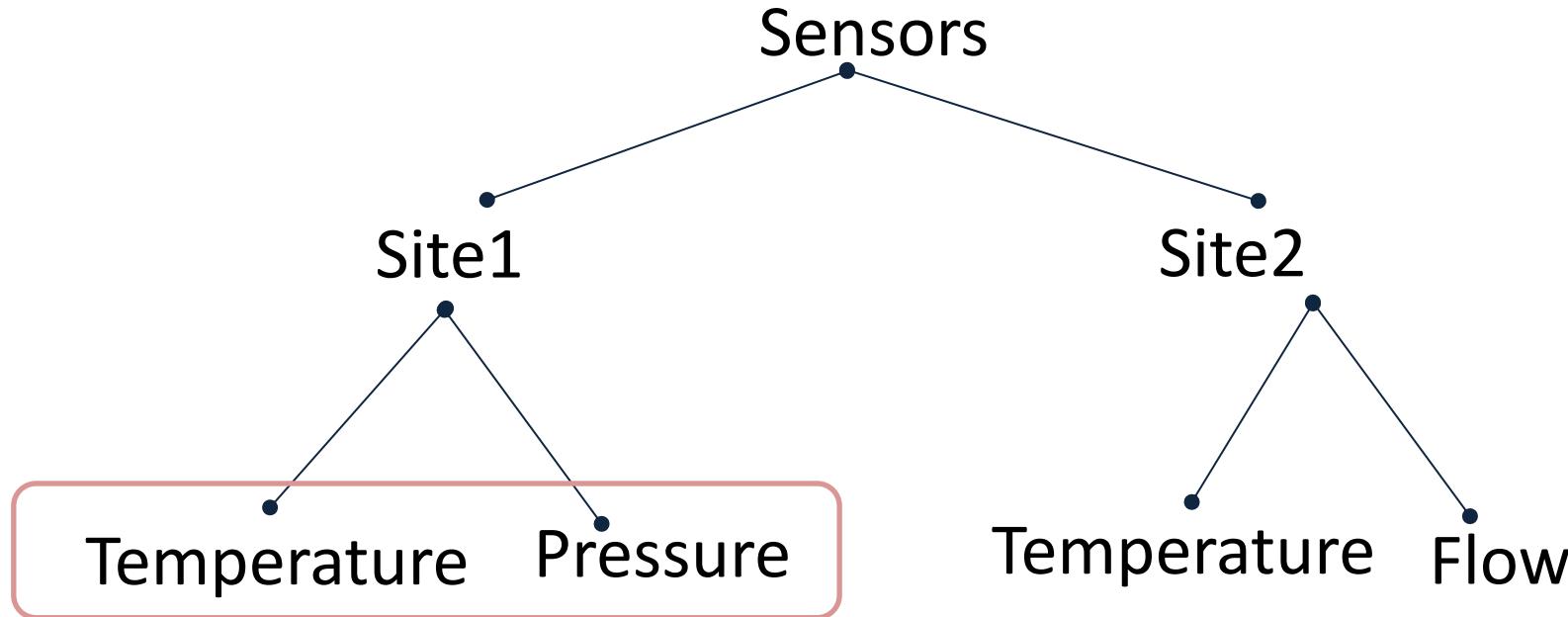
MQTT Topics

Topics are used to organize the data



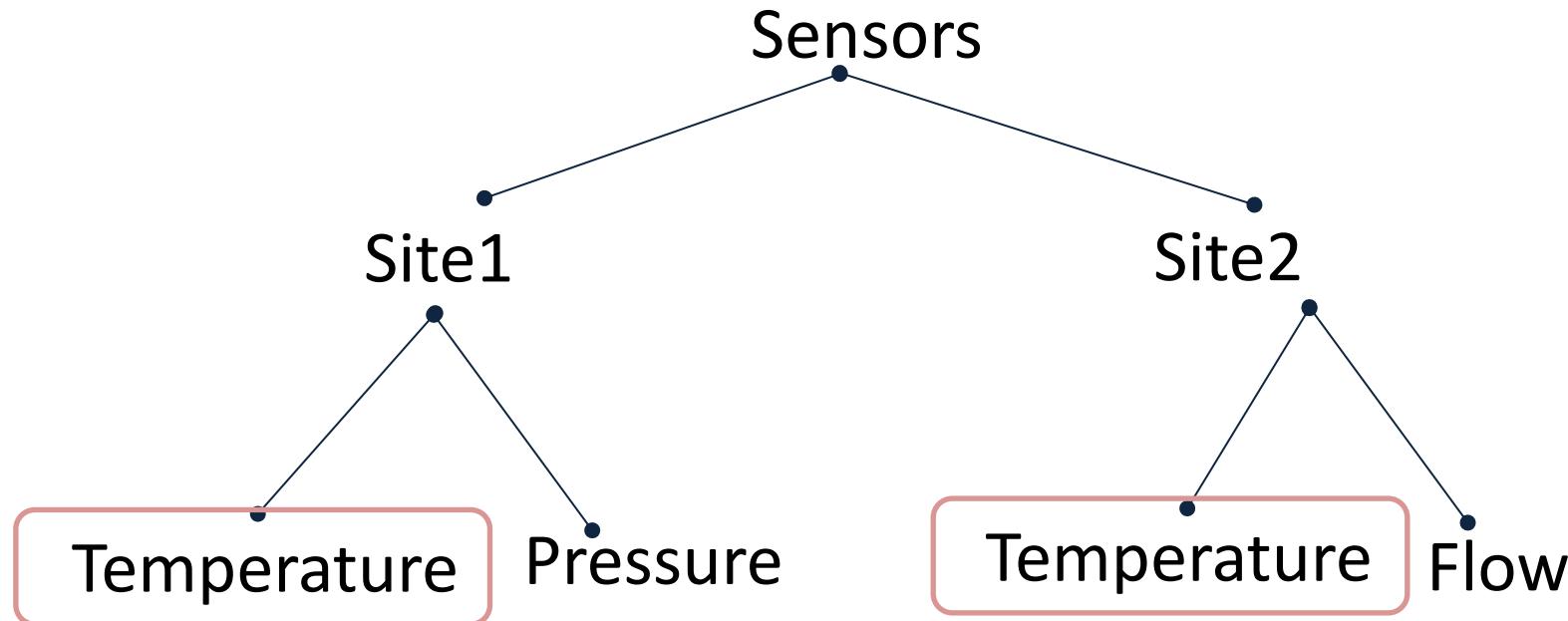
Subscribe on Topics - Wildcards

Wildcards: Sensors/Site1/#



Subscribe on Topics - Wildcards

Wildcards: Sensors/+Temperature



Quality of Service (QoS)

MQTT offers 3 Quality of Service levels:

- QoS **0** - Delivery at most once (“fire and forget”)
 - In QoS 0 there is no guarantee of delivery
- QoS **1** - Delivery at least once
 - QoS 1 guarantees that a message is delivered at least one time to the receiver
- QoS **2** - Delivery exactly once
 - QoS 2 is the highest level of service in MQTT. This level guarantees that each message is received only once by the intended recipients

<https://www.halvorsen.blog>



MQTT Brokers

Hans-Petter Halvorsen

[Table of Contents](#)

Free MQTT Brokers

- Eclipse Mosquitto
<https://mosquitto.org>
- HiveMQ Community Edition (HiveMQ CE)
<https://www.hivemq.com>
- **HiveMQ Cloud**
<https://www.hivemq.com>
- EMQ X MQTT IoT Cloud
<https://www.emqx.com/en/mqtt/public-mqtt5-broker>
- **ThingSpeak** (IoT Cloud Platform that offers an MQTT Broker among others)
<https://thingspeak.com>

<https://www.halvorsen.blog>



HiveMQ Cloud

MQTT Broker in the Cloud

Hans-Petter Halvorsen

[Table of Contents](#)

HiveMQ Cloud

<https://www.hivemq.com>

The screenshot shows the 'Cluster Details' page for a cluster managed by HiveMQ Cloud. The top navigation bar includes the HiveMQ logo and a 'Back to clusters' link. Below the navigation is a tab bar with 'Overview' (which is active), 'Access Management', and 'Getting started'. The main content area is divided into several sections: 'Details' (Hostname: [REDACTED].hivemq.cloud, Port (TLS): 8883, Port (WebSocket + TLS): 8884), 'Cluster Information' (Cluster Type: Free, Cloud Provider: Microsoft Azure), and 'Capacity' (MQTT Client Sessions: 0 / 100, Data Traffic: 0 B / 10 GB, Data Retention Time: 3 Days, Max Message Size: 5 MB). At the bottom left is a red 'DELETE CLUSTER' button, and at the bottom right is a yellow 'UPGRADE CLUSTER' button.

Cluster Details [Back to clusters](#)

Overview Access Management Getting started

Details

Hostname: [REDACTED].hivemq.cloud

Port (TLS): 8883

Port (WebSocket + TLS): 8884

Cluster Information

Cluster Type: Free

Cloud Provider: Microsoft Azure

Capacity

MQTT Client Sessions: 0 / 100

Data Traffic: 0 B / 10 GB

Data Retention Time: 3 Days

Max Message Size: 5 MB

DELETE CLUSTER

UPGRADE CLUSTER

HiveMQ Cloud

<https://www.hivemq.com>

The screenshot shows the HiveMQ Cloud interface for a cluster named "HIVEMQ CLOUD". The "Getting started" tab is active. The page is divided into sections:

- 1. Setup credentials for your IoT Devices**: A step completed with a checkmark.
- 2. Connect your first MQTT clients.**: A step in progress, indicated by a progress bar.
- Tools**: Includes icons for:
 - MQTT CLI**: command-line tool
 - MQTT.fx**: GUI tool
 - mosquitto_pub/sub**: command-line tool
 - HiveMQ Websocket Client**: browser tool
- Programming Languages**: Includes icons for:
 - Java**: hivemq-mqtt-client
 - Python**: Paho Python
 - JavaScript**: mqtt.js
 - Java (Websocket)**: hivemq-mqtt-client
 - C**: Paho C

Here you can find a basic Python example

<https://www.halvorsen.blog>



MQTT Clients

Hans-Petter Halvorsen

[Table of Contents](#)

Free MQTT Clients

- **MQTT X** is an MQTT 5.0 Open-source Desktop Client

<https://mqtx.app>

- **HiveMQ Community Edition (HiveMQ CE)**
 - Both Broker and MQTT Client

<https://www.hivemq.com>

<https://www.halvorsen.blog>



MQTT X

Open-source MQTT Desktop Client

Hans-Petter Halvorsen

[Table of Contents](#)

MQTT X

- MQTT X is an MQTT 5.0 Open-source MQTT Desktop Client
- It work with and Windows, macOS and Linux
- <https://mqtx.app>

MQTTX

MQTTX

File Edit View Window Help

Connections New Collection

No Data

+ New Collection

+ New Connection

Need a Cloud-Native fully-managed MQTT broker? Try [EMQ X Cloud](#) now!

