

<https://www.halvorsen.blog>



MQTT

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



MQTT

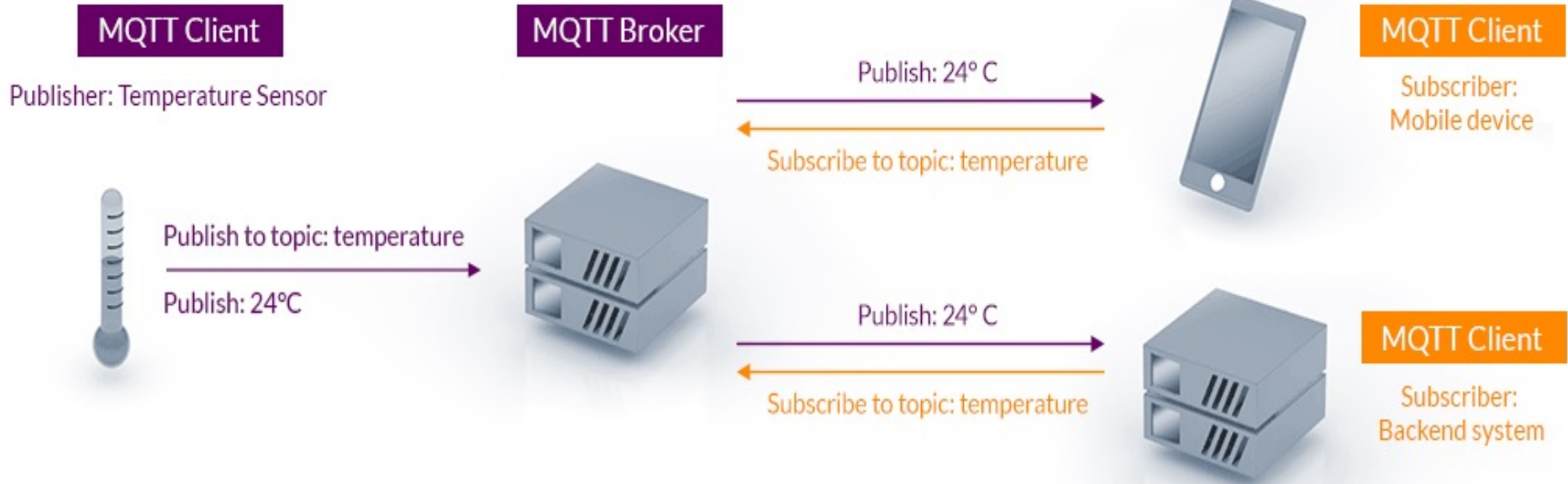
Hans-Petter Halvorsen

[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



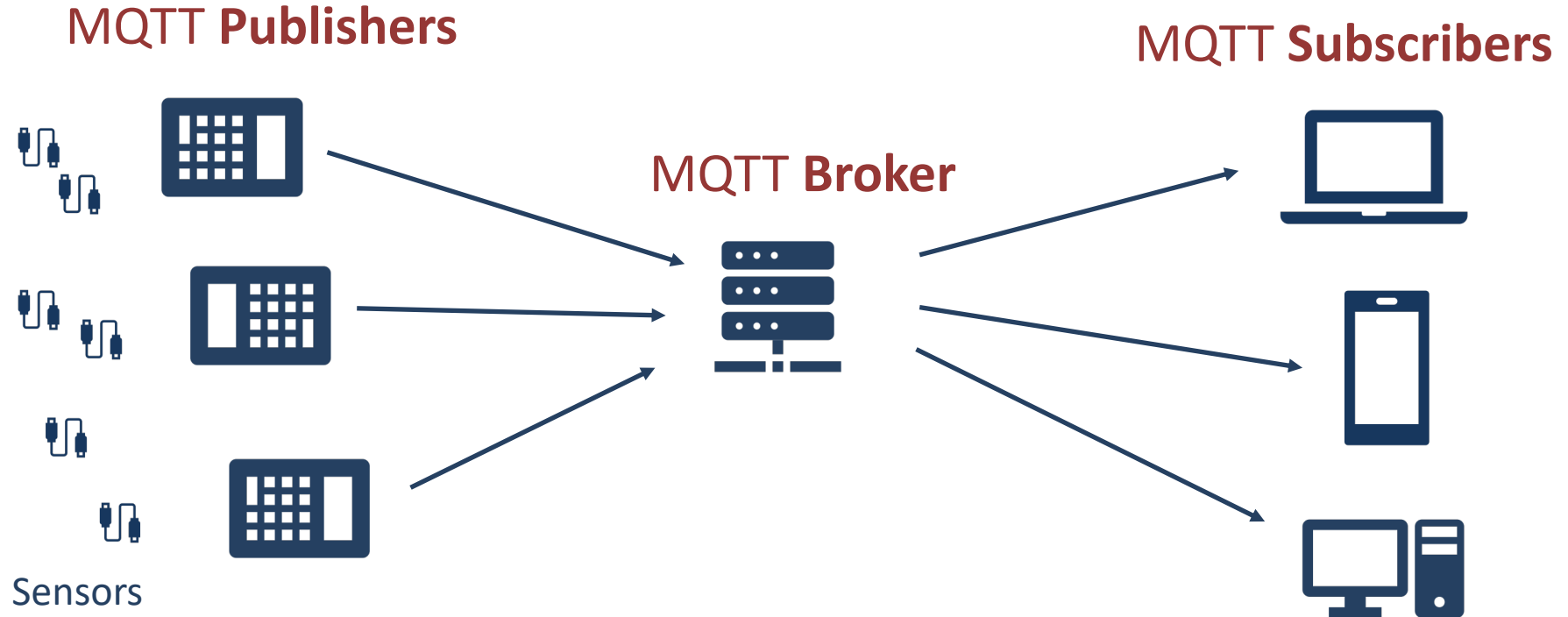
MQTT

- 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

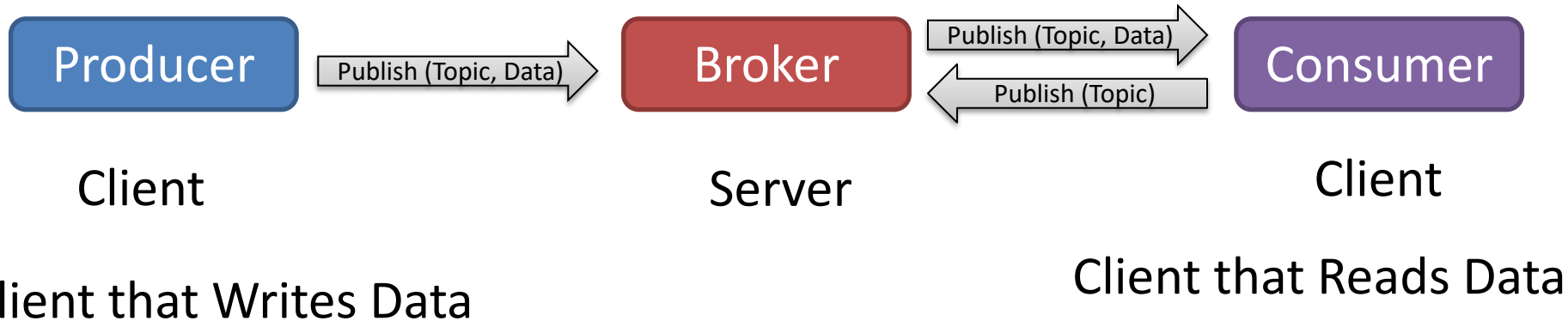
- Internet of Things (IoT): To get data to and from devices on a network.
- MQTT is a lightweight protocol that makes this easier

