

<https://www.halvorsen.blog>



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](#)

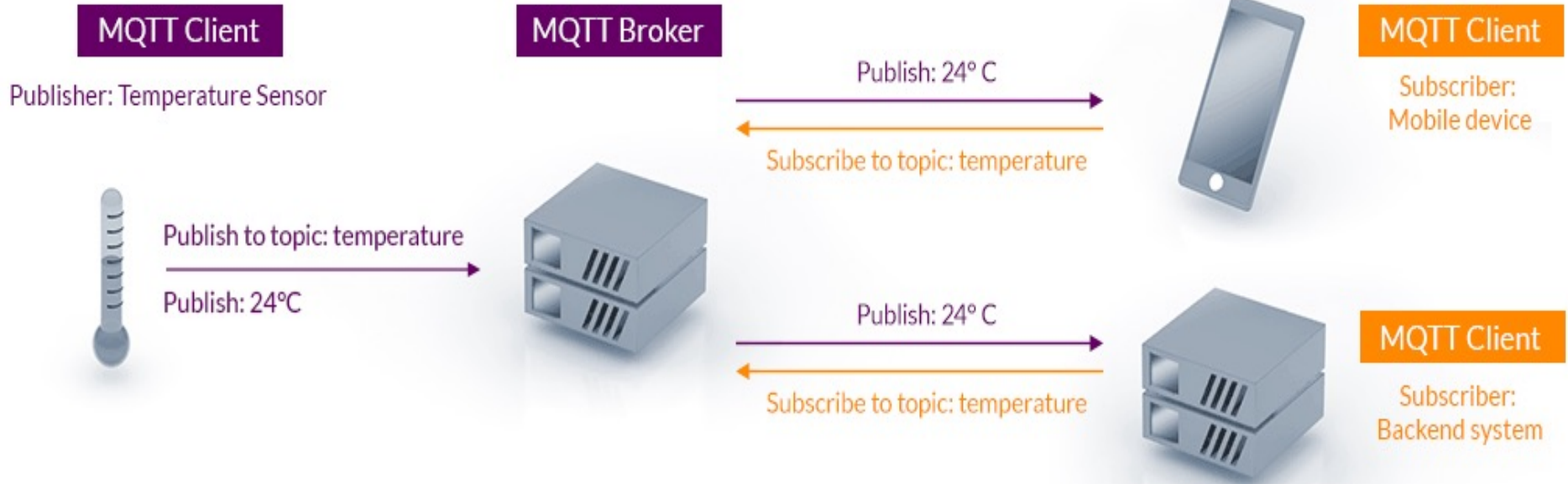


MQTT

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



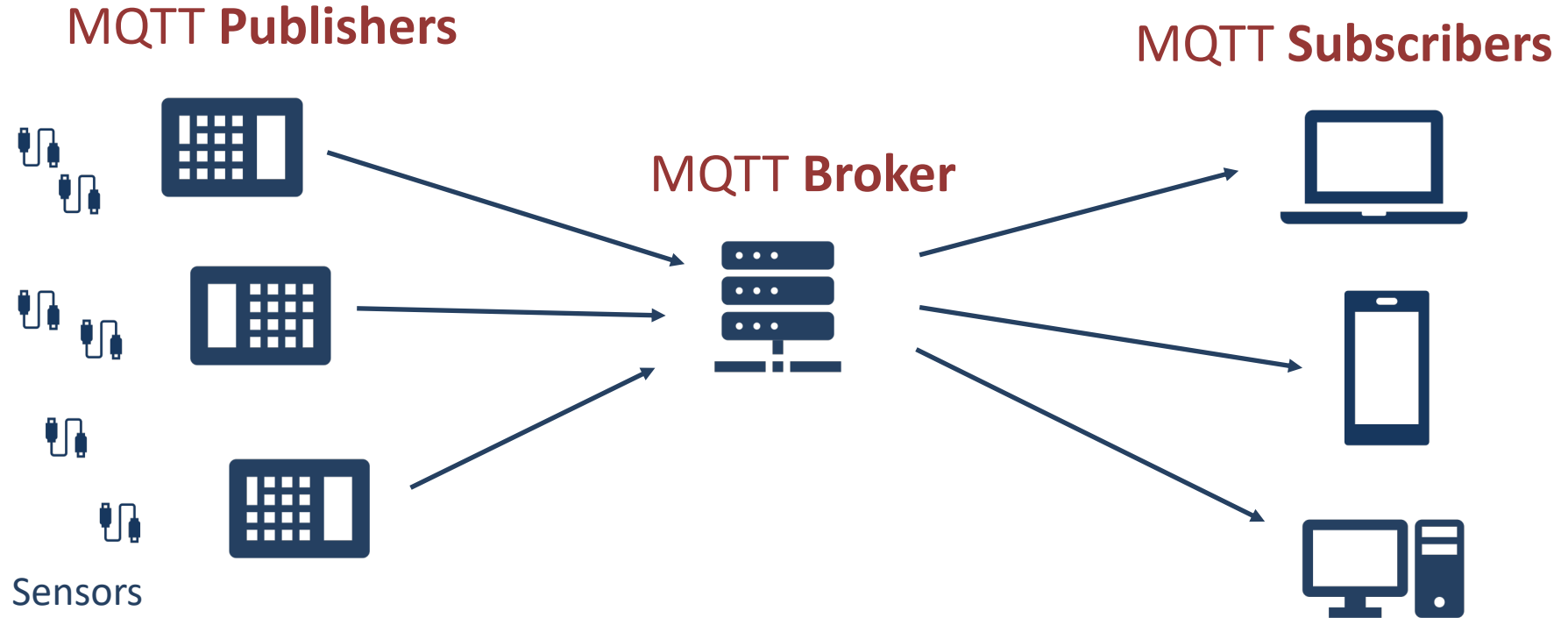
MQTT

- 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