Connect to Broker HiveMQ Cloud using MQTTX Client

The screenshot shows the MQTTX Client application interface. On the left is a dark sidebar with various icons: a green circle with an 'X', a square with a plus sign, a double arrow, a plus sign, a double slash, a clipboard, an info icon, and a gear icon. The main window has a title bar with the application name 'MQTTX' and standard window controls. The menu bar includes File, Edit, View, Window, and Help. Below the menu is a 'Connections' section with a 'New Collection' button. A list of connections shows 'ThingSpeakMQTT@...' and 'HiveMQCloud@aa4fd...', with 'HiveMQCloud@aa4fd...' selected and highlighted with a green triangle icon. The main content area is titled 'Edit' and shows the configuration for the selected connection. It includes fields for Name (redacted), Client ID (redacted).hivemq.cloud, Host (mqtts:// redacted.hivemq.cloud), Port (8883), Username (redacted), Password (redacted), SSL/TLS (true selected), Certificate (CA signed server selected), and SSL Secure (toggle switch on). Below these are 'Advanced' and 'Last Will and Testament' sections.

MQTTX

File Edit View Window Help

Connections New Collection

ThingSpeakMQTT@...

HiveMQCloud@aa4fd...

General

* Name: [redacted]

* Client ID: [redacted].hivemq.cloud

* Host: mqtts:// [redacted].hivemq.cloud

* Port: 8883

Username: [redacted]

Password: [redacted]

SSL/TLS: true false

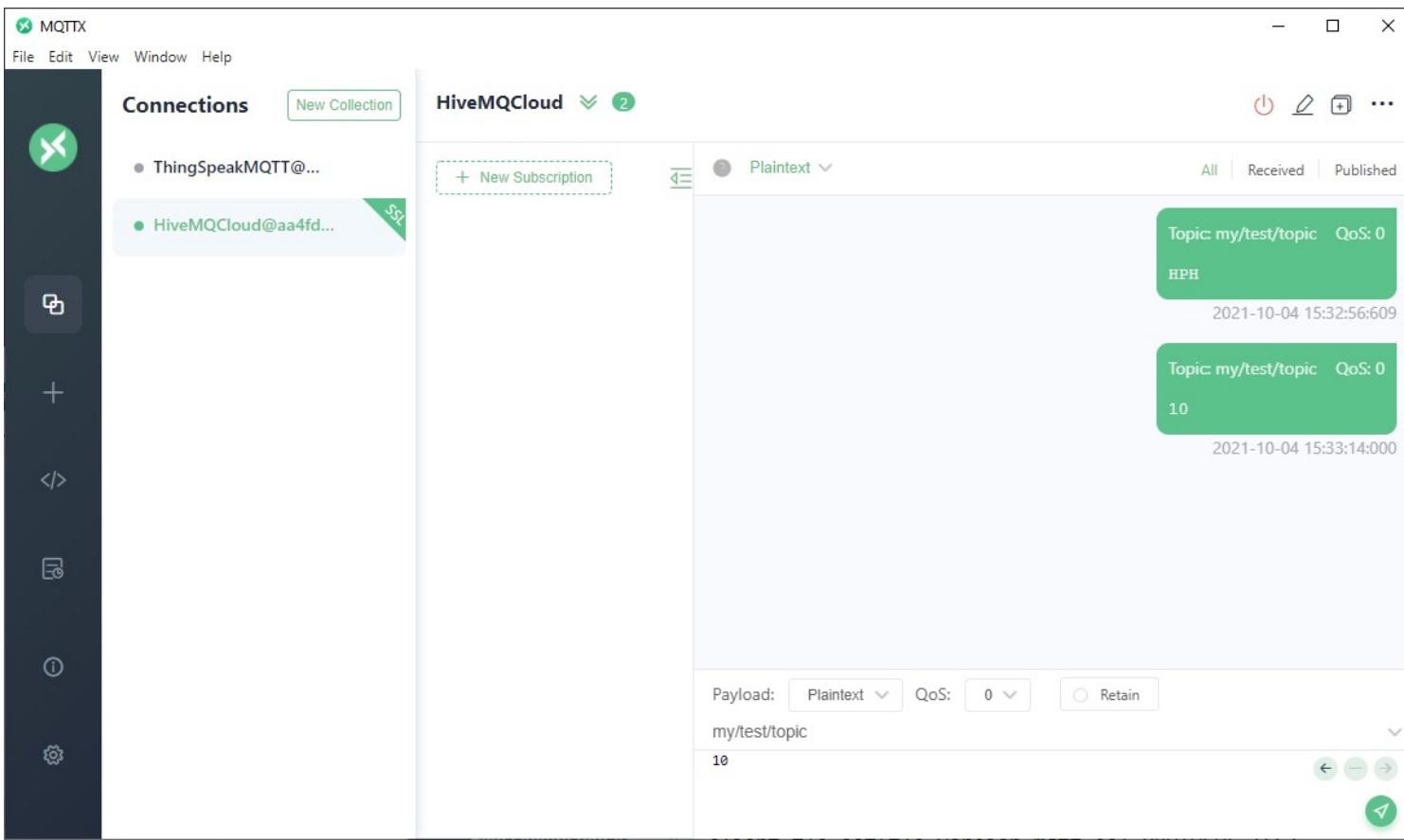
* Certificate: CA signed server Self signed

SSL Secure:

Advanced ▾

Last Will and Testament ▾

Publish to Broker HiveMQ Cloud using MQTTX Client



<https://www.halvorsen.blog>



Python

Using MQTT with Python

Hans-Petter Halvorsen

[Table of Contents](#)

Using MQTT in Python

- The most used MQTT Python Library is paho-mqtt
- We need to install the paho-mqtt Python Library using pip

paho-mqtt

Manage packages for C:\Users\hansha\AppData\Local\Programs\Thonny\python.exe

paho.mqtt

<INSTALL>

- astroid
- asttokens
- bcrypt
- bitstring
- cffi
- colorama
- colorzero
- cryptography
- docutils
- ecdsa
- esptool
- gpiod
- isort
- jedi