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

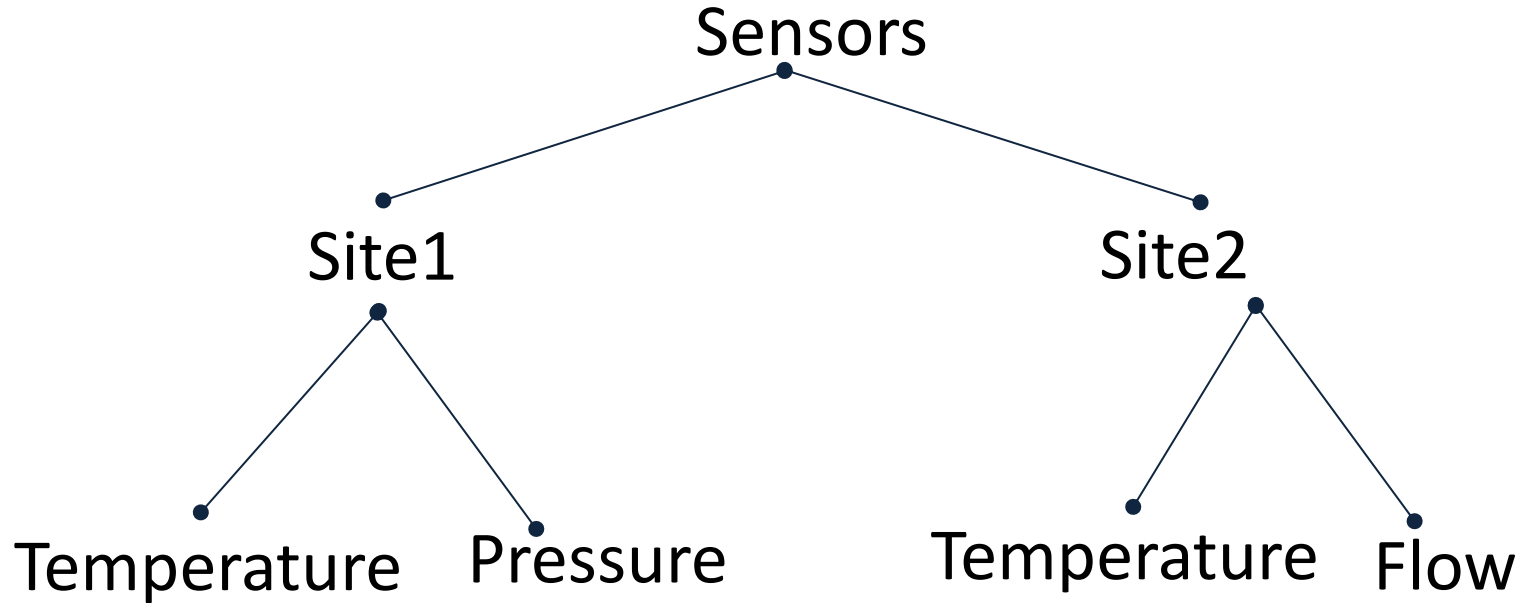
Example:

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

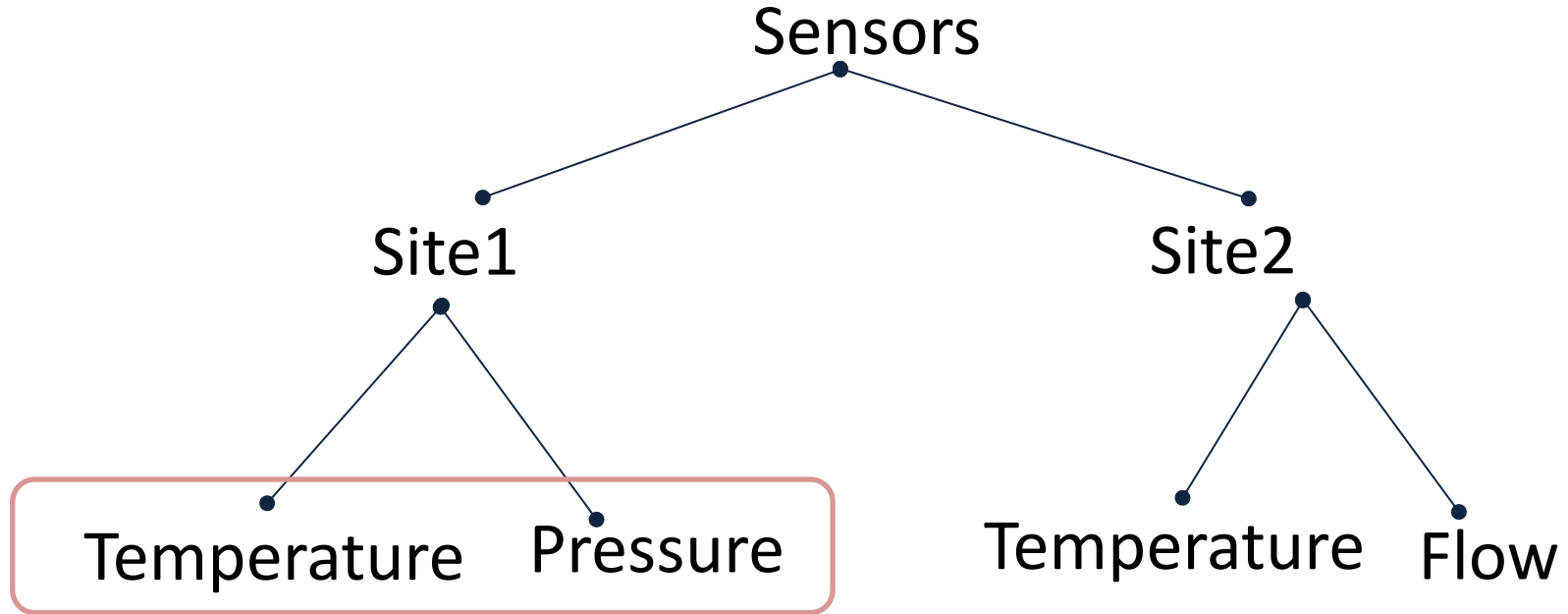
MQTT Topics

Topics are used to organize the data



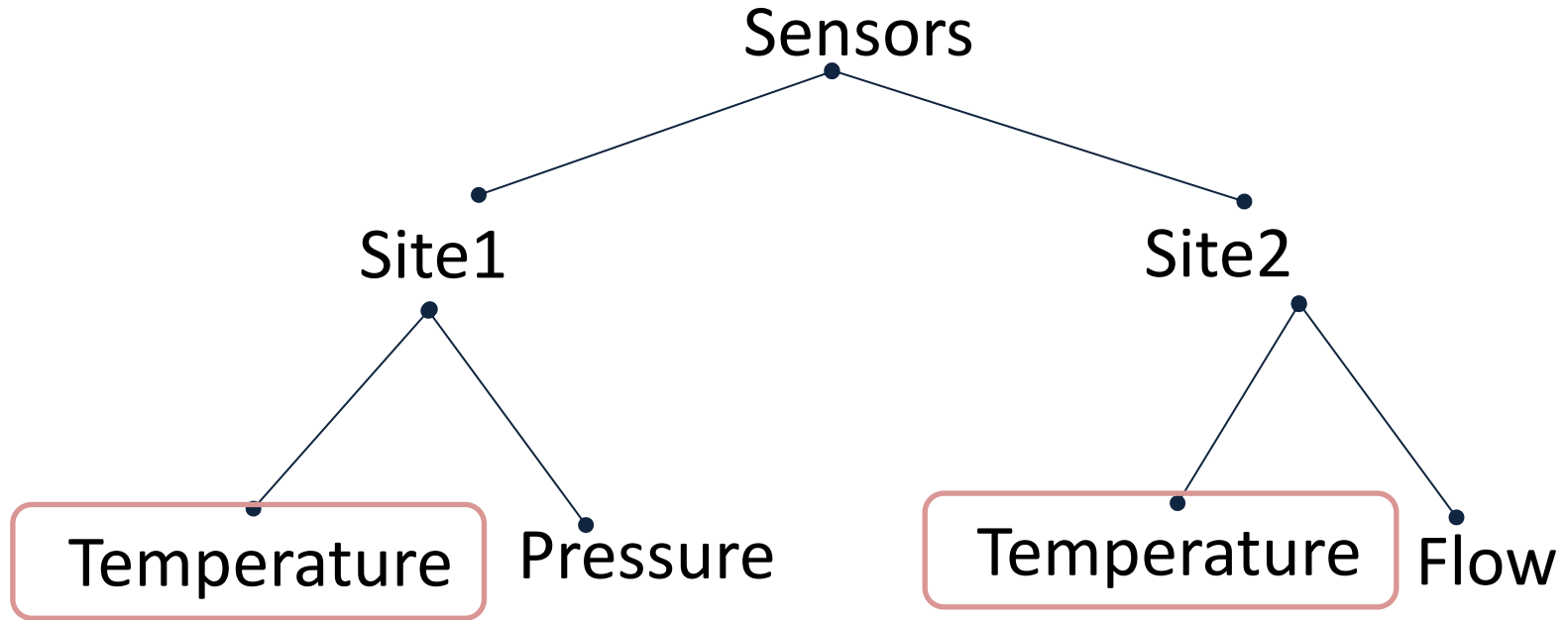
Subscribe on Topics - Wildcards

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



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>



HiveMQ Cloud


MQTT Broker in the Cloud

Hans-Petter Halvorsen

[Table of Contents](#)

HiveMQ Cloud

<https://www.hivemq.com>

 **HIVEMQ**
CLOUD

Cluster Details[Back to clusters](#)

Overview

Access Management

Getting started

Details
Hostname: .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
[UPGRADE CLUSTER](#)

[DELETE CLUSTER](#)

HiveMQ Cloud

<https://www.hivemq.com>

Here you can find a
basic Python example

HIVEMQ
CLOUD

Hans-Petter Halvorsen ▾

Cluster Details[Back to clusters](#)

Overview


Access Management


Getting started


1. Setup credentials for your IoT Devices ✓


2. Connect your first MQTT clients.
Choose your preferred tool or programming language.

Tools


**mqtt-cli**
command-line tool


**MQTT.fx**
GUI tool


**mosquitto_pub/sub**
command-line tool


**HiveMQ Websocket Client**
browser tool


Programming Languages

**Java**
hivemq-mqtt-client

**Python**
Paho Python

**JavaScript**
mqtt.js

**Java (Websocket)**
hivemq-mqtt-client

**C**
Paho C



MQTT Clients

Hans-Petter Halvorsen

[Table of Contents](#)

