https://www.halvorsen.blog

# MQTT

A Communication Protocol popular in Internet of Things Applications

Hans-Petter Halvorsen

# Contents

https://www.halvorsen.blog

# MQTT

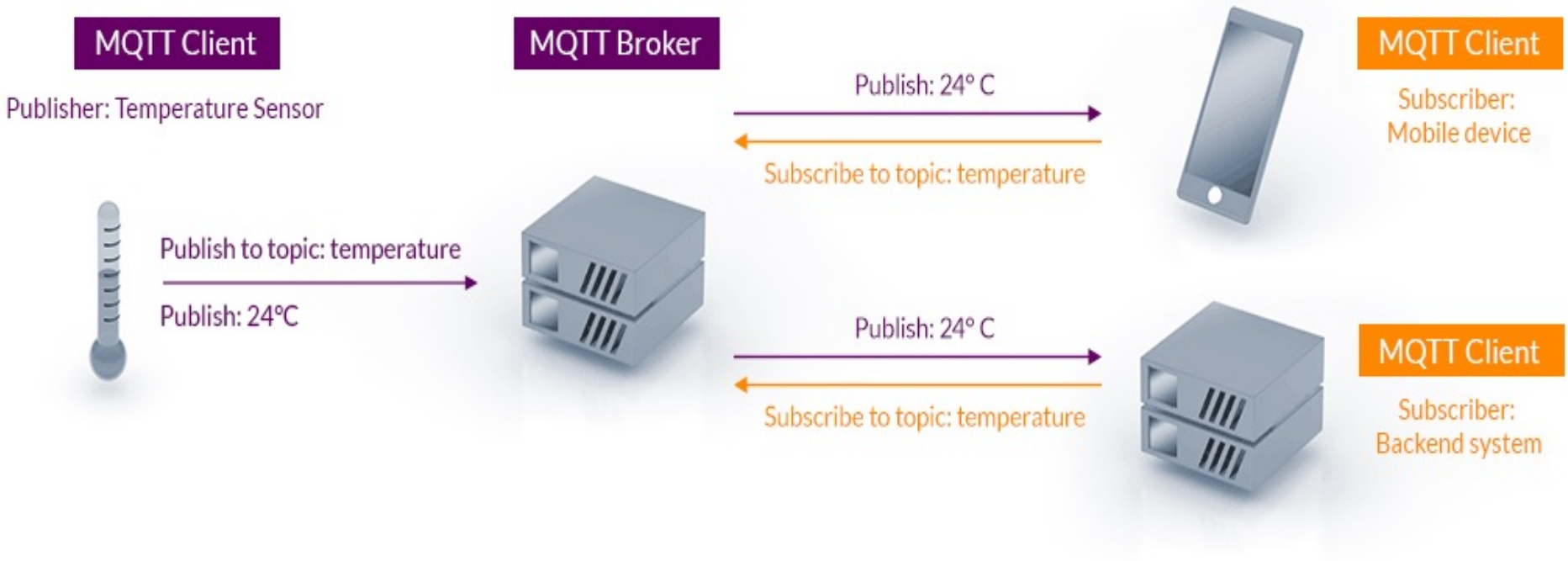
Hans-Petter Halvorsen

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




https://mqtt.org

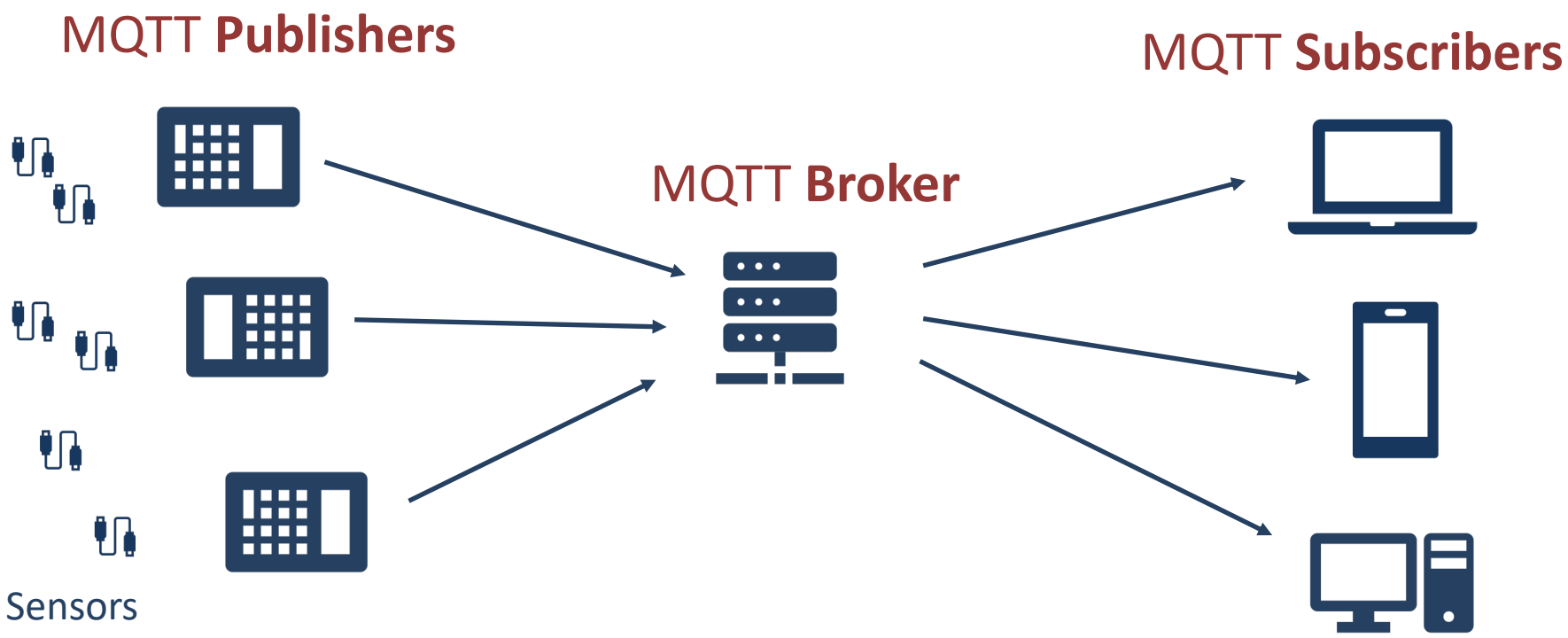
# MQTT

- 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.


Client

Server

Client

Client that Writes Data

Client that Reads Data

# 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

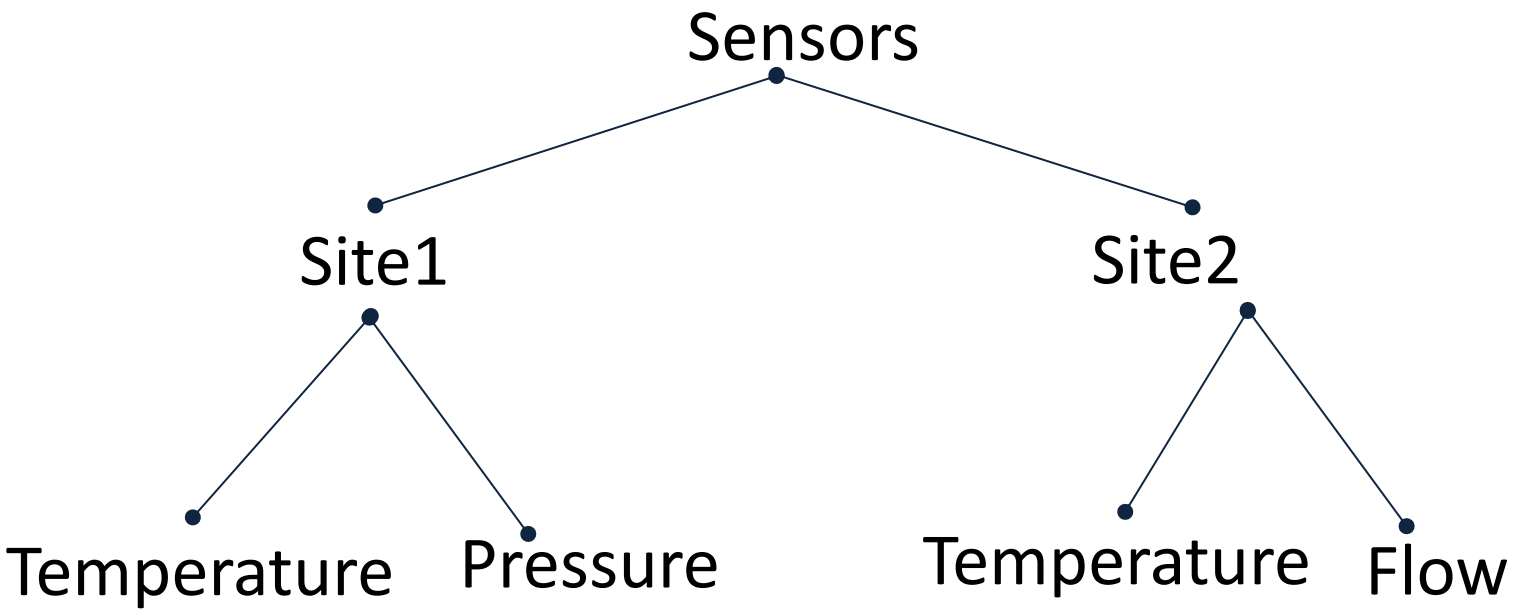
Example:

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

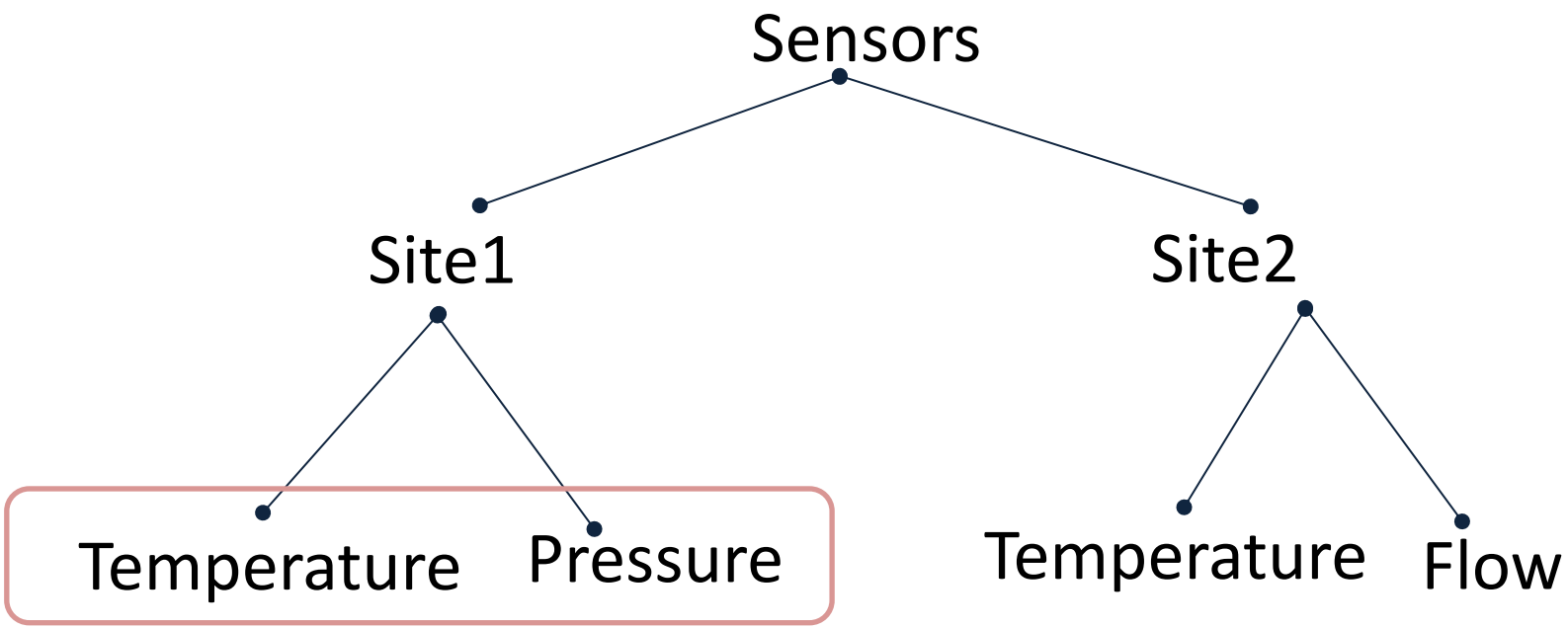
# MQTT Topics

Topics are used to organize the data



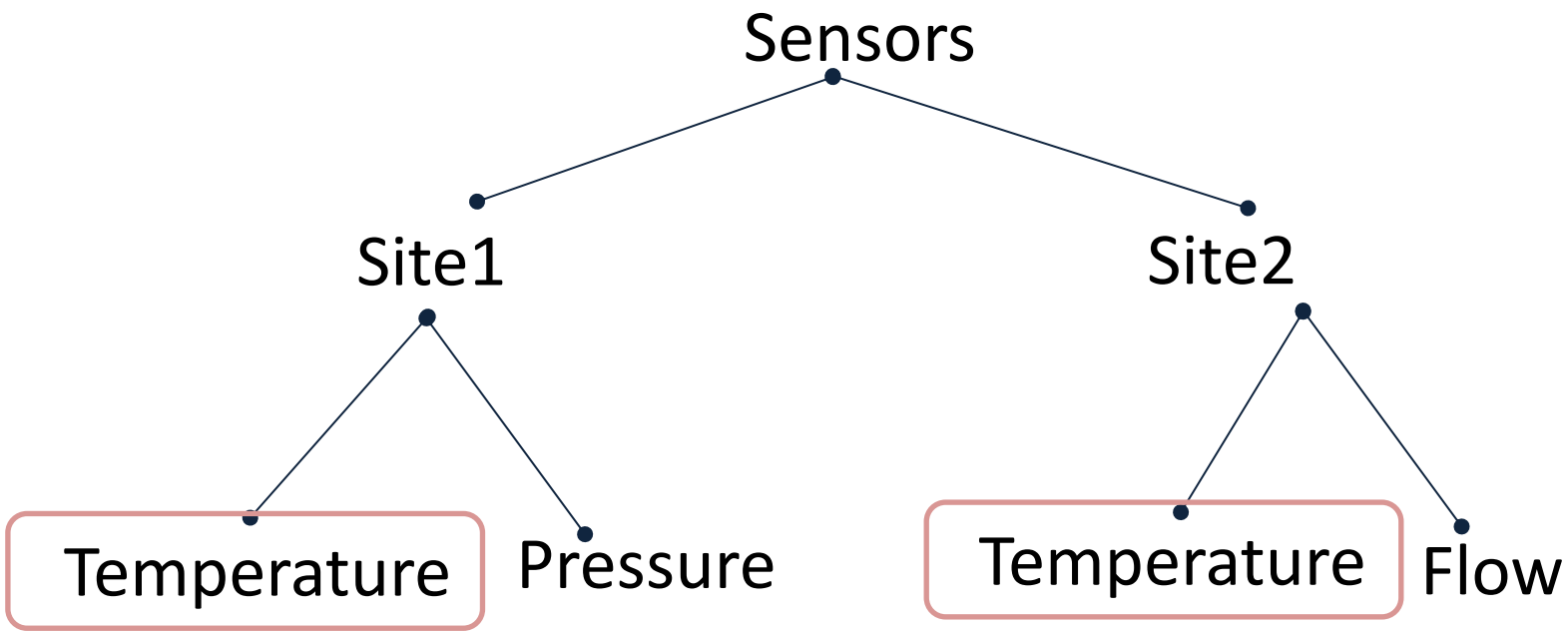
# Subscribe on Topics - Wildcards

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

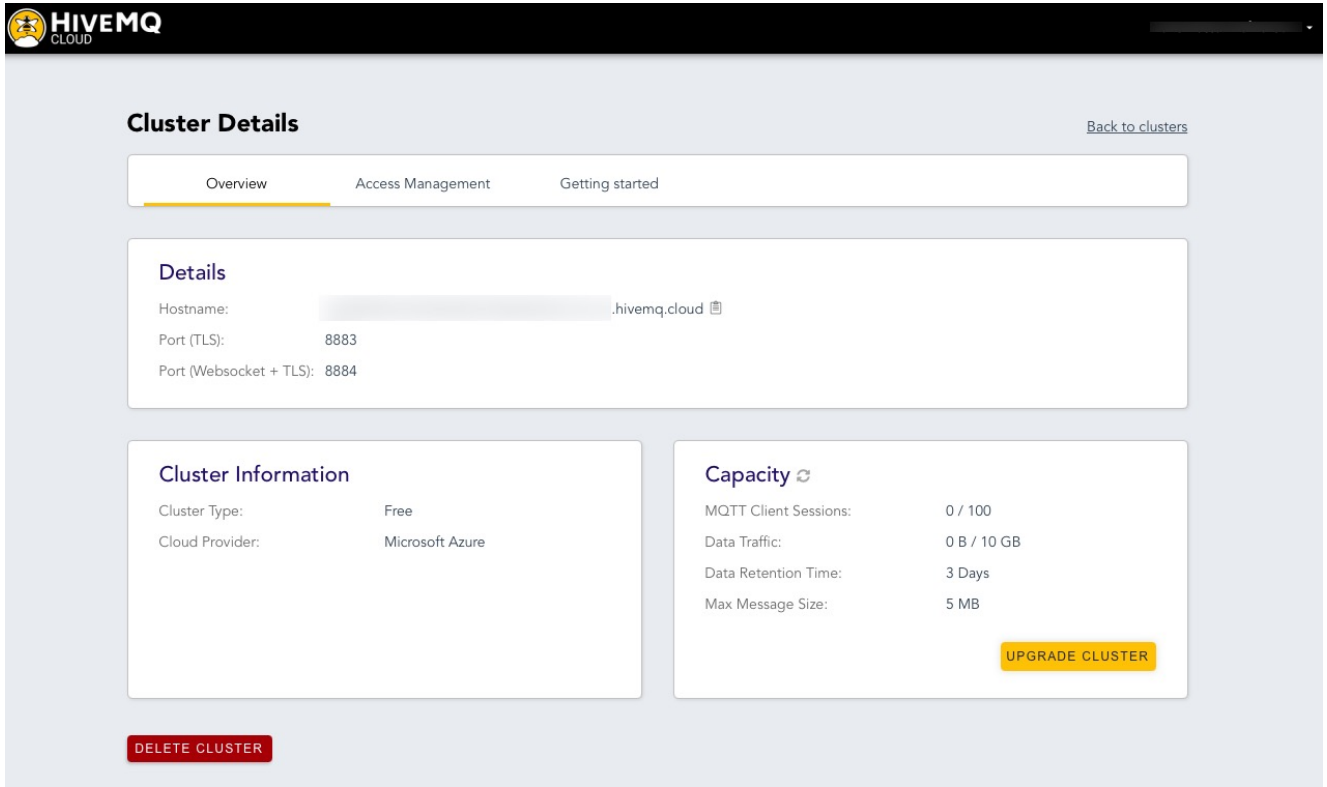