Search results

[paho-mqtt](#)
MQTT version 5.0/3.1.1 client class

[decorated-paho-mqtt](#)
Wrapper for Paho MQTT with declarative

[iottalk-paho-mqtt](#)
MQTT version 5.0/3.1.1 client class

[trio-paho-mqtt](#)
trio async MQTT Client

Manage packages for C:\Users\hansha\AppData\Local\Programs\Thonny\python.exe

paho.mqtt

<INSTALL>

- astroid
- asttokens
- bcrypt
- bitstring
- cffi
- colorama
- colorzero
- cryptography
- docutils
- ecdsa

paho-mqtt

Latest stable version: 1.5.1
Summary: MQTT version 5.0/3.1.1 client class
Author: Roger Light
Homepage: <http://eclipse.org/paho>
PyPI page: <https://pypi.org/project/paho-mqtt/>

We need to install the paho-mqtt Python Library. You can use pip, or as here, the Thonny Python Editor has an easy way to install Python Libraries from a GUI

mypy-extensions
paramiko
parso
ninel

Install ... Close

<https://www.halvorsen.blog>



HiveMQ Cloud and Python

Hans-Petter Halvorsen

[Table of Contents](#)

HiveMO Cloud Python Example

```
import paho.mqtt.client as mqtt

brokerAddress = "xxxxxx"
userName = "xxxxxx"
passWord = "xxxxxx"
topic = "my/test/topic"
data = "Hello"

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

def on_message(client, userdata, msg):
    print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)

client.subscribe(topic)
client.publish(topic, data)

client.loop_forever()
```

Example

We Publish some Data using MQTTX

The screenshot shows the MQTTX application interface. On the left is a sidebar with icons for file operations (New Collection, Connections, Subscriptions, Topics, Devices, Retained Messages, Help). The main area is titled "HiveMQCloud" and shows a list of published messages. A red box highlights the topic "Topic: Sensor/Temperature/TMP36". Below this, three red boxes show the data values "Data: 21", "Data: 22", and "Data: 23". A large red circle highlights the message list area.

Connections

- ThingSpeakMQTT@...
- HiveMQCloud@aa4fd...

HiveMQCloud

+ New Subscription

Plaintext

Topic: Sensor/Temperature/TMP36	QoS: 0	Date
21		2021-10-04 15:51:25:104
22		2021-10-04 15:57:30:859
23		2021-10-04 15:57:44:506

Payload: Plaintext QoS: 0 Retain

Sensor/Temperature/TMP36

23

Python Example

The screenshot shows the Thonny Python IDE interface. The main window displays the code for `mqtt_hivemq_cloud_ex2.py`. The code connects to a MQTT broker at `brokerAddress = "https://mosquitto.hivemq.cloud"`, uses `userName = "MQTT2"` and `passWord = "MQTT2@12345"`, subscribes to the topic `"Sensor/Temperature/TMP36"`, and publishes data `20`. It includes callbacks for connection and message reception. The shell window at the bottom shows the execution of the script and its output.

```
import paho.mqtt.client as mqtt
brokerAddress = "https://mosquitto.hivemq.cloud"
userName = "MQTT2"
passWord = "MQTT2@12345"
topic = "Sensor/Temperature/TMP36"
data = 20
# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))
# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))
# create the client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)
client.subscribe(topic)
client.publish(topic, data)
client.loop_forever()
```

In this Example the Thonny Python Editor has been used

We Subscribe to the Topic using Python – And as you see we get the same Data

```
Python 3.7.9 (bundled)
>>> %Run mqtt_hivemq_cloud_ex2.py
Connected successfully
Received message: Sensor/Temperature/TMP36 -> 20
Received message: Sensor/Temperature/TMP36 -> 21
Received message: Sensor/Temperature/TMP36 -> 22
Received message: Sensor/Temperature/TMP36 -> 23
```

Publish – Subscribe Examples

Thonny - /Users/halvorsen/OneDrive/Documents/Industrial IT and Automation/MQTT/MQTT Python/Publish...

Publish Temperature to HiveMQ Cloud.py

```
1 import paho.mqtt.client as mqtt
2 import random
3 import time
4
5 brokerAddress = "https://test.mosquitto.org:8883/tls/hivemq.cloud"
6 userName = "hmq72"
7 passWord = "hmq72"
8
9 topic = "Sensor/Temperature/TMP36"
10
11 min = 20
12 max = 30
13
14 # The callback for when the client receives a CONNACK response from the server.
15 def on_connect(client, userdata, flags, rc):
16     if rc == 0:
17         print("Connected successfully")
18     else:
19         print("Connect returned result code: " + str(rc))
20
21 # The callback for when a PUBLISH message is received from the server.
22 def on_message(client, userdata, msg):
23     print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))
24
25 # create the client
26 client = mqtt.Client()
27 client.on_connect = on_connect
28 client.on_message = on_message
29
30 client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
31 client.username_pw_set(userName, password)
32 client.connect(brokerAddress, 8883)
33
34 # Publish Temperature Data
35 wait = 20
36 while True:
37     data = random.randint(min, max)
38     print(data)
39     client.publish(topic, data)
40     time.sleep(wait)
41
42
```

Shell

```
24
25
26
27
28
29
```

Python 3.7.9 (bundled)

Python 3.7.9

Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\MQTT\MQTT Python\Subscribe on Topic in HiveMQ Cloud.py @ 2...