Free MQTT Clients

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

<https://mqttx.app>

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

<https://www.hivemq.com>



MQTT X

Open-source MQTT Desktop Client

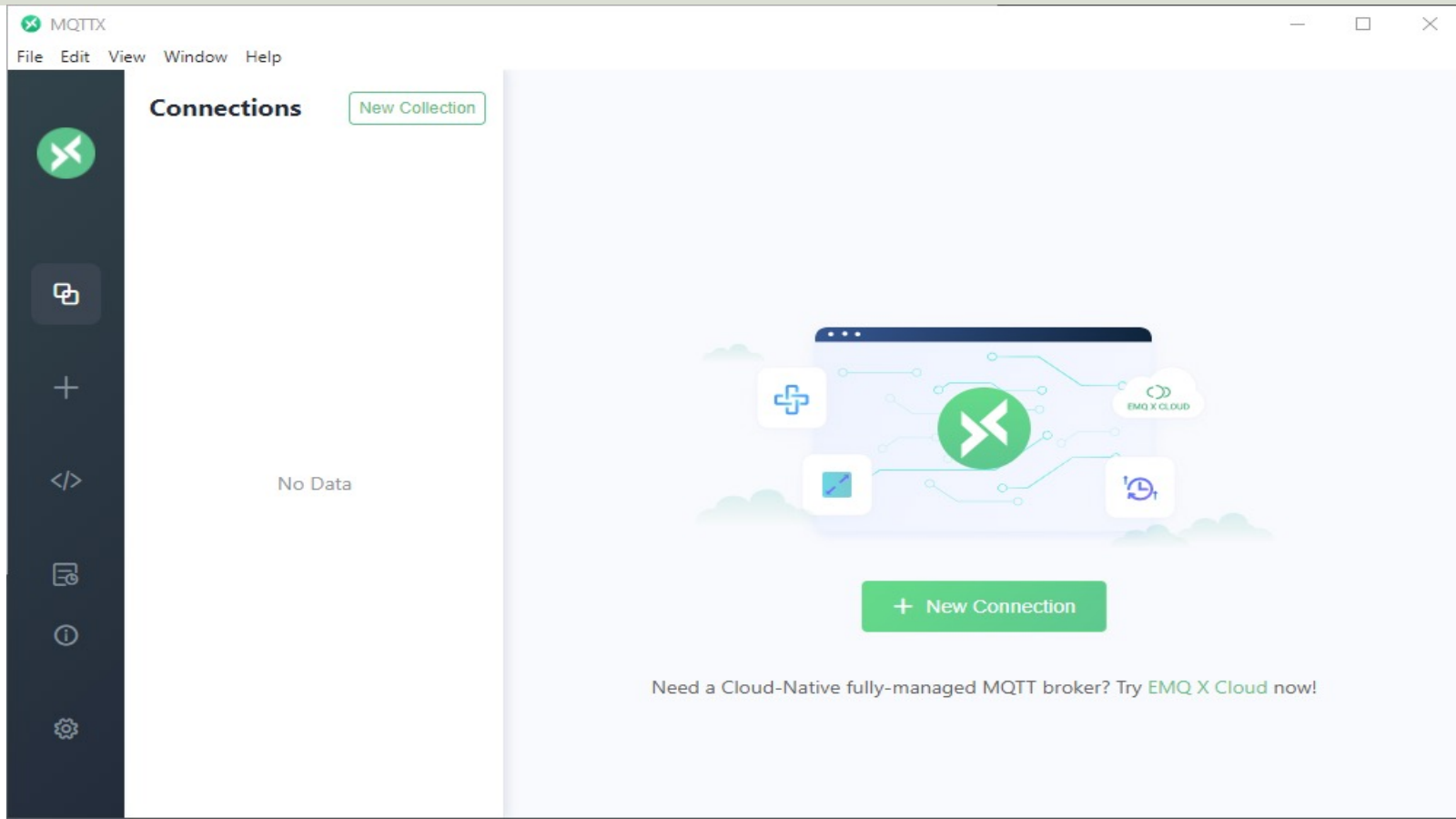
Hans-Petter Halvorsen

[Table of Contents](#)