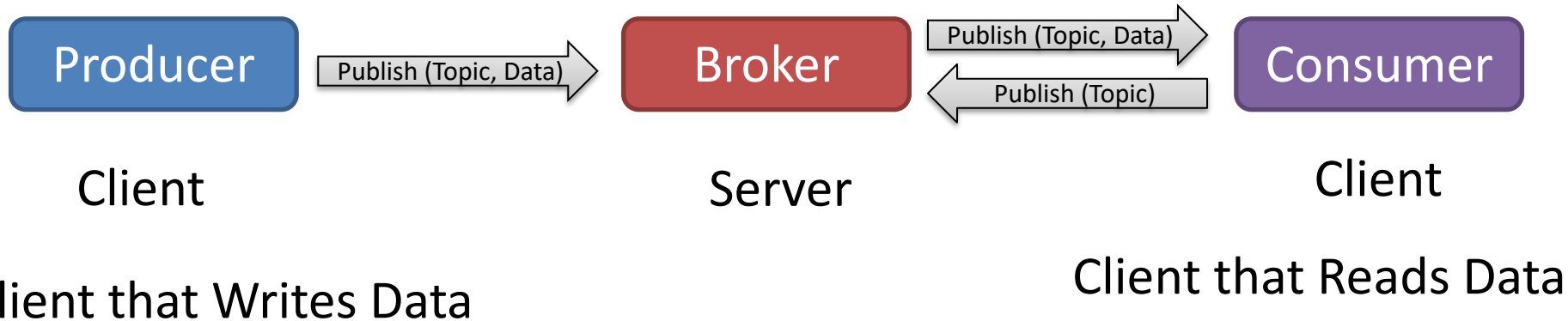
- Internet of Things (IoT): To get data to and from devices on a network.
- MQTT is a lightweight protocol that makes this easier

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

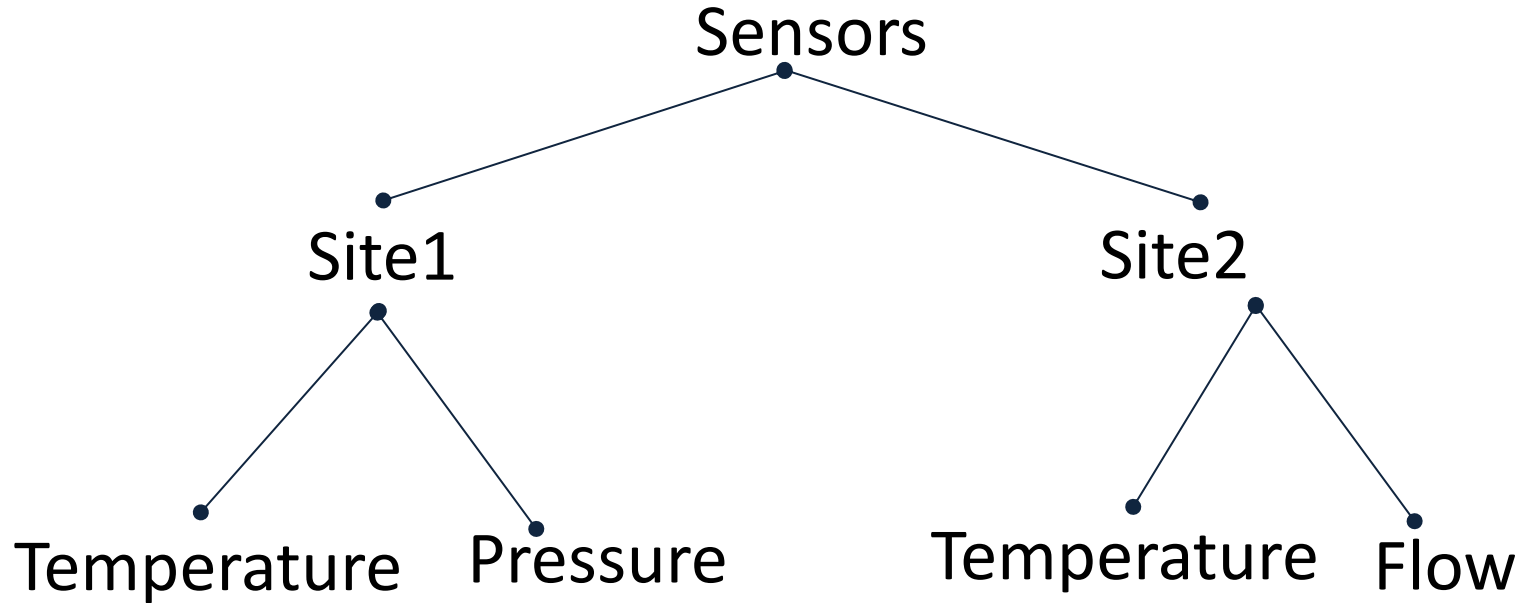
Example:

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

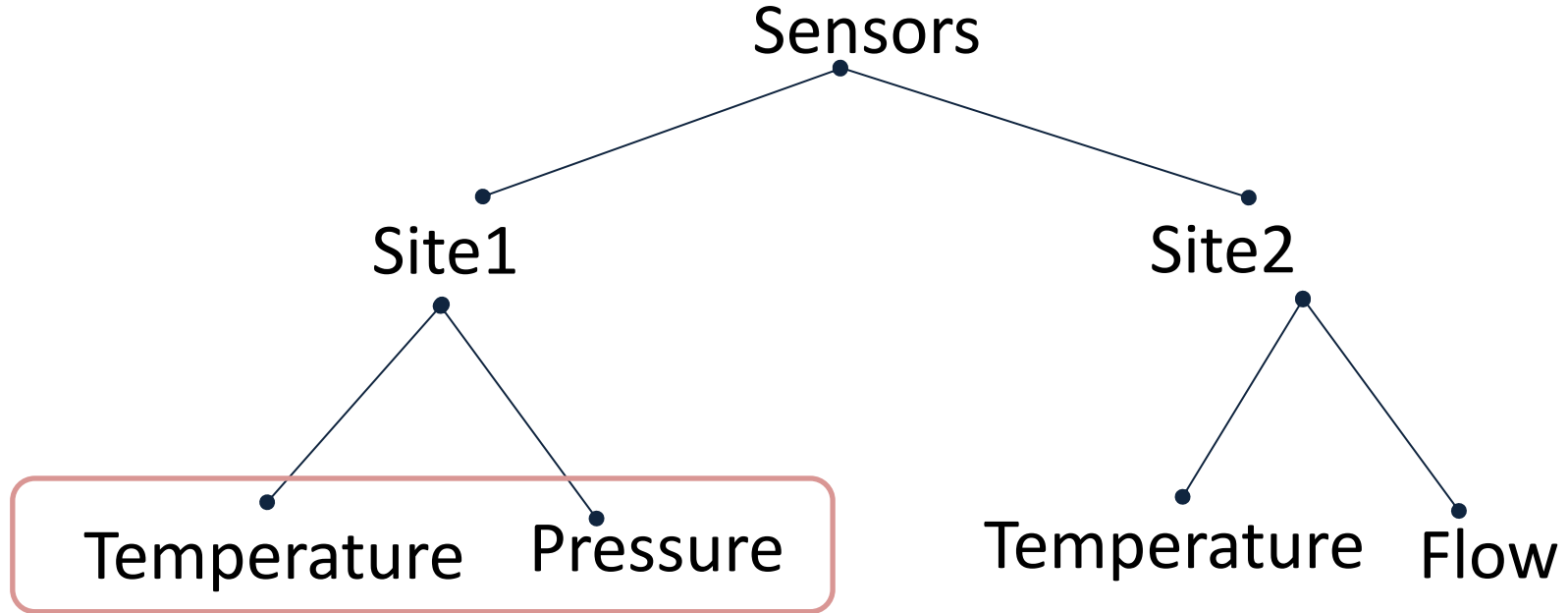
MQTT Topics

Topics are used to organize the data



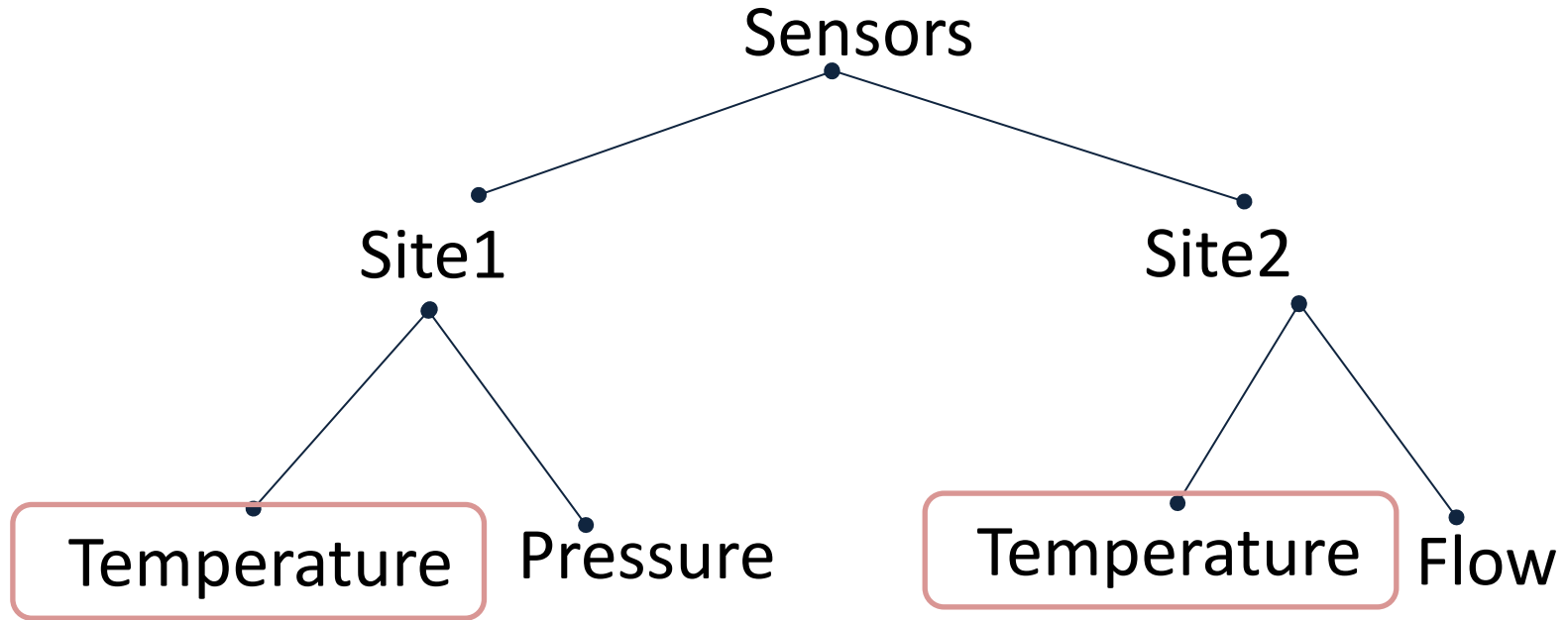
Subscribe on Topics - Wildcards

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

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

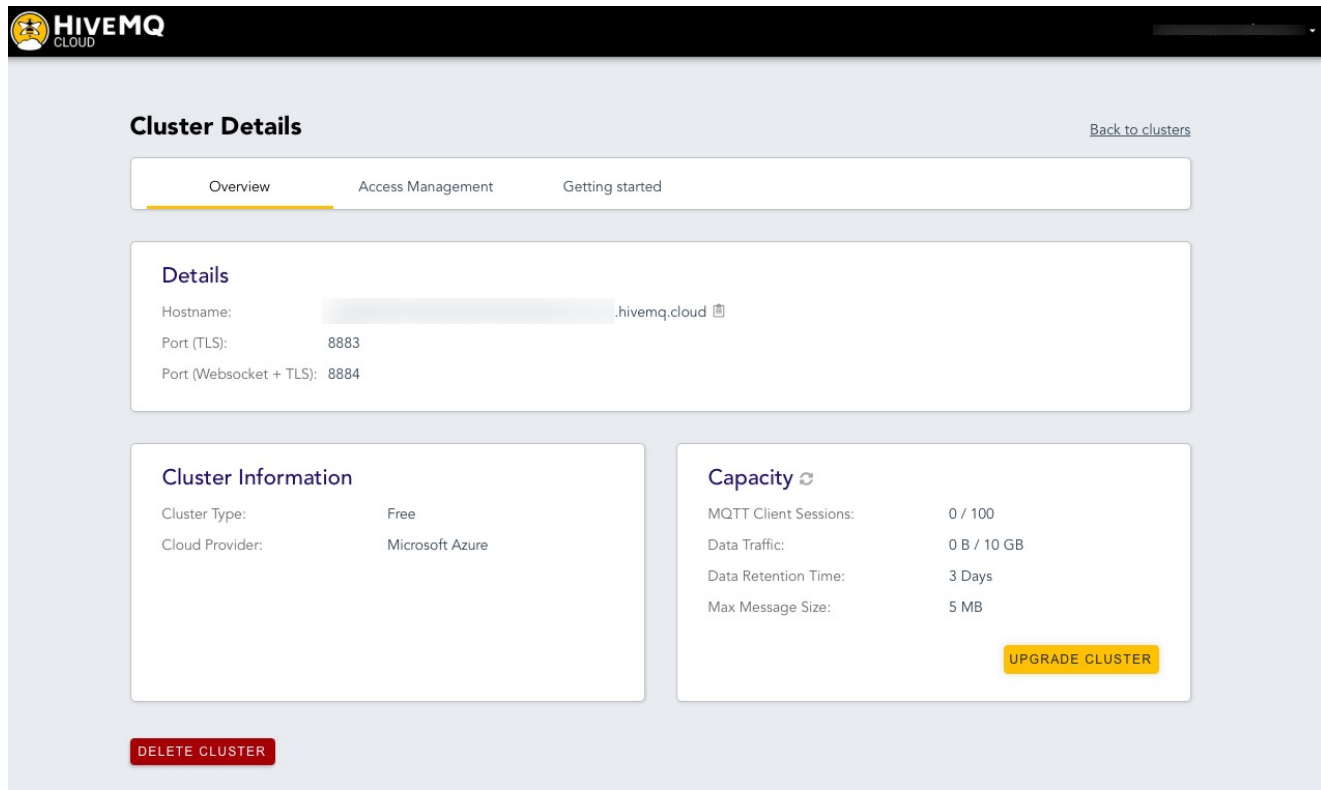
MQTT Broker in the Cloud

Hans-Petter Halvorsen

[Table of Contents](#)

HiveMQ Cloud

<https://www.hivemq.com>



The screenshot displays the HiveMQ Cloud interface. At the top left is the HiveMQ Cloud logo. The main heading is "Cluster Details" with a "Back to clusters" link on the right. Below this is a navigation bar with three tabs: "Overview" (selected), "Access Management", and "Getting started". The "Details" section lists the following information:

- Hostname: [redacted].hivemq.cloud
- Port (TLS): 8883
- Port (Websocket + TLS): 8884

The "Cluster Information" section shows:

- Cluster Type: Free
- Cloud Provider: Microsoft Azure

The "Capacity" section shows:

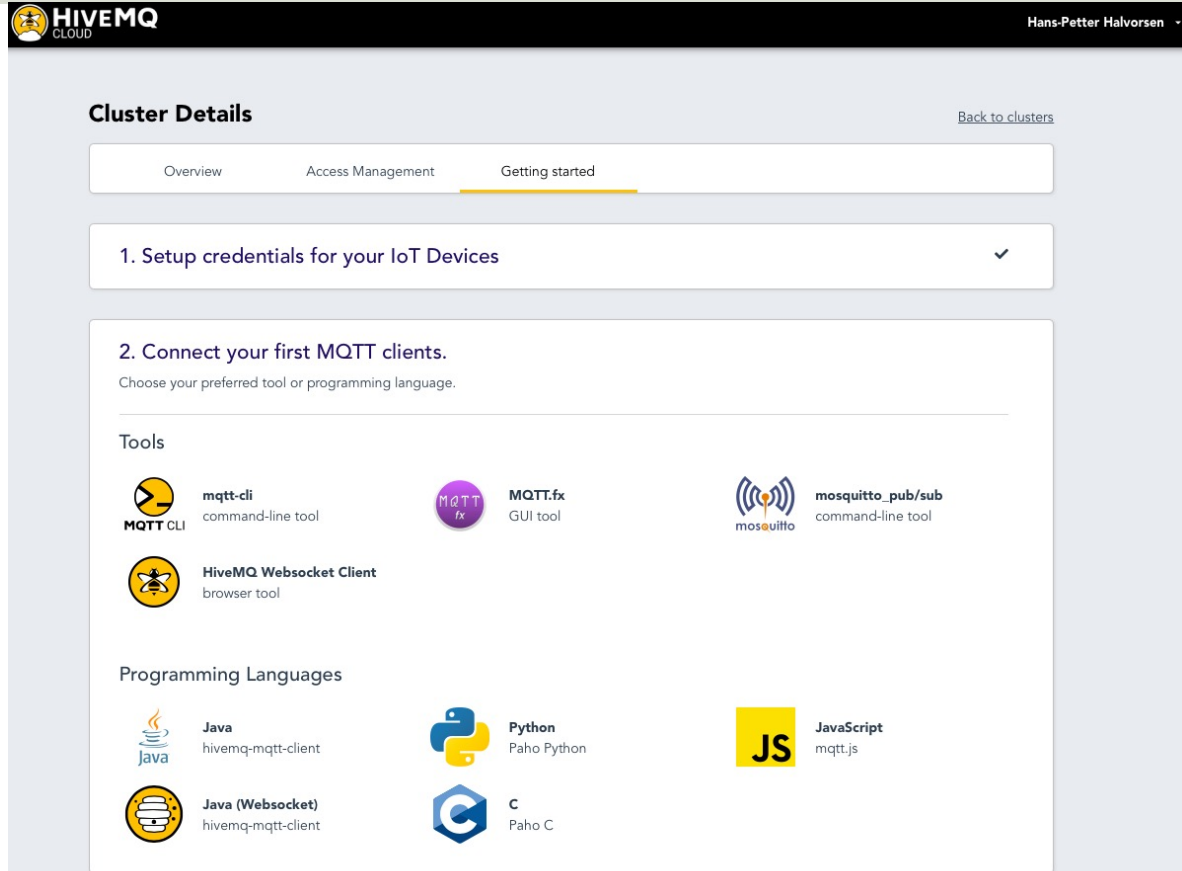
- MQTT Client Sessions: 0 / 100
- Data Traffic: 0 B / 10 GB
- Data Retention Time: 3 Days
- Max Message Size: 5 MB

At the bottom of the capacity section is a yellow "UPGRADE CLUSTER" button. At the bottom left of the dashboard is a red "DELETE CLUSTER" button.

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 MQTT CLI command-line tool	 MQTT.fx GUI tool	 mosquitto_pub/sub mosquitto 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

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>

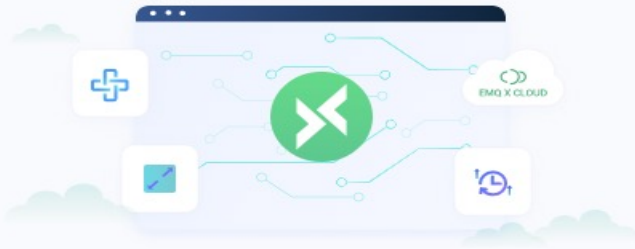
MQTTX

MQTTX

File Edit View Window Help

Connections New Collection

No Data



+ New Connection

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

The image shows the MQTTX application window. The title bar reads 'MQTTX' with standard window controls. The menu bar includes 'File', 'Edit', 'View', 'Window', and 'Help'. On the left is a dark sidebar with icons for a connection, collections, a plus sign, code symbols, a document, an info icon, and a settings gear. The main area is titled 'Connections' and contains a 'New Collection' button. Below this, it says 'No Data'. The central part of the window features a large graphic with a central MQTT icon connected to various service icons, including 'EMQ X CLOUD'. At the bottom of this graphic is a green '+ New Connection' button. Below the graphic, there is a promotional message: 'Need a Cloud-Native fully-managed MQTT broker? Try EMQ X Cloud now!'.