Subscribe on Topic in HiveMQ Cloud.py

```
1 import paho.mqtt.client as mqtt
2
3 brokerAddress = "https://test.mosquitto.org:8883/tls/hivemq.cloud"
4 userName = "hmq72"
5 passWord = "hmq72"
6
7 topic = "Sensor/Temperature/TMP36"
8
9 # The callback for when the client receives a CONNACK response from the server.
10 def on_connect(client, userdata, flags, rc):
11     if rc == 0:
12         print("Connected successfully")
13     else:
14         print("Connect returned result code: " + str(rc))
15
16 # The callback for when a PUBLISH message is received from the server.
17 def on_message(client, userdata, msg):
18     print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))
19
20 # create the client
21 client = mqtt.Client()
22 client.on_connect = on_connect
23 client.on_message = on_message
24
25 client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
26 client.username_pw_set(userName, password)
27 client.connect(brokerAddress, 8883)
28
29 client.subscribe(topic)
30
31 client.loop_forever()
```

Shell

```
Received message: Sensor/Temperature/TMP36 -> 25
Received message: Sensor/Temperature/TMP36 -> 25
Received message: Sensor/Temperature/TMP36 -> 20
Received message: Sensor/Temperature/TMP36 -> 24
Received message: Sensor/Temperature/TMP36 -> 25
Received message: Sensor/Temperature/TMP36 -> 20
Received message: Sensor/Temperature/TMP36 -> 29
```

Python 3.7.9

Publish

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

brokerAddress = "xxxxxx"
userName = "xxxxxx"
passWord = "xxxxxx"

topic = "Sensor/Temperature/TMP36"

min = 20
max = 30

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

# create the client
client = mqtt.Client()
client.on_connect = on_connect

client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)

# Publish Temperature Data
wait = 20
while True:
    data = random.randint(min, max)
    print(data)
    client.publish(topic, data)
    time.sleep(wait)
```

Subscribe

```
import paho.mqtt.client as mqtt

brokerAddress = "xxxxxx"
userName = "xxxxxx"
passWord = "xxxxxxxx"

topic = "Sensor/Temperature/TMP36"

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

def on_message(client, userdata, msg):
    print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))

# create the client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)

client.subscribe(topic)