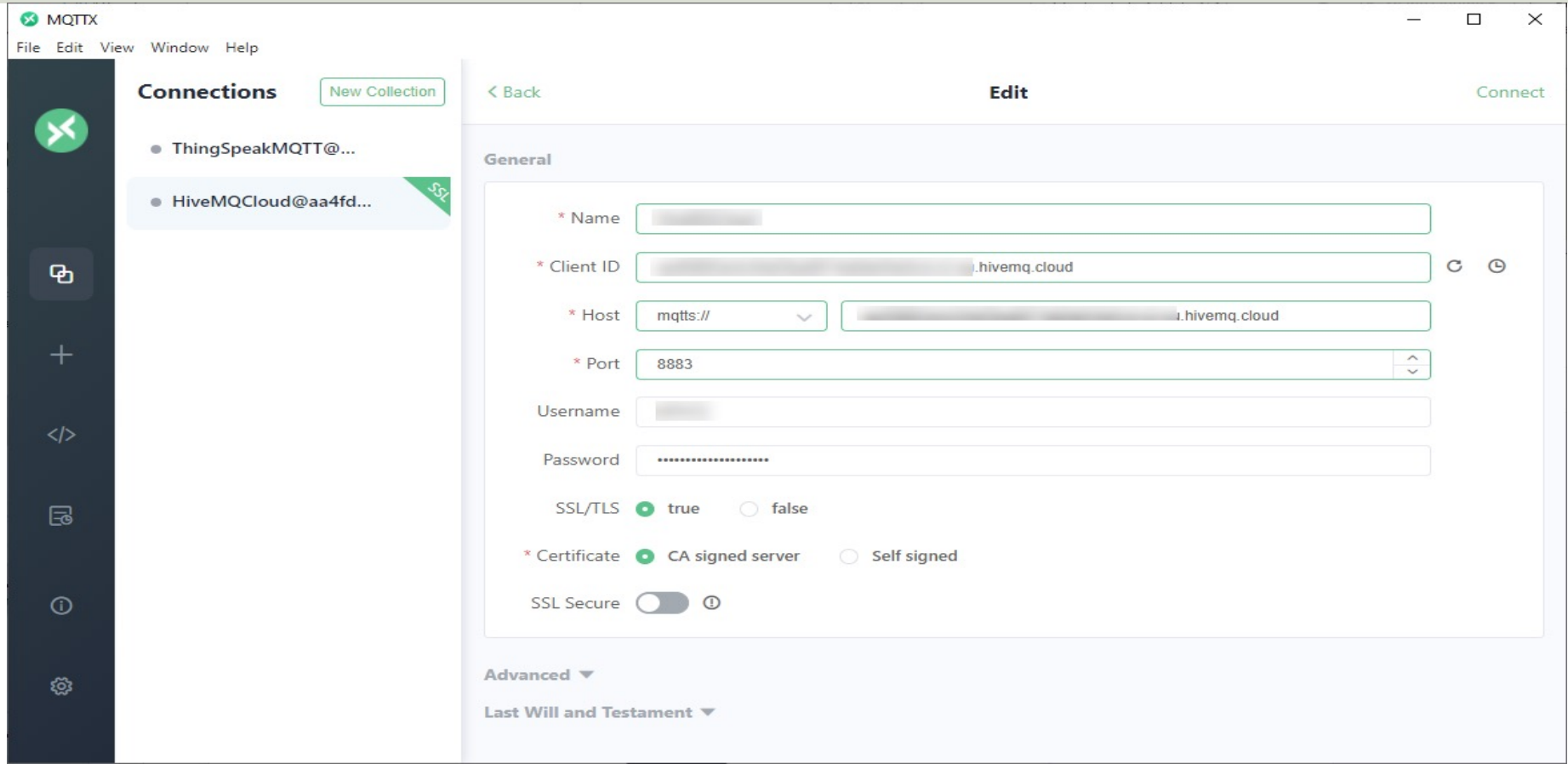
MQTT X

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

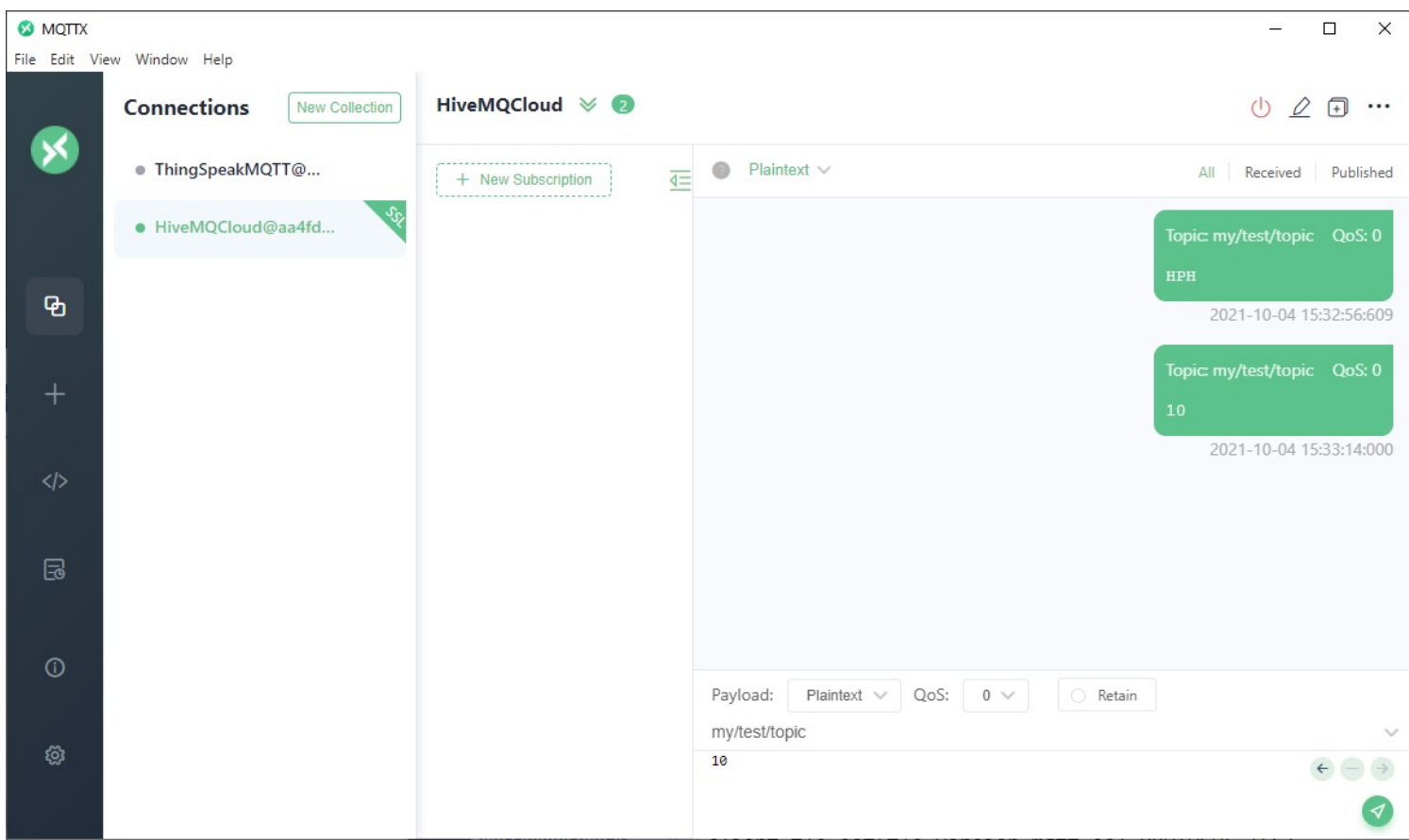
MQTTX



Connect to Broker HiveMQ Cloud using MQTTX Client



Publish to Broker HiveMQ Cloud using MQTTX Client





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

The image shows the Thonny Python Editor's package manager interface. The title bar reads "Manage packages for C:\Users\hansha\AppData\Local\Programs\Thonny\python.exe". The search bar contains "paho.mqtt". The left sidebar lists various Python packages, with "<INSTALL>" selected. The main area displays search results for "paho-mqtt", including the package name, version (5.0/3.1.1), and a description (client class). Below the search results, there is a list of installed packages: astroid, asttokens, bcrypt, bitstring, cffi, colorama, colorzero, cryptography, docutils, ecdsa, esptool, gpiozero, isort, and jedi. At the bottom, there is a list of packages to be installed: mpy-extensions, paramiko, parso, and nin. The "Install" button is visible at the bottom right.

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
- gpiozero
- 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

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/>

<INSTALL>

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

Install

Close

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



HiveMQ Cloud and Python

Hans-Petter Halvorsen

[Table of Contents](#)

HiveMQ Cloud Python Example

```
import paho.mqtt.client as mqtt

brokerAddress = "xxxxx"
userName = "xxxxx"
passWord = "xxxxx"
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, the 'Connections' panel lists two connections: 'ThingSpeakMQTT@...' and 'HiveMQCloud@aa4fd...'. The 'HiveMQCloud' connection is selected. The main panel shows a subscription to the topic 'Sensor/Temperature/TMP36' with a 'Plaintext' payload format. A list of published messages is shown, with the following data points:

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

At the bottom, the 'Payload' field is set to 'Plaintext', 'QoS' is set to '0', and the 'Retain' checkbox is unchecked. The topic 'Sensor/Temperature/TMP36' is entered in the 'Topic' field, and the value '23' is entered in the 'Value' field.

Topic: Sensor/Temperature/TMP36

Data: 21

Data: 22

Data: 23

Python Example

In this Example the Thonny Python Editor has been used

```
mqtt_hivemq_cloud_ex2.py
1 import paho.mqtt.client as mqtt
2
3 brokerAddress = "broker.hivemq.com"
4 userName = "HIVEMQ"
5 passWord = "HIVEMQ@hivemq5"
6
7 topic = "Sensor/Temperature/TMP36"
8 data = 20
9
10 # The callback for when the client receives a CONNACK response from the server.
11 def on_connect(client, userdata, flags, rc):
12     if rc == 0:
13         print("Connected successfully")
14     else:
15         print("Connect returned result code: " + str(rc))
16
17 # The callback for when a PUBLISH message is received from the server.
18 def on_message(client, userdata, msg):
19     print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))
20
21 # create the client
22 client = mqtt.Client()
23 client.on_connect = on_connect
24 client.on_message = on_message
25
26 client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
27 client.username_pw_set(userName, passWord)
28 client.connect(brokerAddress, 8883)
29
30
31 client.subscribe(topic)
32
33 client.publish(topic, data)
34
35
36 client.loop_forever()
```

```
Shell
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
```

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

Publish – Subscribe Examples

The screenshot shows a Windows desktop environment. At the top, the taskbar displays several application icons, including a web browser, a file explorer, and a terminal. The main window is a code editor titled "Publish Temperature to HiveMQ Cloud.py". The code is a Python script that uses the Paho MQTT client library to connect to a HiveMQ cloud instance and publish temperature data.

