



Published in Python Point



Sign in to Medium with Google



M M Kamalraj

kamaljp@gmail.com

Continue as M M

You have **2** free member-only stories left this month. [Sign up for Medium and get an extra one](#)



Techletters

Follow

Aug 30, 2020 · 6 min read · ✨ · 🎧 Listen



Save



MQTT Beginners Guide

The IoT protocol explained with Python



Internet of Things (IoT) and the need for communication. MQTT for the win. [\(Source\)](#)

Why should you spend 5 minutes of your life reading this story? Because after reading this...

- ...you have a basic understanding of the MQTT protocol

- ...you know when to use MQTT
- ...you have the practical experience and can follow along the provided examples using the free *Mosquitto* MQTT broker and Python MQTT client library *Paho-MQTT*.

I recommend also following along with this article with my video on YouTube, which explains some more details behind the theoretical concepts and technical implementation.

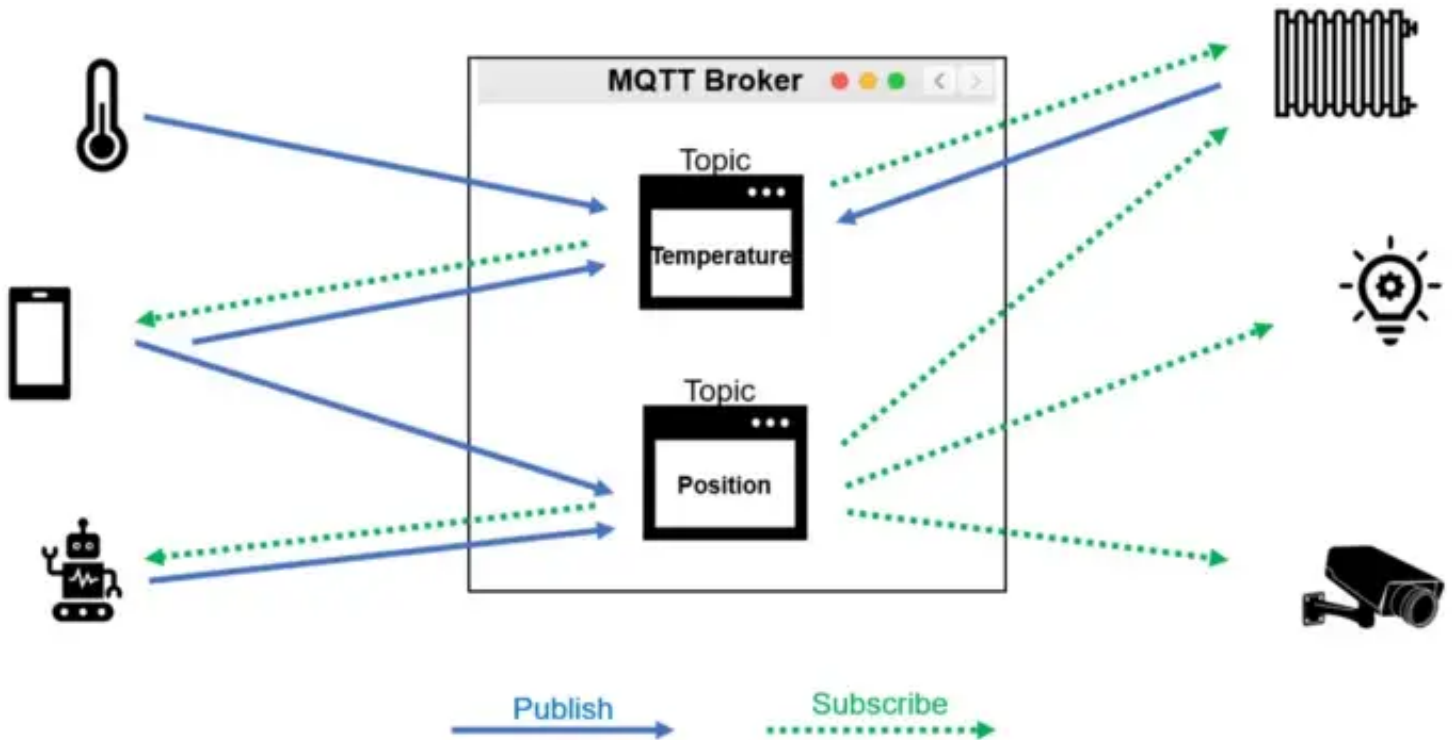
MQTT Basics

MQTT is not a new protocol. The first version was published back in 1999, but its popularity is dramatically increasing recently, especially in the Internet Of Things (IoT) context.

There are more and more devices, sensors, phones, embedded systems like connected cars or smart homes with the need to communicate with each other. With the rise of the IoT revolution, MQTT is experiencing a second spring.

MQTT is an open and simple client server publish/subscribe message transport protocol designed for machine-to-machine communication between different devices in environments of

high latency and low network bandwidth. Alright — what the hell does this mean? Let's break it down after a short overview picture.



MQTT broker and topics connecting IoT devices. (Source: self-made)

- **Machine-to-machine communication**

This one is quite easy. One system needs to exchange information with other systems. In the IoT context, the need for communication between devices increases dramatically.

Everything is connected — in a smart home, your fridge can talk to your vacuum cleaner. You can switch on the lights with your smartphone and order new dog food with your voice over smart speakers.

All this communication needs to be enabled, structured, reliable, and secure. For this matter communication standards are defined by protocols like HTTP, MQTT, or others.

- **Publish/subscribe messaging transport protocol**

MQTT is a protocol with a special publish/subscribe implementation. Devices are not directly talking to each other. Instead, communication is structured into topics and handled over a central server (broker).