client.loop_forever()
```

Summary

- Example 1
 - Python Publish Data to a Topic
 - MQTT X Client Subscribing on the same Topic
- Example 2
 - MQTT X Client Publish Data to a Topic
 - Python Subscribing on the same Topic
- Example 3
 - Python Publish Data to a Topic
 - Python Subscribing on the same Topic

<https://www.halvorsen.blog>



ThingSpeak

Internet of Things Cloud Service

Hans-Petter Halvorsen

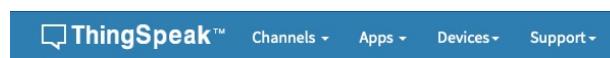
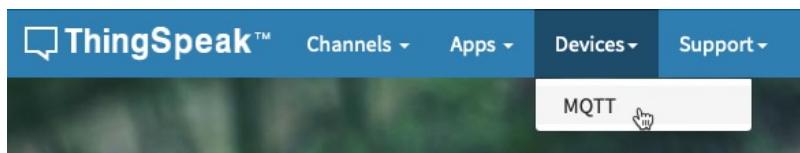
[Table of Contents](#)

MQTT ThingSpeak

- <https://mathworks.com/help/thingspeak/use-desktop-mqtt-client-to-publish-to-a-channel.html>

Configure MQTT in ThingSpeak

<https://thingspeak.com>



MQTT Devices

Add a new device

Device Information

Name*

Description

Authorize channels to access ⓘ

-- Select a Channel --

...

Authorized Channel ⓘ	Allow Publish	Allow Subscribe
No channels authorized.		

ThingSpeak and MQTTX

The screenshot shows the MQTTX application window. On the left is a sidebar with icons for New Collection, Back, Connect, No Data, and other settings. The main area has tabs for Connections and New. The New tab is active, showing a 'General' configuration panel with fields for Name, Client ID, Host (set to mqtt:// and mqtt3.thingspeak.com), Port (set to 1883), Username, and Password. An SSL/TLS dropdown is set to false. Below this is an Advanced section and a Last Will and Testament section.

ThingSpeak MQTT Broker:
mqtt:// mqtt3.thingspeak.com

Publish to Channel Field

Payload:

Plaintext ▾

QoS:

0 ▾

Retain

channels/<ChannelID>/publish/fields/field1

45



Topic: channels/<ChannelID>/publish/fields/field1
Data: 45

Publish to Channel Field

The image shows a screenshot of the MQTTX application and the ThingSpeak website. On the left, the MQTTX interface displays a connection named "ThingSpeakMQTT" with a single subscription to the topic "channels/871951/publish/fields/field1". A message with payload "45" is shown, timestamped 2021-10-04 11:28:42:707. This message is highlighted with a red rounded rectangle. On the right, the ThingSpeak "Sensor System" channel page shows the published data. The channel ID is 871951, created 2 years ago, and has 1 entry. Two charts, "Field 1 Chart" and "Field 2 Chart", both titled "Sensor System", are displayed, each showing a single data point at the timestamp 11:27:56.000. The ThingSpeak logo and URL "ThingSpeak.com" are visible at the bottom of the charts.

Topic: channels/<ChannelID>/publish/fields/field1
Data: 45

Publish to Channel Field

ThingSpeak™

Channels Apps Devices Support Commercial Use How to Buy HH

Sensor System

Channel ID: 871951
Author: hansha
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

+ Add Visualizations + Add Widgets Export recent data

MATLAB Analysis MATLAB Visualization

Topic: channels/<ChannelID>/publish/fields/field1
Data: 45

Channel 2 of 3 < >

Channel Stats

Created: 2 years ago Entries: 1

Field 1 Chart

Sensor System

Field Label 1

45

Date 12:48:21.000

ThingSpeak.com

Field 2 Chart

Sensor System

Field Label 2

Date

ThingSpeak.com

Publish to Channel Feed

Here we will Publish to **multiple** Fields within a Channel

The screenshot shows the MQTDX application interface. On the left is a sidebar with various icons and a 'Connections' section containing a single entry: 'ThingSpeakMQTT@...'. The main window title is 'ThingSpeakMQTT' with a connection count of 2. Below the title are buttons for 'Plaintext' (selected), 'All', 'Received', and 'Published'. Two messages are listed:

- Topic: channels/871951/publish/fields/field1 QoS: 0
45
2021-10-04 11:28:42:707
- Topic: channels/871951/publish QoS: 0
field1=20&field2=30&status=MQTTPUBLISH

A red callout box highlights the second message with the following text:

Topic: channels/<ChannelID>/publish
Data: field1=20&field2=30&status=MQTTPUBLISH