## MQTT Broker in the Cloud

Hans-Petter Halvorsen

Table of Contents

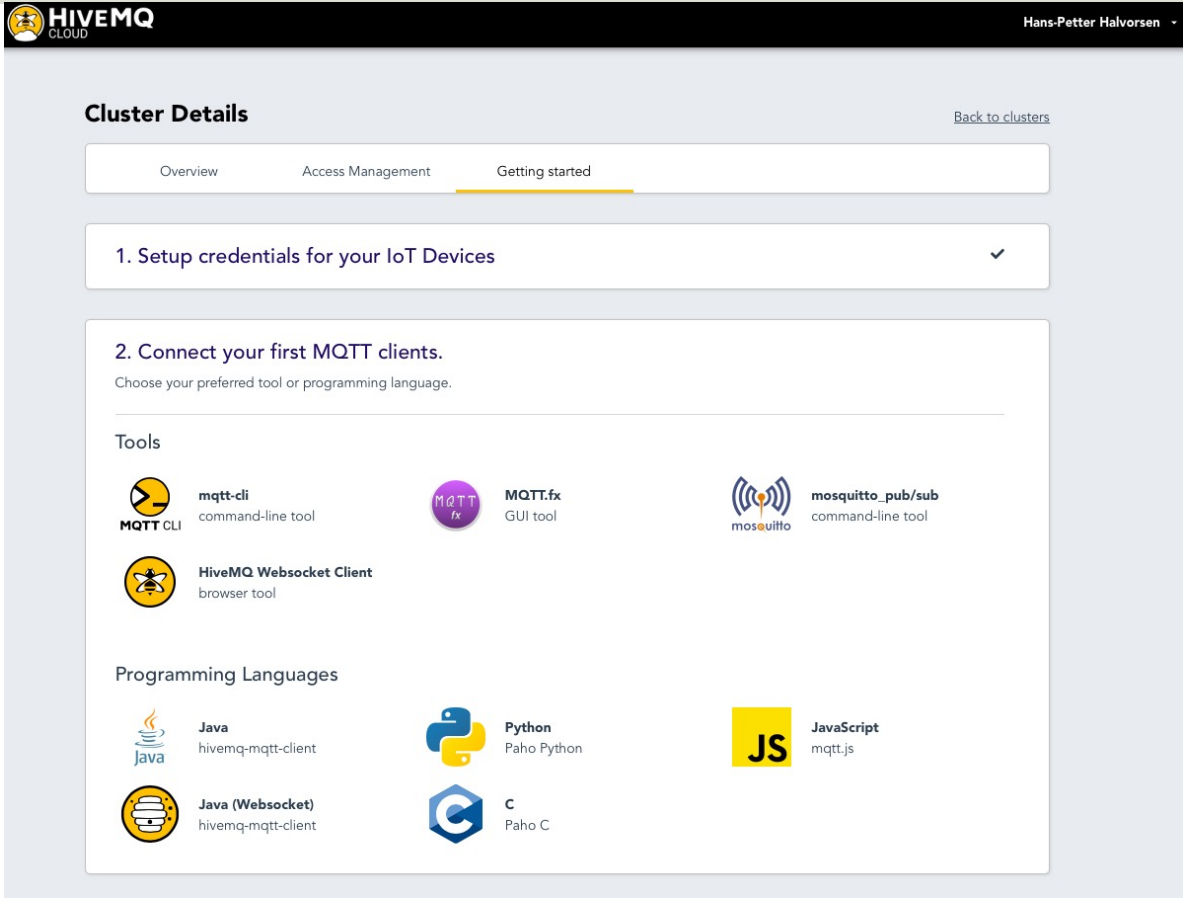
# HiveMQ Cloud

https://www.hivemq.com



# HiveMQ Cloud

https://www.hivemq.com




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://mqttx.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