Connect to Broker HiveMQ Cloud using MQTTX Client

The screenshot displays the MQTTX application window. The title bar reads "MQTTX" and the menu bar includes "File", "Edit", "View", "Window", and "Help". On the left, a dark sidebar contains navigation icons: a green circle with a white arrow, a square with a white plus sign, a white plus sign, a white left-right arrow, a white document icon, a white information icon, and a white gear icon. The main area is titled "Connections" and features a "New Collection" button. Two connection entries are listed: "ThingSpeakMQTT@..." and "HiveMQCloud@aa4fd...", with the latter marked with a green "SSL" badge. The "HiveMQCloud@aa4fd..." connection is selected, and its configuration is shown in the "Edit" view. The "General" section includes fields for Name, Client ID (with a refresh icon), Host (a dropdown menu set to "mqtt://", followed by a text field containing ".hivemq.cloud"), Port (set to "8883"), Username, and Password (masked with dots). Below these are radio buttons for "SSL/TLS" (set to "true") and "Certificate" (set to "CA signed server"). An "SSL Secure" toggle switch is also present. At the bottom, there are sections for "Advanced" and "Last Will and Testament", both with dropdown arrows.

MQTTX

File Edit View Window Help

Connections New Collection

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

Edit Connect

General

* Name

* Client ID ↻ 🔒

* Host mqtt://

* Port

Username

Password

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

The screenshot displays the MQTTX application interface. On the left, a dark sidebar contains navigation icons. The main window is titled "MQTTX" and includes a menu bar with "File", "Edit", "View", "Window", and "Help".

The "Connections" panel on the left shows two connections: "ThingSpeakMQTT@..." and "HiveMQCloud@aa4fd...", with the latter being the active connection. A "New Collection" button is visible above the connections list.

The central pane shows the "HiveMQCloud" connection selected. A "New Subscription" button is present. The "Payload" section is set to "Plaintext".

The message history on the right shows two published messages to the topic "my/test/topic":

- Message 1: Payload "HPH", QoS: 0, timestamp "2021-10-04 15:32:56:609".
- Message 2: Payload "10", QoS: 0, timestamp "2021-10-04 15:33:14:000".

At the bottom, the "Payload" field is set to "Plaintext", "QoS" is "0", and the "Retain" checkbox is unchecked. The topic "my/test/topic" and the payload "10" are visible in the bottom right corner.



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 two overlapping windows of the Thonny Python Editor's package manager. The background window displays search results for 'paho-mqtt', listing several packages with their descriptions. The foreground window shows the details for the selected 'paho-mqtt' package, including its latest stable version (1.5.1), summary, author, homepage, and PyPI page. A red text box is overlaid on the bottom of the windows, providing instructions on how to install the library.

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

paho-mqtt

<INSTALL>

- astroid
- asttokens
- bcrypt
- bitstring
- cff
- colorama
- colorzero
- cryptography
- docutils
- ecdsa
- esptool
- gpiozero
- isort
- ipython

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

Search on PyPI

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
- cff
- colorama
- colorzero
- cryptography
- docutils
- ecdsa
- mypy-extensions
- paramiko
- parso
- nin

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

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 displays the MQTTX application interface. On the left, a sidebar contains navigation icons. The main window shows a 'Connections' panel with two connections: 'ThingSpeakMQTT@...' and 'HiveMQCloud@aa4fd...'. The 'HiveMQCloud' connection is selected, and a 'New Subscription' button is visible. The main area shows a 'Plaintext' subscription with a list of published messages. The messages are displayed in green boxes with the following details:

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

At the bottom, the 'Payload' is set to 'Plaintext', 'QoS' is 0, and the 'Retain' option is unchecked. The topic 'Sensor/Temperature/TMP36' is displayed, and the number '23' is shown at the bottom left of the message list.

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 x
1 import paho.mqtt.client as mqtt
2
3 brokerAddress = "XXXXXXXXXXXXXXXXXXXXX.hivemq.cloud"
4 userName = "XXXXXXXX"
5 password = "XXXXXXXXXXXXXXXXXXXXX"
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 x
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


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

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

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

<https://www.halvorsen.blog>



ThingSpeak

Internet of Things Cloud Service

Hans-Petter Halvorsen

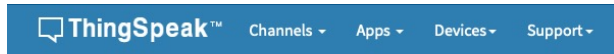
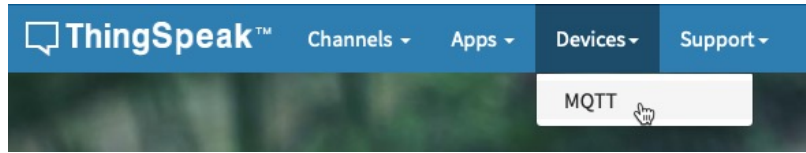
[Table of Contents](#)

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 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. The title bar reads 'MQTX'. The menu bar includes 'File', 'Edit', 'View', 'Window', and 'Help'. On the left is a dark sidebar with icons for home, connections, add, code, data, info, and settings. The main area is titled 'Connections' and contains a 'New Collection' button. A 'New' dialog is open, showing configuration fields: Name, Client ID, Host (set to 'mqtt3.thingspeak.com'), Port (set to '1883'), Username, Password, and SSL/TLS (set to 'false').

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 MQTTX interface shows a connection to ThingSpeakMQTT. A message is published to the topic `channels/871951/publish/fields/field1` with a QoS of 0 and a payload of 45. The message is displayed in a green box with the timestamp 2021-10-04 11:28:42:707. Below the message, the payload is shown as `channels/871951/publish/fields/field1` and the value `45`. The interface includes a 'Connections' panel on the left and a 'ThingSpeakMQTT' connection status at the top.