At the bottom of the main window, there is a payload editor with the following fields:

- Payload: channels/871951/publish
- QoS: 0
- Retain:
- Data: field1=20&field2=30&status=MQTTPUBLISH

Publish to Channel Feed

ThingSpeak™ [Channels](#) [Apps](#) [Devices](#) [Support](#) [Commercial Use](#) [How to Buy](#) [HH](#)

Sensor System

Channel ID: [REDACTED]
Author: [hansha](#)
Access: Private

Private View [Public View](#) [Channel Settings](#) [Sharing](#) [API Keys](#) [Data Import / Export](#)

[+ Add Visualizations](#) [+ Add Widgets](#) [Export recent data](#) [MATLAB Analysis](#) [MATLAB Visualization](#)

Channel 2 of 3 < >

Channel Stats

Created: [2 years ago](#)
Entries: 2

Field 1 Chart

Sensor System

Date	Field Label 1
11:30	45
12:00	20

ThingSpeak.com

Field 2 Chart

Sensor System

Date	Field Label 2
12:13:46.000	30
11:30	45

ThingSpeak.com

Subscribe to a Channel Feed

The screenshot shows the MQTTX application interface. On the left is a sidebar with various icons: a green circle with an 'X', a square with a plus sign, a plus sign, a double slash, a square with a circular arrow, a question mark, and a gear. The main window has a title bar with 'MQTTX' and standard window controls. The menu bar includes 'File', 'Edit', 'View', 'Window', and 'Help'. Below the menu is a 'Connections' section with a 'ThingSpeakMQTT' collection containing one connection entry. A red box highlights the '+ New Subscription' button. The central area is titled 'New Subscription' and contains fields for 'Topic' (set to 'channels/!/subscribe'), 'QoS' (set to '0'), 'Color' (set to '#1FC6A0'), and 'Alias' (Payload: 'Plai', Topic: 'Topic'). At the bottom right are 'Cancel' and 'Confirm' buttons.

MQTTX

File Edit View Window Help

Connections

New Collection

ThingSpeakMQTT 2

+ New Subscription

New Subscription

* Topic

channels/!/subscribe

* QoS

0

#1FC6A0

Color

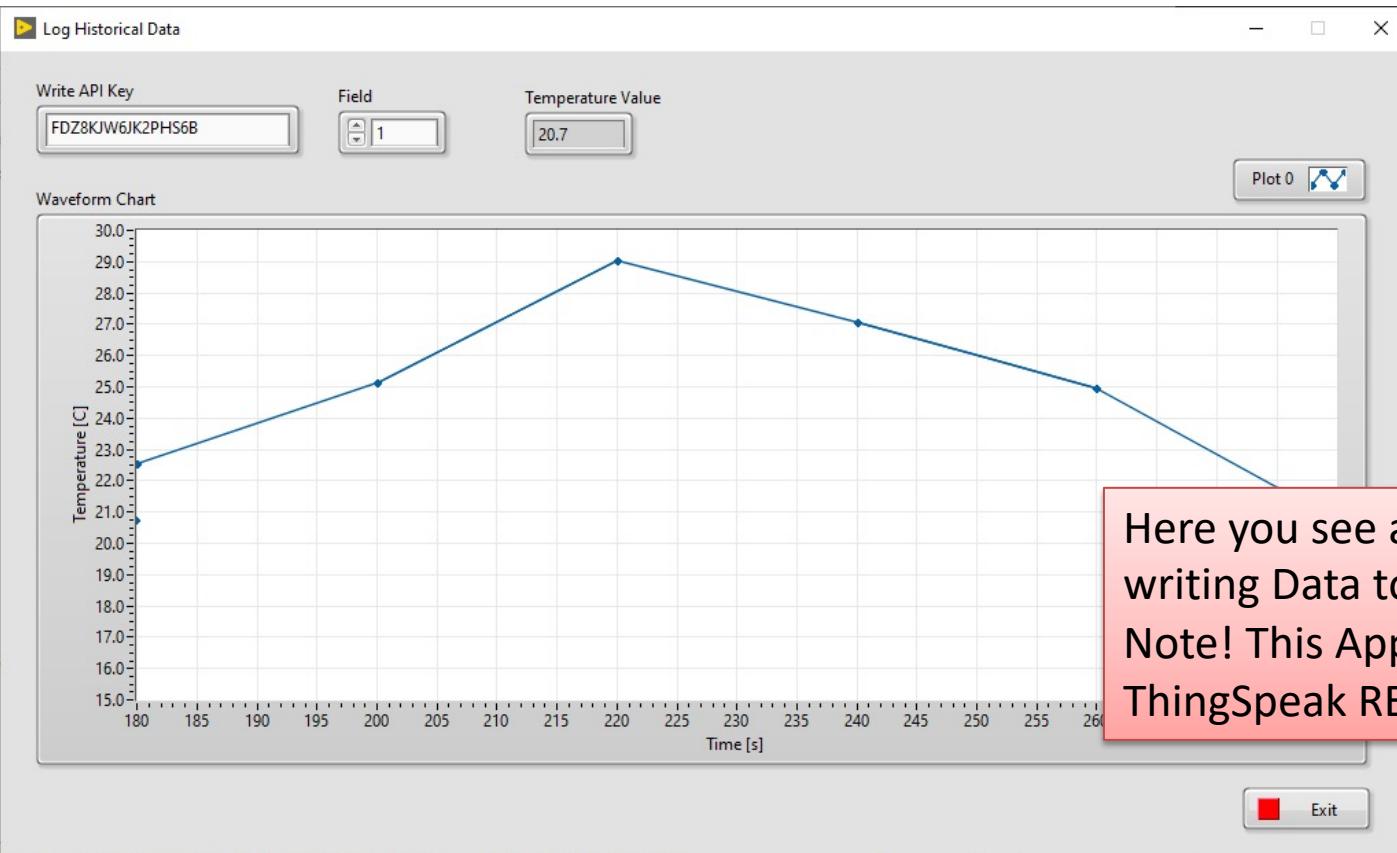
Alias

Payload: Plai

Topic

Cancel Confirm

Subscribe to a Channel Feed



Subscribe to a Channel Feed

The screenshot shows the MQTTX application interface. The top navigation bar includes 'MQTTX' (with a green checkmark icon), 'File', 'Edit', 'View', 'Window', and 'Help'. The title bar shows 'ThingSpeakMQTT' with a dropdown arrow and a green notification badge with the number '3'. The left sidebar has a dark theme with icons for 'Connections' (highlighted in green), 'New Collection', 'MQTT', 'TCP', 'HTTP', 'Websocket', 'Cloud', 'MQTT Broker', 'MQTT Client', and 'MQTT Publish'. The main area displays a 'Connections' list with one entry: 'ThingSpeakMQTT@...'. Below it is a 'New Subscription' button. A message list shows a published message from 'channels/871951...' with QoS 0. The message payload is: `field1=20&field2=30&status=MQTTPUBLISH`, timestamped 2021-10-04 12:14:33:077. Another message is shown with a topic of 'channels/871951/subscribe' and QoS 0, containing a JSON object with various field values. The timestamp for this message is 2021-10-04 12:37:58:642. At the bottom, there are buttons for 'Payload: Plaintext', 'QoS: 0', 'Retain', 'Topic', and navigation controls.

MQTTX

File Edit View Window Help

Connections New Collection

ThingSpeakMQTT 3

ThingSpeakMQTT@...

+ New Subscription

channels/871951... QoS 0

Topic: channels/871951/subscribe QoS: 0

{ "channel_id":871951,"created_at":"2021-10-04T10:37:12Z","entry_id":12,"field1":22.56,"field2":null,"field3":null,"field4":null,"field5":null,"field6":null,"field7":null,"field8":null,"latitude":null,"longitude":null,"elevation":null,"status":null}

2021-10-04 12:37:58:642

Payload: Plaintext QoS: 0 Retain Topic

<https://www.halvorsen.blog>



ThingSpeak and Python

Hans-Petter Halvorsen

[Table of Contents](#)

Thony - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\MQTT\MQTT ThingSpeak\Python Examples\Publish Temperature to...

File Edit View Run Tools Help

Publish Temperature to ThingSpeak.py