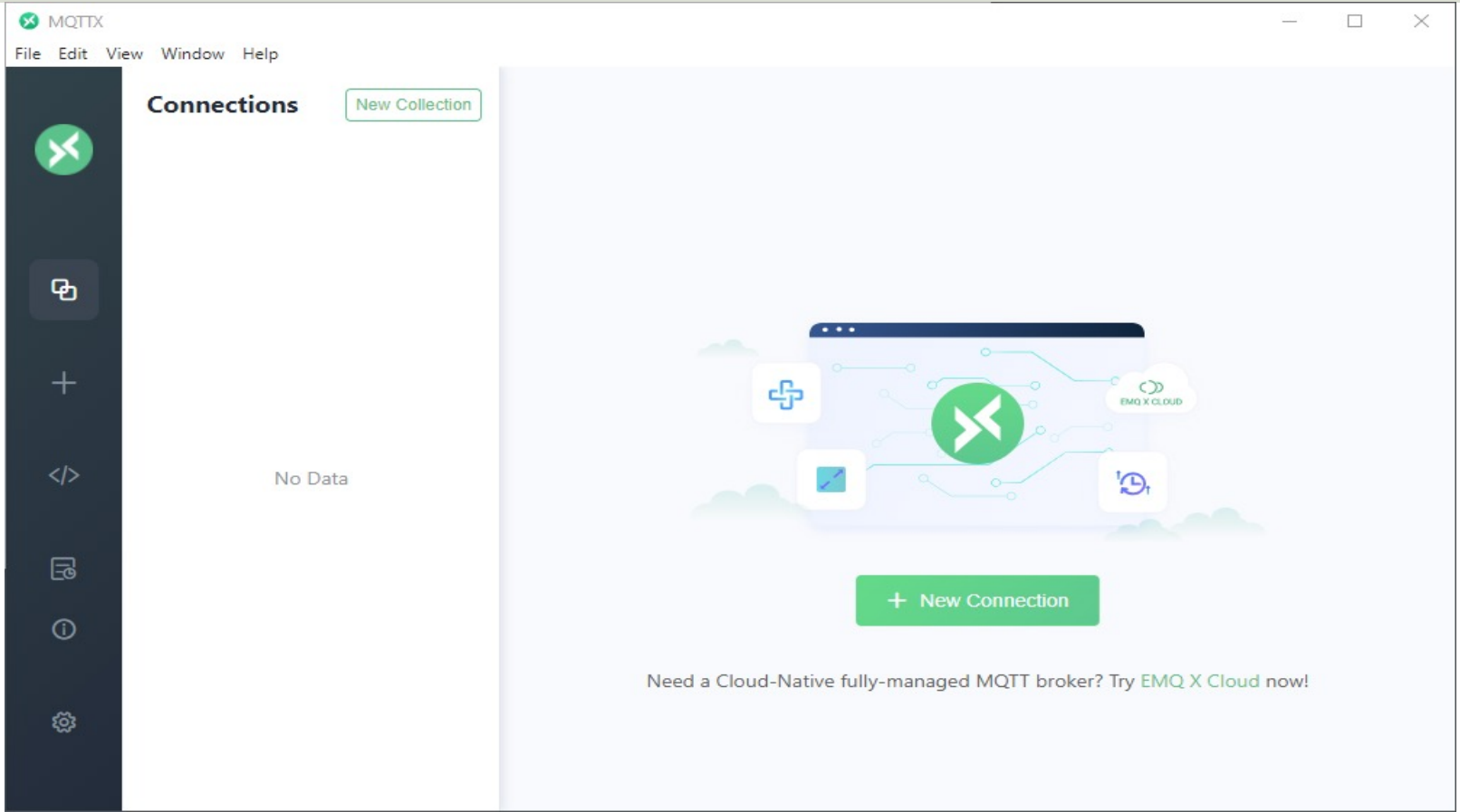
Hans-Petter Halvorsen

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://mqttx.app

# MQTTX


MQTTX
File
Edit
View
Window
Help
Connections
New Collection
No Data
EMQ X CLOUD
+ New Connection
Need a Cloud-Native fully-managed MQTT broker? Try EMQ X Cloud now!

# Connect to Broker HiveMQ Cloud using MQTTX Client

MQTTX

File Edit View Window Help

**Connections** New Collection

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

< Back **Edit** Connect

General

* Name

* Client ID .hivemq.cloud

* Host mqtts:// .hivemq.cloud

* Port 8883

Username

Password ••••••••••••••••••••

SSL/TLS true false

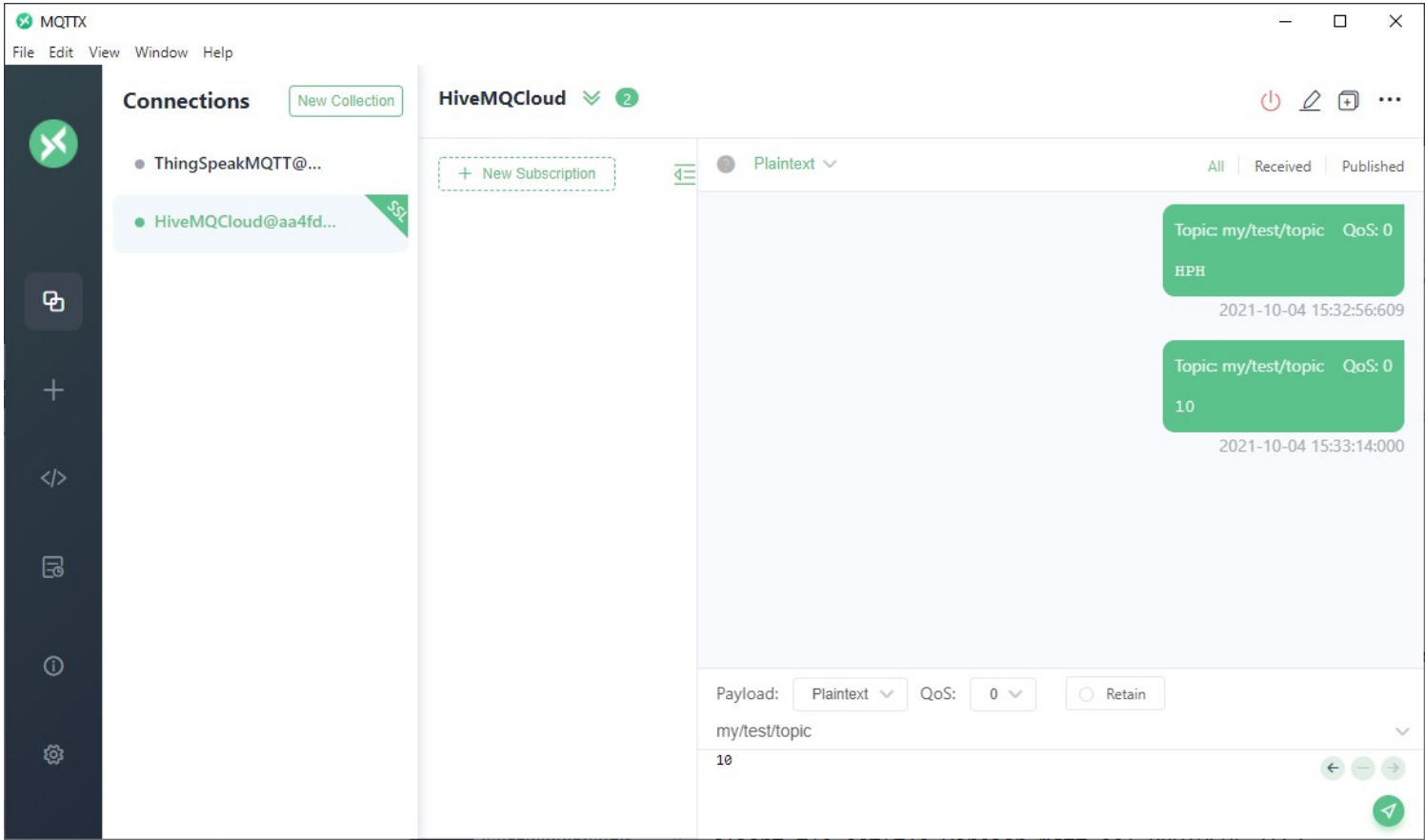
* Certificate CA signed server Self signed

SSL Secure

Advanced

Last Will and Testament

# Publish to Broker HiveMQ Cloud using MQTTX Client



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
gpiozero
isort

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

<INSTALL>
astroid
asttokens
bcrypt
bitstring
cffi
colorama
colorzero
cryptography
docutils
ecdsa
mypy-extensions
paramiko
parso