```
1 import paho.mqtt.client as mqtt
2 import random
3 import time
4
5 brokerAddress = "mqtt://mqtt.hivemq.cloud"
6 userName = "hivm2"
7 password = "hivm2"
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
43
```

Below the code editor is a terminal window titled "Shell". It shows the output of the script, which is a series of random integers between 20 and 30, printed on each line. The terminal also shows the prompt "Python 3.7.9 (bundled)" and a cursor.

```
Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\MQTT\MQTT Python\Subscribe on Topic in HiveMQ Cloud.py @ 2...
File Edit View Run Tools Help

Subscribe on Topic in HiveMQ Cloud.py x
1 import paho.mqtt.client as mqtt
2
3 brokerAddress = "a4a49b3d-7c7b-4388-8000-4499c4499c44@eu.hivemq.cloud"
4 userName = "i"
5 password = "i"
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()
32
33
Shell x
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
```

```
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)
```

Publish

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

```
brokerAddress = "xxxxxxx"
```

```
userName = "xxxxxxx"
```

```
passWord = "xxxxxxx"
```

```
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()
```

Subscribe

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



ThingSpeak

Internet of Things Cloud Service

Hans-Petter Halvorsen

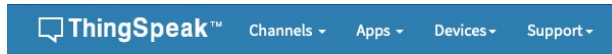
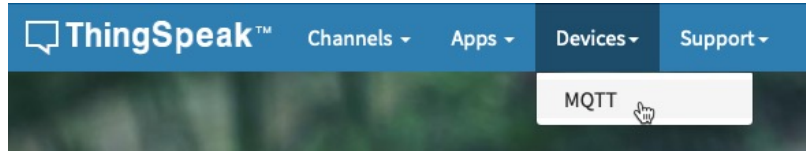
[Table of Contents](#)

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 Device ▾

Add a new device ✕

Device Information

Name*

Description

Tell us more about your device...

Authorize channels to access ⓘ

-- Select a Channel --

...

Add Channel

Authorized Channel ⓘ

Allow Publish

Allow Subscribe

No channels authorized.

Cancel

Add Device

ThingSpeak and MQTTX

The screenshot shows the MQTTX application window. On the left is a dark sidebar with icons for connections, collections, and data. The main area is titled 'Connections' and shows a 'New Collection' button. A 'New' dialog box is open, displaying the 'General' tab for configuring a new MQTT connection. The fields are as follows:

- Name:** A text input field.
- Client ID:** A text input field.
- Host:** A dropdown menu showing 'mqtt://' and a text input field containing 'mqtt3.thingspeak.com'.
- Port:** A text input field containing '1883'.
- Username:** A text input field.
- Password:** A text input field with masked characters.
- SSL/TLS:** Radio buttons for 'true' and 'false', with 'false' selected.

Below the 'General' tab, the 'Advanced' tab is partially visible.

ThingSpeak MQTT Broker:

`mqtt:// mqtt3.thingspeak.com`

Publish to Channel Field

Payload:

Plaintext ▾

QoS:

0 ▾

☐ Retain

channels/<ChannelID>/publish/fields/field1

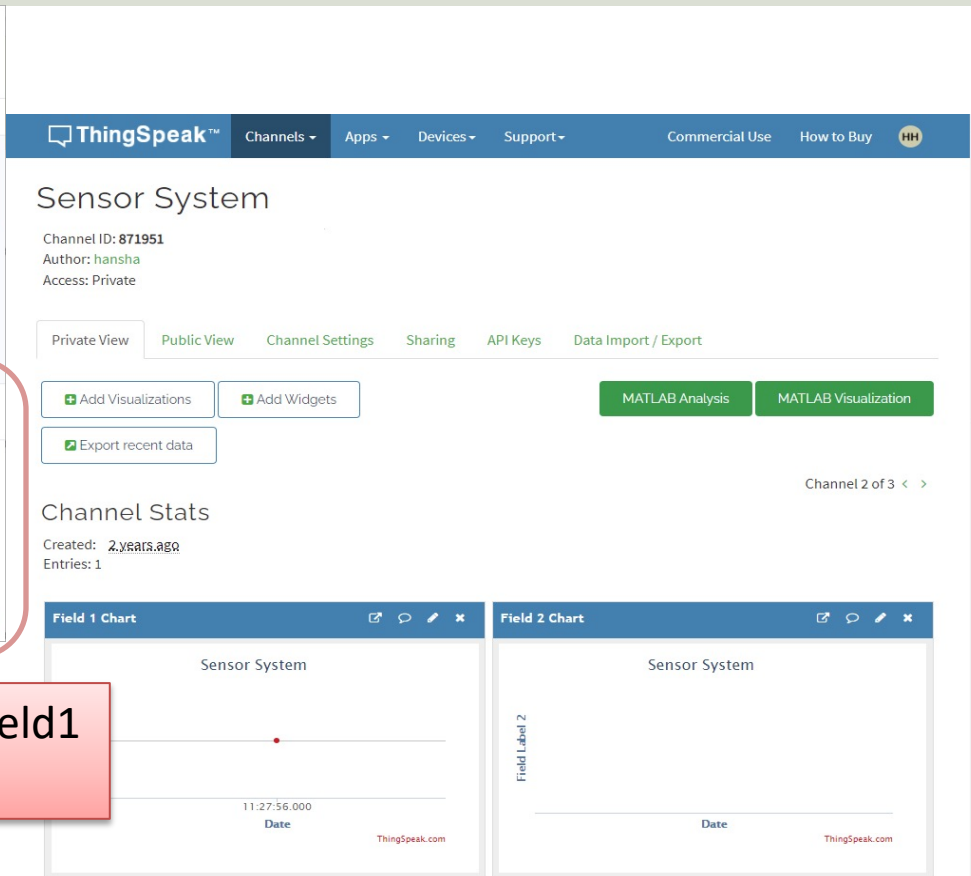
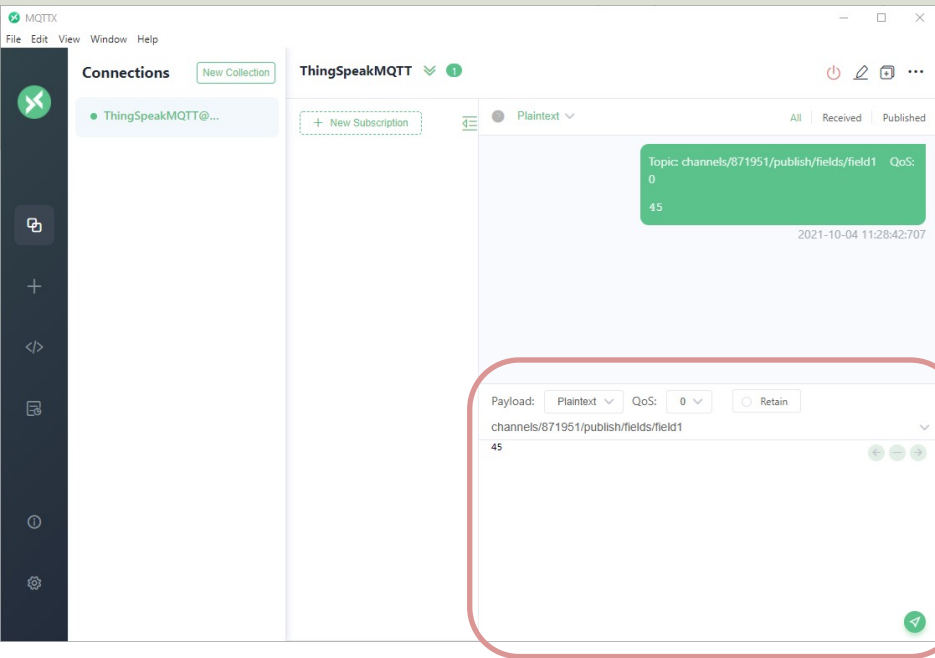


45



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

Publish to Channel Field



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

Publish to Channel Field

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

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

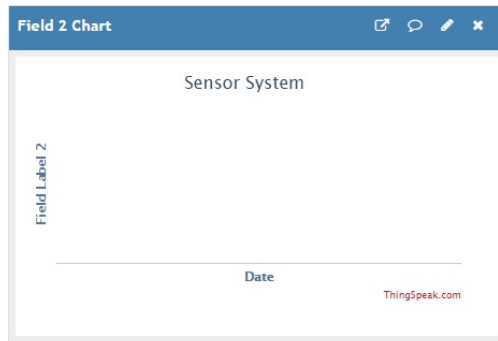
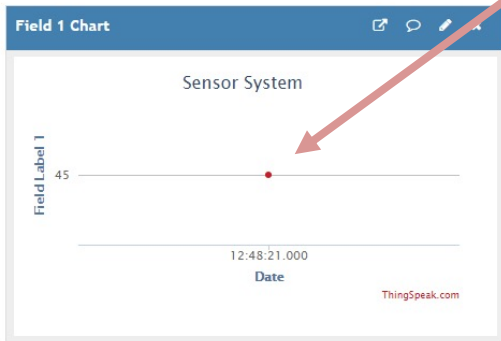
MATLAB Analysis

MATLAB Visualization

Channel 2 of 3 < >

Channel Stats

Created: ~~2 years ago~~
Entries: 1



Publish to Channel Feed

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