```
1 import paho.mqtt.client as mqtt
2 import random
3 import time
4
5 brokerAddress = "mqtt3.thingspeak.com"
6 port = 1883
7 clientId = "L_o-----o"
8 userName = "L_o-----o"
9 passWord = "-----"
10 channelID = "071991"
11 field = "field1"
12 topic = "channels/" + channelID + "/publish/fields/" + field
13
14 min = 20
15 max = 30
16
17 # The callback for when the client receives a CONNACK response from the server.
18 def on_connect(client, userdata, flags, rc):
19     if rc == 0:
20         print("Connected successfully")
21     else:
22         print("Connect returned result code: " + str(rc))
23
24 # create the client
25 client = mqtt.Client(clientId)
26 client.on_connect = on_connect
27
28 client.username_pw_set(userName, password)
29 client.connect(brokerAddress, port)
30
31 # Publish Temperature Data
32 wait = 30
```

Shell x

```
25
24
30
22
27
28
27
25
```

Publish

Publish

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

brokerAddress = "mqtt3.thingspeak.com"
port = 1883
clientId = "xxxxxx"
userName = "xxxxxx"
passWord = "xxxxxx"
channelID = "xxxxxx"
field = "field1"
topic = "channels/" + channelID + "/publish/fields/" + field

min = 20
max = 30

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

# create the client
client = mqtt.Client(clientId)
client.on_connect = on_connect

client.username_pw_set(userName, passWord)
client.connect(brokerAddress, port)

# Publish Temperature Data
wait = 20
while True:
    data = random.randint(min, max)
    print(data)
    client.publish(topic, data)
    time.sleep(wait)
```

Thonny - /Users/halvorsen/OneDrive/Documents/Industrial IT and Automation/MQTT/MQTT ThingSpeak/Python Examples/Subscribe on Topic in ThingSpeak.py @ 34 : 1

Subscribe

```
<untitled> Subscribe on Topic in ThingSpeak.py
```

```
1 import paho.mqtt.client as mqtt
2
3 brokerAddress = "mqtt3.thingspeak.com"
4 port = 1883
5 clientId = "Log_Levels_2019-07-15_10-30"
6 userName = "Log_Levels_2019-07-15_10-30"
7 passWord = "f4c1e72a9c14a7d5a3d"
8 channelID = "871951"
9 field = "field1"
10 topic = "channels/" + channelID + "/publish/fields/" + field
11
12 # The callback for when the client receives a CONNACK response from the server.
13 def on_connect(client, userdata, flags, rc):
14     if rc == 0:
15         print("Connected successfully")
16     else:
17         print("Connect returned result code: " + str(rc))
18
19 # The callback for when a PUBLISH message is received from the server.
20 def on_message(client, userdata, msg):
21     print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))
22
23 # create the client
24 client = mqtt.Client(clientId)
25 client.on_connect = on_connect
26 client.on_message = on_message
27
28 client.username_pw_set(userName, passWord)
29 client.connect(brokerAddress, port)
30
31 client.subscribe(topic)
32
```