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 ... 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

https://www.halvorsen.blog

# HiveMQ Cloud and Python

Hans-Petter Halvorsen

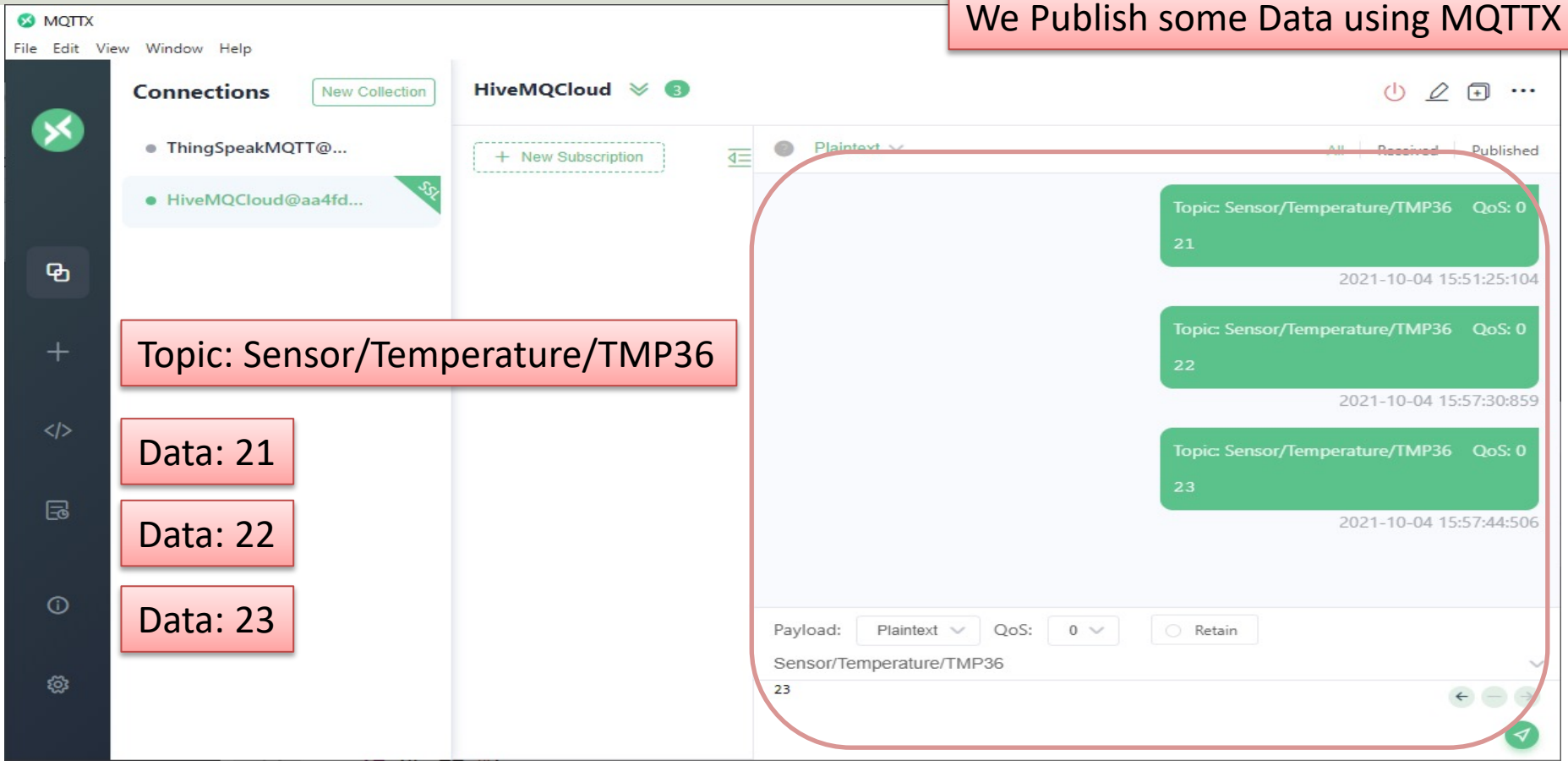
Table of Contents

# 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




Topic: Sensor/Temperature/TMP36

Data: 21

Data: 22

Data: 23

# Python Example

Thonny - /Users/halvorsen/OneDrive/Documents/Industrial IT and Automation/MQTT/MQTT Python/mqtt_hivemq_cloud_ex2.py @ 8 : 10

mqtt_hivemq_cloud_ex2.py

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

brokerAddress = "[redacted].hivemq.cloud"
userName = "[redacted]2"
passWord = "[redacted]5"

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

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

Python 3.7.9

# Publish – Subscribe Examples

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

Publish Temperature to HiveMQ Cloud.py

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

brokerAddress = "[redacted].hivemq.cloud"
userName = "[redacted]2"
passWord = "[redacted]"

topic = "Sensor/Temperature/TMP36"

min = 20
max = 30

# 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)

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

Shell

```
20
23
24
25
27
25
25
20
24
25
20
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...

File Edit View Run Tools Help

Subscribe on Topic in HiveMQ Cloud.py

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

brokerAddress = "[redacted]u.hivemq.cloud"
userName = "[redacted]"
passWord = "[redacted]"

topic = "Sensor/Temperature/TMP36"

# 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.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

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

brokerAddress = "xxxxxx"
userName = "xxxxxx"
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)
```