The ThingSpeak channel page for 'Sensor System' (Channel ID: 871951) is shown. The channel is private and created 2 years ago. The page includes options for 'Private View', 'Public View', 'Channel Settings', 'Sharing', 'API Keys', and 'Data Import / Export'. There are buttons for 'Add Visualizations', 'Add Widgets', and 'Export recent data'. Two charts are visible: 'Field 1 Chart' and 'Field 2 Chart'. The 'Field 1 Chart' shows a single data point at 11:27:56.000 with a value of 45. The 'Field 2 Chart' is currently empty.

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

Publish to Channel Field

Sensor System

Channel ID: 871951
Author: hansha
Access: Private

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

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

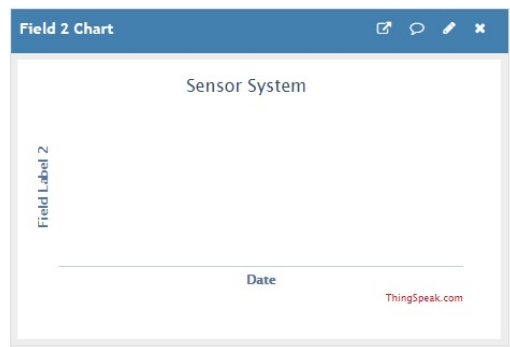
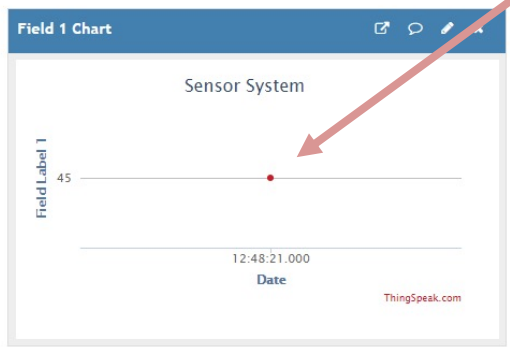
+ Add Visualizations + Add Widgets Export recent data

MATLAB Analysis MATLAB Visualization

Channel 2 of 3 < >

Channel Stats

Created: 2 years ago
Entries: 1



Publish to Channel Feed

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

The screenshot shows the MQTTX application interface. On the left, there is a sidebar with navigation icons. The main window is titled 'MQTTX' and has a menu bar with 'File', 'Edit', 'View', 'Window', and 'Help'. Below the menu bar, there is a 'Connections' section with a 'New Collection' button and a connection named 'ThingSpeakMQTT@...'. The main area is titled 'ThingSpeakMQTT' and has a 'New Subscription' button. A subscription is shown for 'Plaintext' with a 'QoS: 0' dropdown. The message history shows two received messages:

- Message 1: Topic: channels/871951/publish/fields/field1, QoS: 0, Payload: 45, Received at: 2021-10-04 11:28:42:707
- Message 2: Topic: channels/871951/publish, QoS: 0, 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 subscription topic is 'channels/871951/publish' and the payload is 'field1=20&field2=30&status=MQTTPUBLISH'.

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

Publish to Channel Feed

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

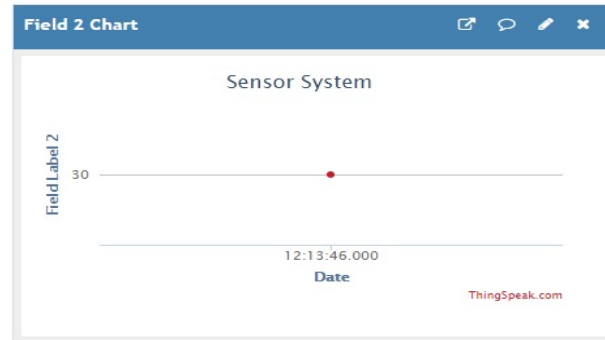
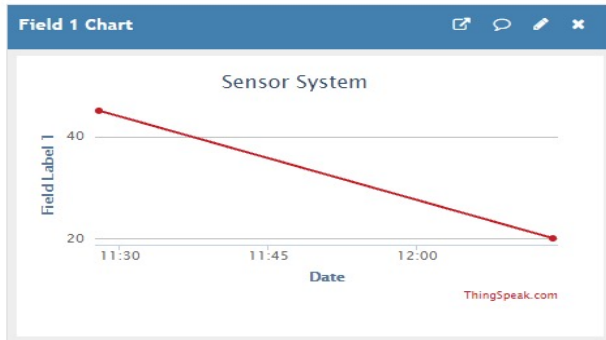
MATLAB Visualization

Channel 2 of 3 < >

Channel Stats

Created: [2 years ago](#)