Shell

```
Python 3.7.9 (bundled)
>>> %cd '/Users/halvorsen/OneDrive/Documents/Industrial IT and Automation/MQTT/MQTT ThingSpeak/Python Examples'
>>> %Run 'Subscribe on Topic in ThingSpeak.py'

Connected successfully
Received message: channels/871951/publish/fields/field1 -> 21
Received message: channels/871951/publish/fields/field1 -> 30
Received message: channels/871951/publish/fields/field1 -> 30
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 26
Received message: channels/871951/publish/fields/field1 -> 24
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 24
Received message: channels/871951/publish/fields/field1 -> 30
Received message: channels/871951/publish/fields/field1 -> 22
Received message: channels/871951/publish/fields/field1 -> 27
Received message: channels/871951/publish/fields/field1 -> 28
Received message: channels/871951/publish/fields/field1 -> 27
Received message: channels/871951/publish/fields/field1 -> 25
```

```
import paho.mqtt.client as mqtt

brokerAddress = "mqtt3.thingspeak.com"
port = 1883
clientId = "xxxxxx"
userName = "xxxxxx"
passWord = "xxxxxx"
channelID = "xxxxxx"
field = "field1"
topic = "channels/" + channelID + "/publish/fields/" + field

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

def on_message(client, userdata, msg):
    print("Received message: " + msg.topic + " -> " +
msg.payload.decode("utf-8"))

# create the client
client = mqtt.Client(clientId)
client.on_connect = on_connect
client.on_message = on_message

client.username_pw_set(userName, passWord)
client.connect(brokerAddress, port)

client.subscribe(topic)

client.loop_forever()
```

Subscribe

Thonny - C:\Users\shasha\OneDrive\Documents\Industrial IT and Automation\MQTT\ThingSpeak\Python Examples\Publish Temperature to... - □ X Thonny - /Users/halvorsen/OneDrive/Documents/Industrial IT and Automation/MQTT/ThingSpeak/Python Examples/Subscribe on Topic in ThingSpeak.py @ 34:1

Publish Temperature to ThingSpeak.py x

Publish

```
1 import paho.mqtt.client as mqtt
2 import random
3 import time
4
5 brokerAddress = "mqtt3.thingspeak.com"
6 port = 1883
7 clientId = "L_-----"
8 userName = "L-----"
9 passWord = "-----"
10 channelID = "071951"
11 field = "field1"
12 topic = "channels/" + channelID + "/publish/fields/" + field
13
14 min = 20
15 max = 30
16
17 # The callback for when the client receives a CONNACK response from the server.
18 def on_connect(client, userdata, flags, rc):
19     if rc == 0:
20         print("Connected successfully")
21     else:
22         print("Connect returned result code: " + str(rc))
23
24 # create the client
25 client = mqtt.Client(clientId)
26 client.on_connect = on_connect
27
28 client.username_pw_set(userName, passWord)
29 client.connect(brokerAddress, port)
30
31 # Publish Temperature Data
32 wait = 20
```

Shell x

```
25
24
30
22
27
28
29
30
31
32
```

Subscribe

```
<untitled> Subscribe on Topic in ThingSpeak.py x
1 import paho.mqtt.client as mqtt
2
3 brokerAddress = "mqtt3.thingspeak.com"
4 port = 1883
5 clientId = "L_-----"
6 userName = "L-----"
7 passWord = "-----"
8 channelID = "-----"
9 field = "field1"
10 topic = "channels/" + channelID + "/publish/fields/" + field
11
12 # The callback for when the client receives a CONNACK response from the server.
13 def on_connect(client, userdata, flags, rc):
14     if rc == 0:
15         print("Connected successfully")
16     else:
17         print("Connect returned result code: " + str(rc))
18
19 # The callback for when a PUBLISH message is received from the server.
20 def on_message(client, userdata, msg):
21     print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))
22
23 # create the client
24 client = mqtt.Client(clientId)
25 client.on_connect = on_connect
26 client.on_message = on_message
27
28 client.username_pw_set(userName, passWord)
29 client.connect(brokerAddress, port)
30
31 client.subscribe(topic)
32
```

Shell x

```
Python 3.7.9 (bundled)
>>> %cd '/Users/halvorsen/OneDrive/Documents/Industrial IT and Automation/MQTT/ThingSpeak/Python Examples'
>>> !Run 'Subscribe on Topic in ThingSpeak.py'
Connected successfully
Received message: channels/871951/publish/fields/field1 -> 21
Received message: channels/871951/publish/fields/field1 -> 30
Received message: channels/871951/publish/fields/field1 -> 30
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 26
Received message: channels/871951/publish/fields/field1 -> 24
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 25
Received message: channels/871951/publish/fields/field1 -> 24
Received message: channels/871951/publish/fields/field1 -> 30
Received message: channels/871951/publish/fields/field1 -> 22
Received message: channels/871951/publish/fields/field1 -> 27
Received message: channels/871951/publish/fields/field1 -> 28
Received message: channels/871951/publish/fields/field1 -> 27
Received message: channels/871951/publish/fields/field1 -> 25
```

Python 3.7.9

Summary

- A short introduction to **MQTT** has been given
- Introduction to some **MQTT Brokers**
- Use of **MQTT Desktop Client** software
 - **MQTT X**
- **Python Examples**
 - **HiveMQ Cloud**
 - **ThingSpeak**

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>