# Subscribe

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

brokerAddress = "xxxxxx"
userName = "xxxxxx"
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()
```

# 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

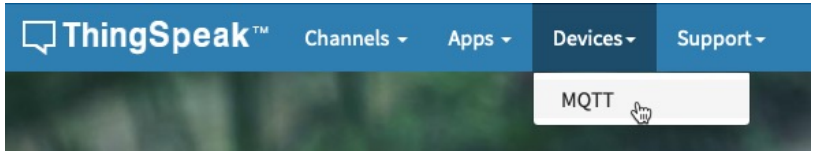
Table of Contents

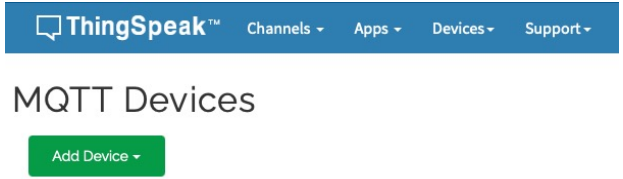
# MQTT ThingSpeak

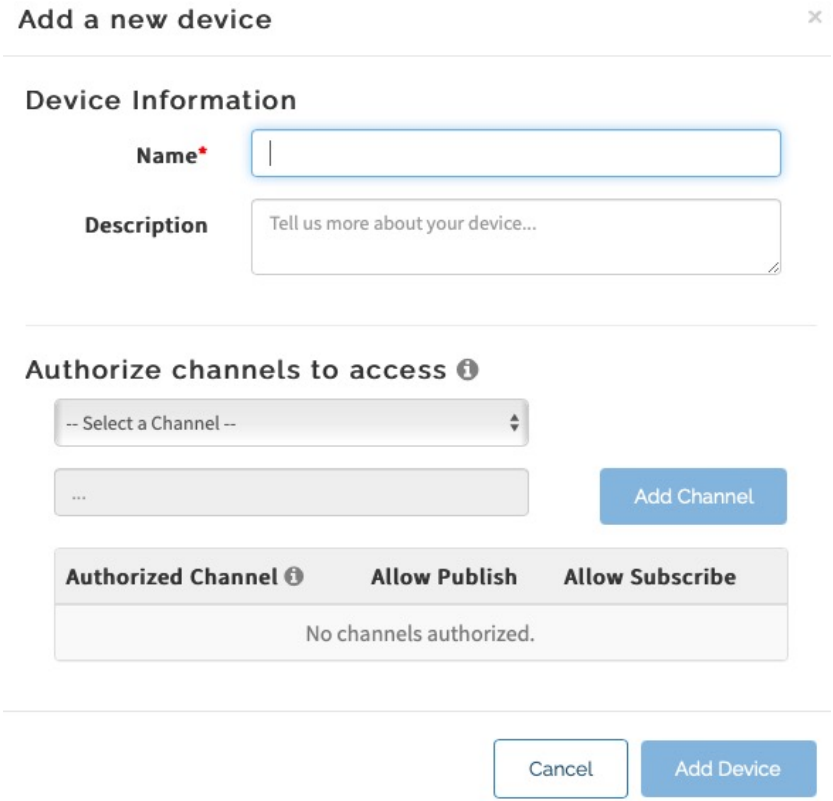
- https://mathworks.com/help/thingspeak/use-desktop-mqtt-client-to-publish-to-a-channel.html

# Configure MQTT in ThingSpeak

https://thingspeak.com



# ThingSpeak and MQTTX

MQTTX

File Edit View Window Help

Connections | New Collection

No Data

< Back | New | Connect

General

* Name

* Client ID

* Host: mqtt:// | mqtt3.thingspeak.com

* Port: 1883

Username

Password: ••••••••••••••••••••••

SSL/TLS: true | false

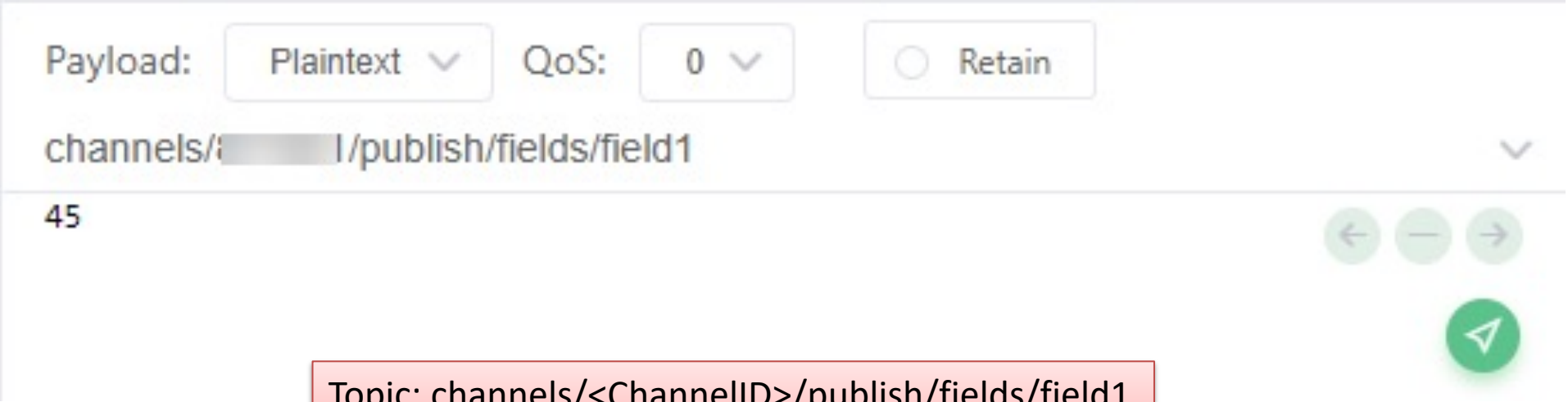
Advanced

Last Will and Testament

ThingSpeak MQTT Broker:

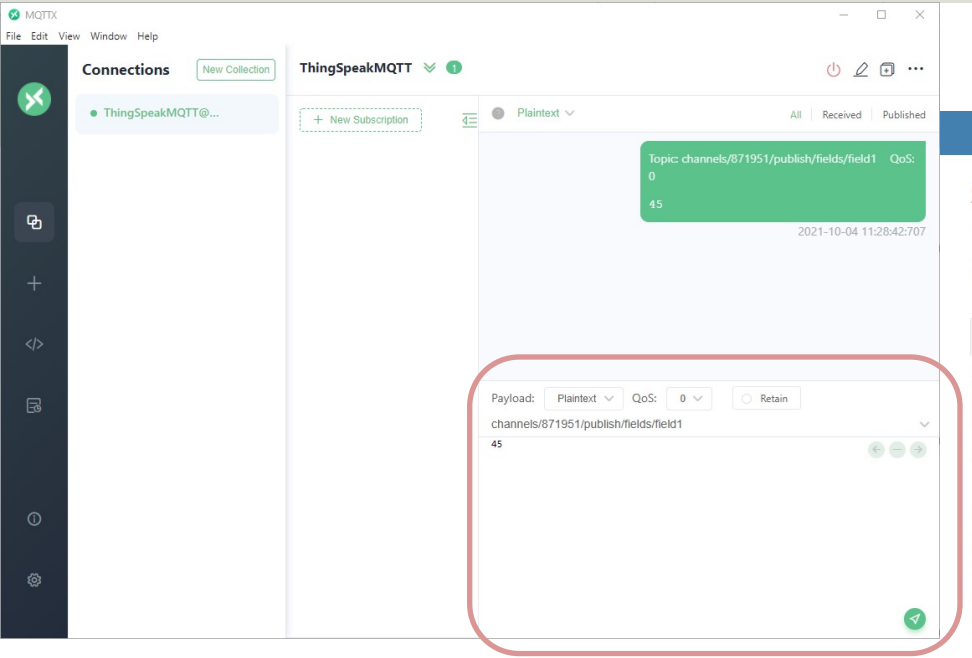
mqtt:// mqtt3.thingspeak.com

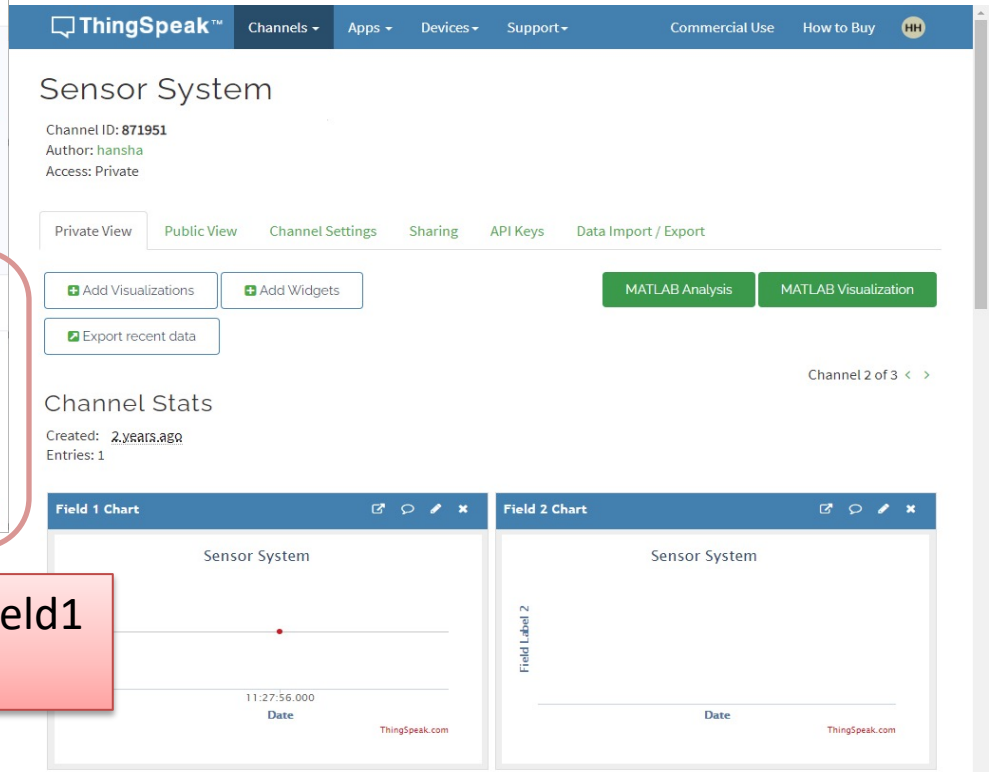
# **Publish** to Channel **Field**




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

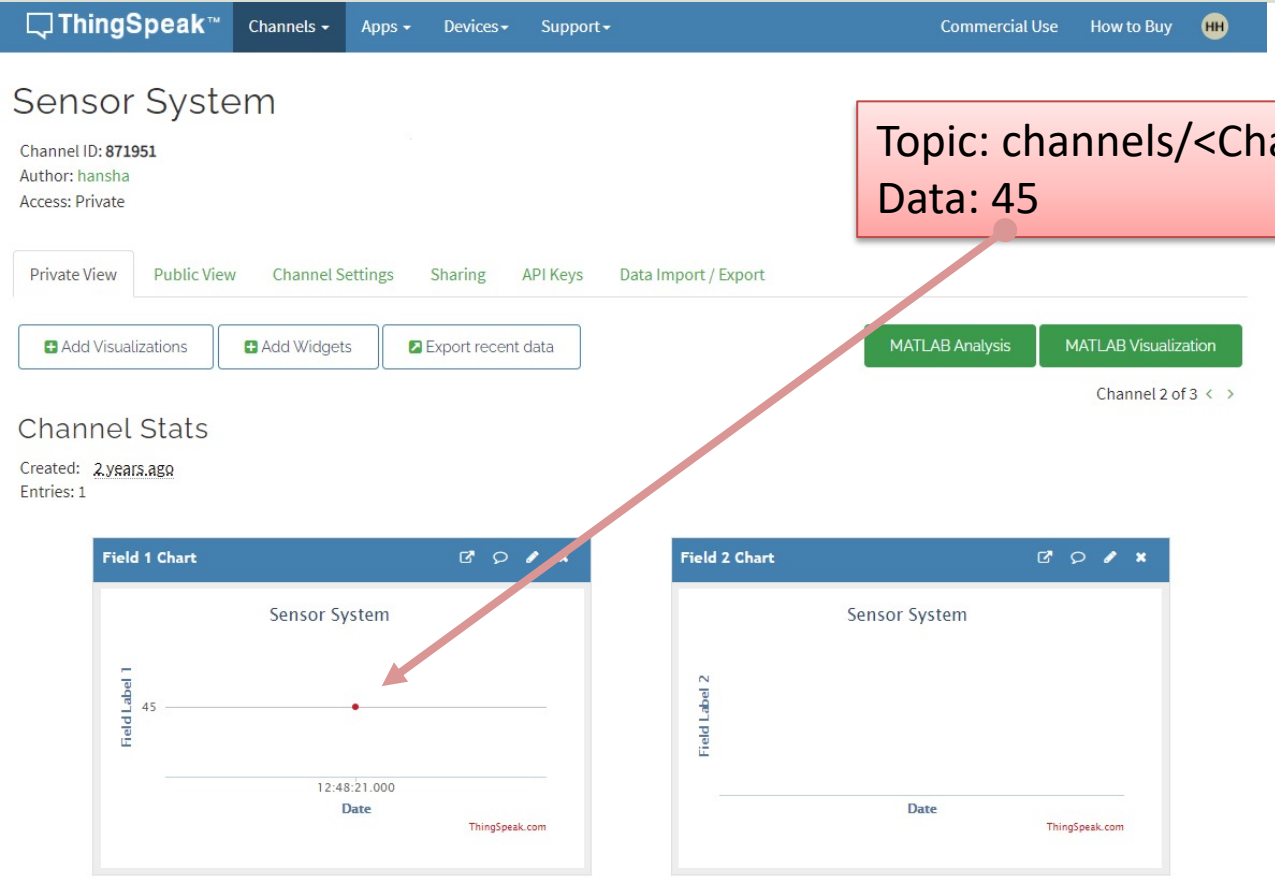
# Publish to Channel Field




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

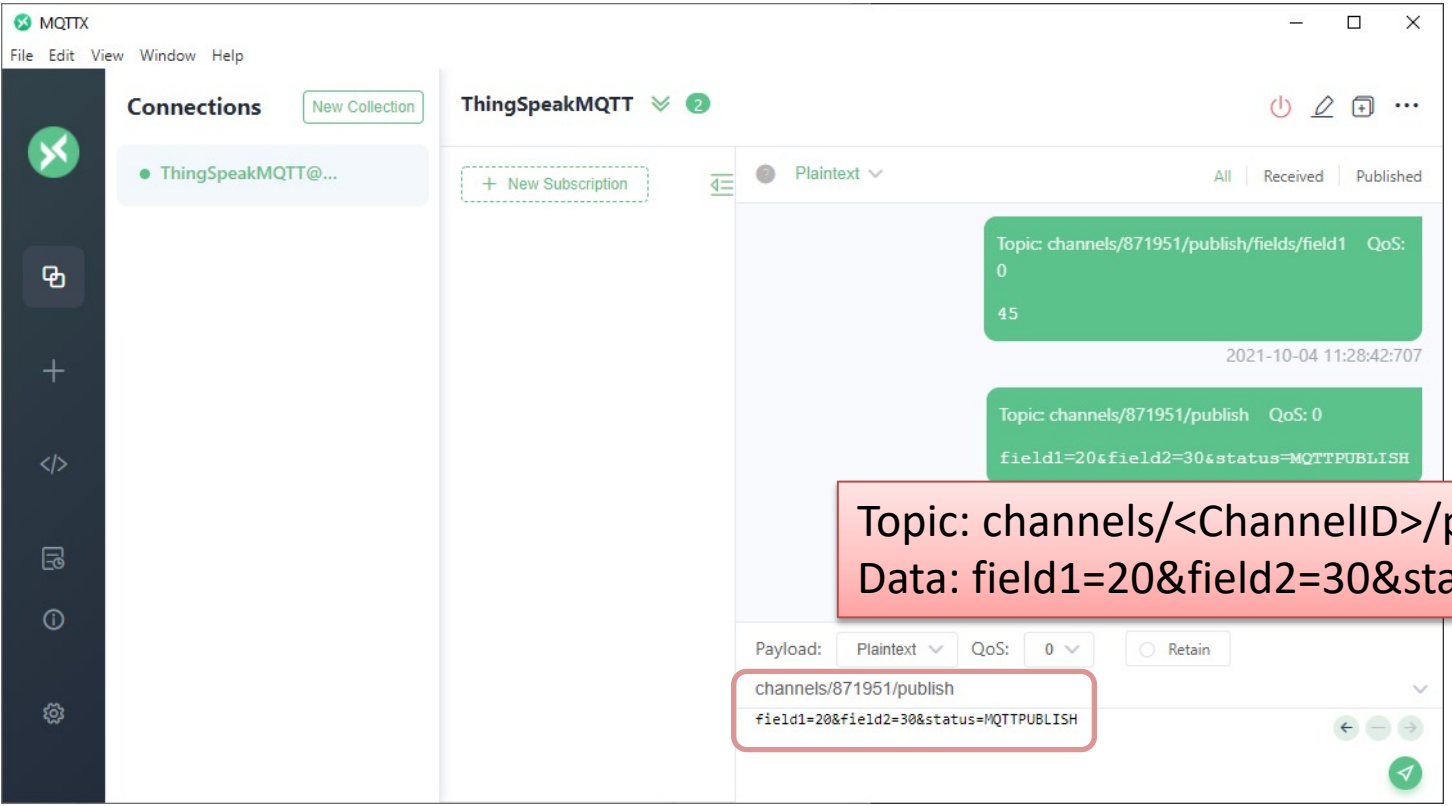
# Publish to Channel Field




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

# **Publish** to Channel **Feed**

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