The screenshot displays the MQTTX application window. On the left is a dark sidebar with navigation icons. The main area is divided into three panes. The left pane, titled 'Connections', shows a connection to 'ThingSpeakMQTT@...'. The middle pane, titled 'ThingSpeakMQTT', contains a '+ New Subscription' button and a message list. The right pane shows the details of a published message. The message details include the topic 'channels/871951/publish/fields/field1' and the payload '45'. Below this, another message detail shows the topic 'channels/871951/publish' and the payload 'field1=20&field2=30&status=MQTTPUBLISH'. At the bottom, the 'Payload' field is set to 'Plaintext', 'QoS' is '0', and the 'Retain' checkbox is unchecked. The 'Topic' field is set to 'channels/871951/publish' and the 'Payload' field is set to 'field1=20&field2=30&status=MQTTPUBLISH'.

MQTTX

File Edit View Window Help

Connections New Collection

ThingSpeakMQTT 2

+ New Subscription

Plaintext

All Received Published

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

Topic: channels/871951/publish

field1=20&field2=30&status=MQTTPUBLISH

Payload: Plaintext QoS: 0 Retain

Topic: channels/<ChannelID>/publish
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:

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

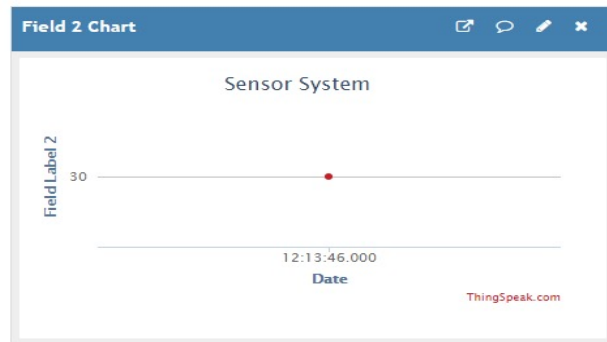
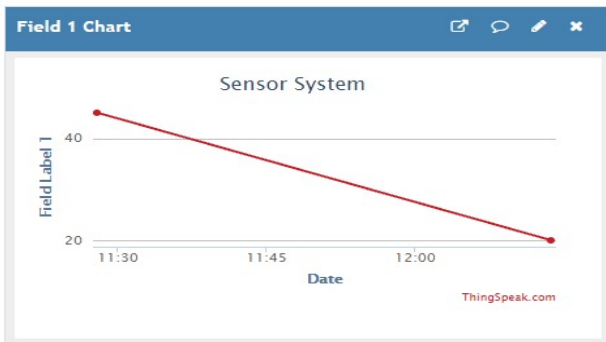
MATLAB Visualization

Channel 2 of 3 < >

Channel Stats

Created: 2 years ago

Entries: 2



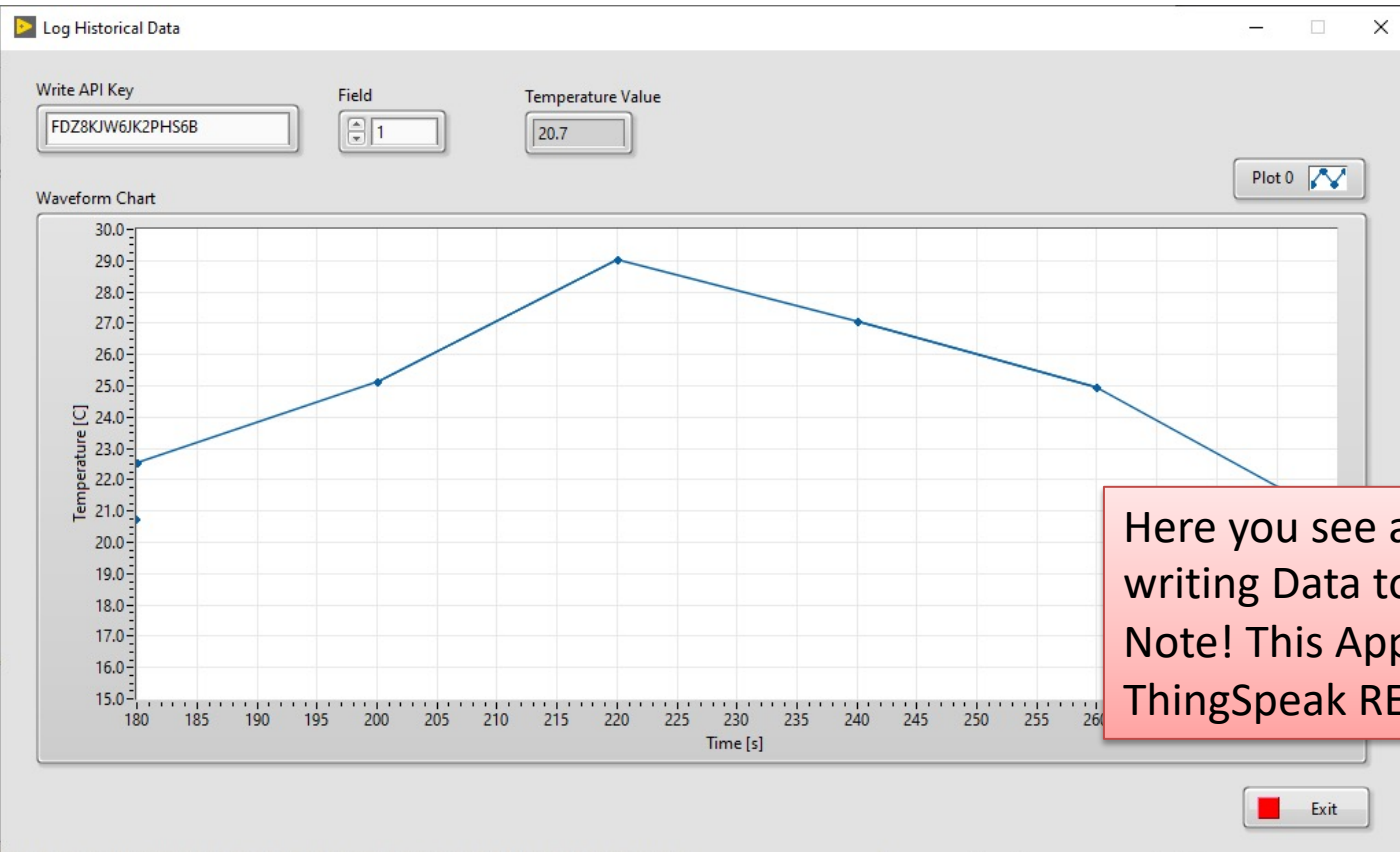
Subscribe to a Channel Feed

The screenshot shows the MQTTX application interface. On the left, a dark sidebar contains icons for connections, collections, and settings. The main window has a menu bar (File, Edit, View, Window, Help) and a title bar (MQTTX). Below the menu bar, there's a 'Connections' section with a 'New Collection' button and a list of connections. One connection, 'ThingSpeakMQTT@...', is selected. A red box highlights a '+ New Subscription' button in the connection's context menu. To the right, a 'New Subscription' dialog box is open. It has a close button (X) in the top right corner. The dialog contains the following fields:

- * Topic**: A text input field containing 'channels/t...l/subscribe'.
- * QoS**: A dropdown menu set to '0'.
- Color**: A color picker set to '#1FC6A0'.
- Alias**: A text input field.

At the bottom right of the dialog, there are 'Cancel' and 'Confirm' buttons.

Subscribe to a Channel Feed



Here you see a LabVIEW Application writing Data to ThingSpeak.
Note! This Application is using the ThingSpeak REST API and not MQTT

Subscribe to a Channel Feed

The screenshot displays the MQTTX application window. On the left is a dark sidebar with icons for connections, collections, subscriptions, and settings. The main area is divided into three panels:

- Connections:** Shows a connection named "ThingSpeakMQTT@...".
- Subscriptions:** A "New Subscription" button is visible. Below it, a subscription is configured for the topic "channels/871951..." with a QoS of 0.
- Message View:** Displays received messages. The first message is a "Publish" action with a green header: "Topic: channels/871951/publish QoS: 0" and a payload: "field1=20&field2=30&status=MQTTPUBLISH". The second message is a "Subscribe" action with a grey header: "Topic: channels/871951/subscribe QoS: 0" and a JSON payload: {"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}. Below the messages, there are controls for "Payload" (set to Plaintext), "QoS" (set to 0), and a "Retain" checkbox.



ThingSpeak and Python

Hans-Petter Halvorsen

[Table of Contents](#)

Thonny - 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 x