Entries: 2



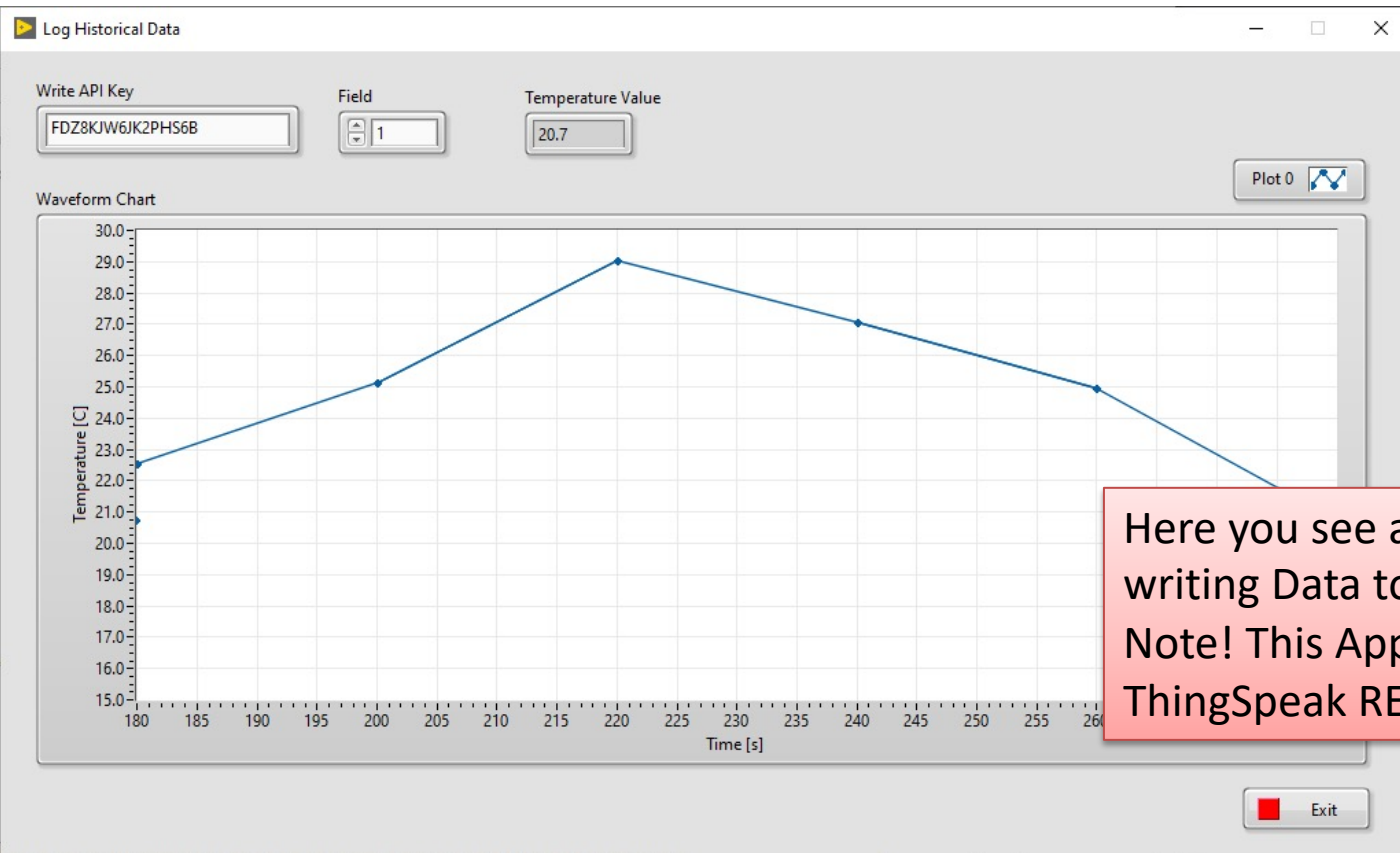
Subscribe to a Channel Feed

The screenshot shows the MQTTX application interface. On the left, a sidebar contains navigation icons. The main window displays a 'Connections' panel with a 'ThingSpeakMQTT' connection. A '+ New Subscription' button is highlighted with a red box. A 'New Subscription' dialog box is open, showing the following configuration:

- Topic:** channels/1/subscribe
- QoS:** 0
- Color:** #1FC6A0
- Alias:** (empty)

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 interface. On the left is a dark sidebar with navigation icons. The main window is titled 'MQTTX' and has a menu bar with 'File', 'Edit', 'View', 'Window', and 'Help'. Below the menu bar, there's a 'Connections' section with a 'New Collection' button and a connection named 'ThingSpeakMQTT@...'. The main area is titled 'ThingSpeakMQTT' and shows a subscription configuration for the topic 'channels/871951...' with a QoS of 0. A 'New Subscription' button is visible. The subscription is set to 'Plaintext' and has a 'Received' status. The message content is shown in a light blue box, containing a JSON object with various fields. The timestamp '2021-10-04 12:37:58:642' is displayed below the message. At the bottom, there are controls for 'Payload' (set to Plaintext), 'QoS' (set to 0), and 'Retain' (unchecked). The 'Topic' field is empty. Navigation arrows are visible at the bottom right.

MQTTX

File Edit View Window Help

Connections New Collection

- ThingSpeakMQTT@...

ThingSpeakMQTT 3

+ New Subscription

channels/871951... QoS 0

Plaintext All Received Published

Topic: channels/871951/subscribe QoS: 0

```
field1=20&field2=30&status=MQTTPUBLISH
```

2021-10-04 12:14:33:077

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



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 = "571050"
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
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 = "EgYtXyYmV1ZnR5Q0001j0"
6 userName = "EgYtXyYmV1ZnR5Q0001j0"
7 password = "EgYtXyYmV1ZnR5Q0001j0A3D"
8 channelId = "1234567890"
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/Subscribe on Topic in ThingSpeak.py'
>>> %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

Publish

```

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 = "LogL2Xy9w7v2h9Q00a1J0"
8 userName = "LogL2Xy9w7v2h9Q00a1J0"
9 password = "5y6L87MVC164U7dub3bA3D"
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 min = 20

```

```

Shell
25
24
30
22
27
28
27
25

```

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 = "LogL2Xy9w7v2h9Q00a1J0"
6 userName = "LogL2Xy9w7v2h9Q00a1J0"
7 password = "5y6L87MVC164U7dub3bA3D"
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 'C:\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

```


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>