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

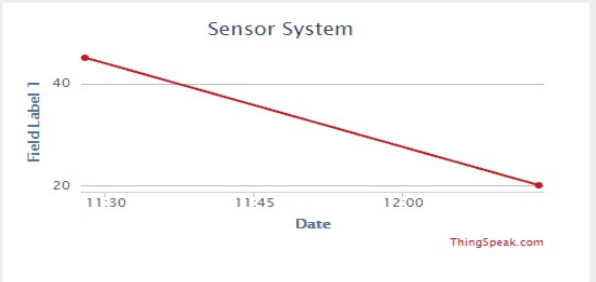
Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

Channel 2 of 3

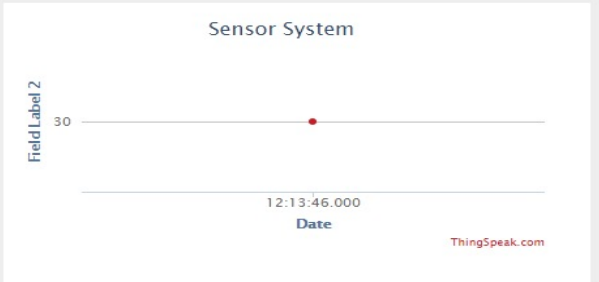
## Channel Stats

Created: 2 years ago
Entries: 2

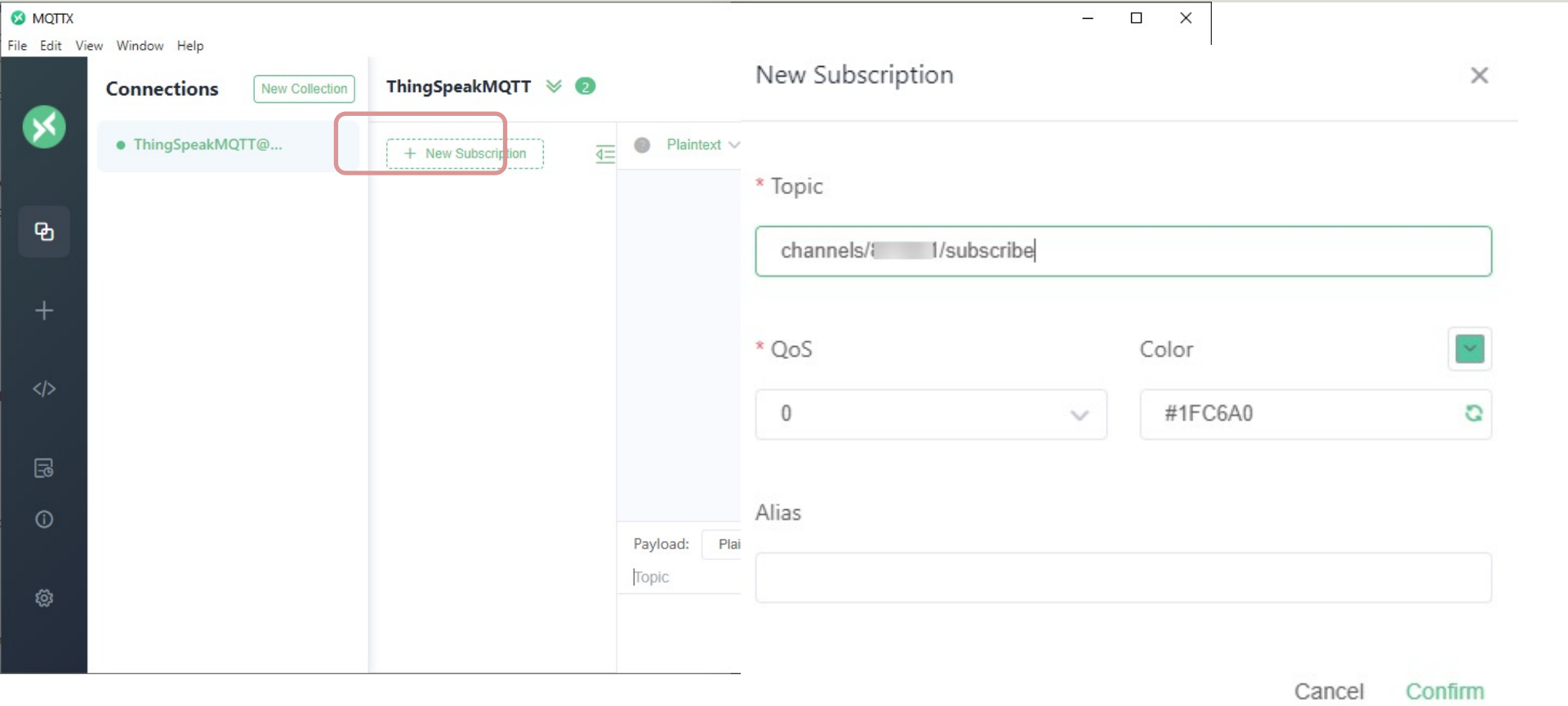
**Field 1 Chart**



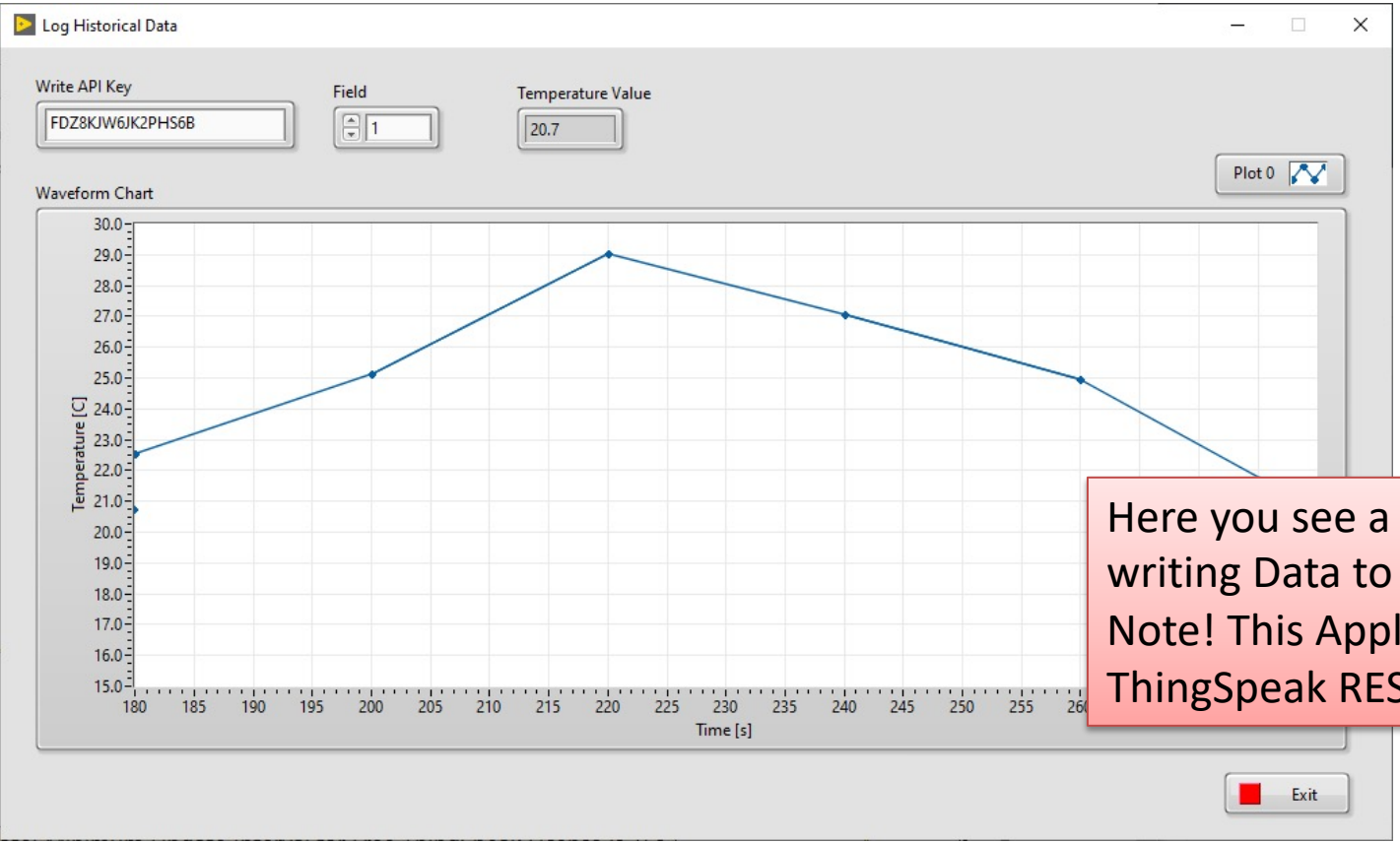

**Field 2 Chart**



# Subscribe to a Channel Feed



# **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**

MQTTX

File Edit View Window Help

**Connections** New Collection

ThingSpeakMQTT@...

**ThingSpeakMQTT** 3

+ New Subscription

channels/871951... QoS 0

Plaintext All Received Published

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

https://www.halvorsen.blog

# 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

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

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

min = 20
max = 30

# 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))

# 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
```

Shell

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

Python 3.7.9

# Publish

# Publish

```python
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

<untitled> | Subscribe on Topic in ThingSpeak.py

# Subscribe

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

brokerAddress = "mqtt3.thingspeak.com"
port = 1883
clientId = "[redacted]"
userName = "[redacted]"
passWord = "[redacted]A3D"
channelID = "[redacted]"
field = "field1"
topic = "channels/" + channelID + "/publish/fields/" + field

# 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(clientId)
client.on_connect = on_connect
client.on_message = on_message

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

client.subscribe(topic)
```

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
```

Python 3.7.9

```python
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\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

# Publish

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

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

min = 20
max = 30

# 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))

# 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
```

Shell

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

Python 3.7.9

Thonny - /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

# Subscribe

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

brokerAddress = "mqtt3.thingspeak.com"
port = 1883
clientId = "[redacted]"
userName = "[redacted]"
passWord = "[redacted]cA3D"
channelID = "[redacted]"
field = "field1"
topic = "channels/" + channelID + "/publish/fields/" + field

# 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(clientId)
client.on_connect = on_connect
client.on_message = on_message

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

client.subscribe(topic)
```

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
```

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