```
1 import paho.mqtt.client as mqtt
2 import random
3 import time
4
5 brokerAddress = "mqtt3.thingspeak.com"
6 port = 1883
7 clientId = "xxxxxxxxxxxxxxxxxxxx"
8 userName = "xxxxxxxxxxxxxxxxxxxx"
9 password = "xxxxxxxxxxxx"
10 channelID = "571051"
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 unit = 20
```

Shell x

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

Python 3.7.9

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)
```

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 = "EgYtXhYjA4V1R3Q00015o"
6 userName = "EgYtXhYjA4V1R3Q00015o"
7 passWord = "17-LEFMYG14GHTd3m3tpA3D"
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

```
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 -> 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 = "xxxxxxx"
userName = "xxxxxxx"
passWord = "xxxxxxx"
channelID = "xxxxxxx"
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

```
Thorny - 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 = "Logg...y...v...q...a...30"
8 userName = "Logg...y...v...q...a...30"
9 password = "Logg...y...v...q...a...30"
10 channelId = "871951"
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 while True:
33     min = 20
34     max = 30
35     value = random.randint(min, max)
36     payload = str(value)
37     client.publish(topic, payload)
38     time.sleep(10)
39
40 Shell ×
41 25
42 24
43 30
44 22
45 28
46 27
47 25
Python 3.7.9
```

Publish

```
Thorny - C:\Users\halvorsen\OneDrive\Documents\Industrial IT and Automation\MQTT\MQTT ThingSpeak\Python Examples\Subscribe on Topic in ThingSpeak.py @ 34:1
<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 = "Logg...y...v...q...a...30"
6 userName = "Logg...y...v...q...a...30"
7 password = "Logg...y...v...q...a...30"
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
33 Shell ×
Python 3.7.9 (bundled)
>>> %cd 'C:\Users\halvorsen\OneDrive\Documents\Industrial IT and Automation\MQTT\MQTT ThingSpeak\Python Examples'
>>> %cd 'C:\Users\halvorsen\OneDrive\Documents\Industrial IT and Automation\MQTT\MQTT ThingSpeak\Python Examples'
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 -> 30
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
```

Subscribe

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>