Devices can publish information to one or more topics. At the same time, they can subscribe to one or more topics to receive the information published to these topics. This allows one-to-many message distribution and decoupling of applications.

→ **Example 1:** Take a look at the picture above. The smartphone can publish its current position on the *Position* topic. Interested devices can subscribe to this topic and will get informed whenever a new message is published.

If the smartphone position is located outside your flat, the lightbulbs and heating devices can turn off while the security camera and the vacuum cleaner can switch on. The smartphone does not need to inform all the devices separately. It can simply publish the information to the central topic and whatever device is interested in the information can subscribe to this.

→ **Example 2:** The outside temperature sensor can publish the outside temperature to the *Temperature* topic and your heating devices can subscribe to the same. Whenever the outside temperature is below a defined amount, the heating devices can turn on.

At the same time, the smartphone subscribes to the topic to get informed about the current indoor and outdoor temperature and inform its human about the same. The smartphone can additionally publish temperature control messages to the topic to regulate the heating device which also acts as a subscriber to the topic.

- **High Latency and low network bandwidth**

The MQTT protocol is lightweight, efficient, and contains only a small footprint. This makes it a perfect match for IoT devices and scenarios which are often running within unstable environments and using unwired connections like WiFi, Bluetooth, or satellite.

Now enough of reading. Let's jump into an example implementation using *Mosquitto* MQTT broker and Python MQTT client library *Paho-MQTT*.

Example implementation with Mosquitto MQTT and Python

Preparation

You can install an MQTT broker locally or use one of the below online MQTT brokers for free:

- HiveMQ: broker.hivemq.com

- Eclipse Mosquitto: test.mosquitto.org
- Eclipse IoT: iot.eclipse.org

In this example, I am using the free online MQTT broker Mosquitto (test.mosquitto.org) which you can easily connect to (publish and/or subscribe) using an MQTT client. As a client, we are using Python **Paho-MQTT** which can be installed with cmd command *pip install paho-mqtt*

Publishing to a Topic

We will use the example from the picture above and write two Python applications that publish information to the **Temperature** topic. The first application will publish the inside temperature on this topic.

```
1  import paho.mqtt.client as mqtt
2  from random import randrange, uniform
3  import time
4
5  mqttBroker = "mqtt.eclipseprojects.io"
6
7  client = mqtt.Client("Temperature_Inside")
8  client.connect(mqttBroker)
9
10 while True:
11     randNumber = uniform(20.0, 21.0)
12     client.publish("TEMPERATURE", randNumber)
13     print("Just published " + str(randNumber) + " to topic TEMPERATURE")
14     time.sleep(1)
```

MQTT_publisher.py hosted with ♥ by GitHub

[view raw](#)

- First, we are importing the Python MQTT client paho-mqtt (line 1) and two helper libraries random & time (*lines 2–3*) which are only necessary to construct this example
- we need to define the location of the MQTT Broker (*line 5*). I am using the free online broker Eclipse Mosquitto, but you can use any

- We are setting up a new MQTT client (*line 7*) and tell the client to connect to the broker (*line 8*) which was specified before
- At last, we are setting up a while loop (*lines 10–14*) which will publish the current inside temperature to the Temperature topic each second (*line 14*). The inside temperature will be a random float number between 20 and 21 (*line 11*) which is published to the Temperature topic (*line 12*) and then printed to the console (*line 13*).

If you run the script in your console, it will look like this.

```
Just published 20.152963374957682 to topic TEMPERATURE
Just published 20.161864524496735 to topic TEMPERATURE
Just published 20.832600714734365 to topic TEMPERATURE
Just published 20.139163091102564 to topic TEMPERATURE
Just published 20.596117184032405 to topic TEMPERATURE
Just published 20.050333160761117 to topic TEMPERATURE
Just published 20.25035838170374 to topic TEMPERATURE
Just published 20.76368639769279 to topic TEMPERATURE
Just published 20.532681815720827 to topic TEMPERATURE
```

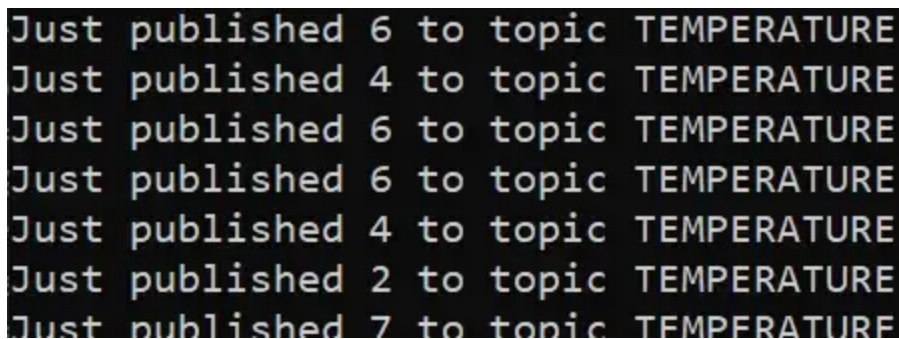
We do exactly the same again — this time with the outside temperature. I think no further explanation is needed, only *line 11* differs from the coding above which will set the temperature to a random int number between 0 and 10. Additionally, the client gets a different name (*line 7*).

```
1 import paho.mqtt.client as mqtt
2 from random import randrange, uniform
3 import time
4
5 mqttBroker = "mqtt.eclipseprojects.io"
6
7 client = mqtt.Client("Temperature_Inside")
8 client.connect(mqttBroker)
9
10 while True:
11     randNumber = uniform(20.0, 21.0)
12     client.publish("TEMPERATURE", randNumber)
13     print("Just published " + str(randNumber) + " to topic TEMPERATURE")
14     time.sleep(1)
```

MQTT_publisher.py hosted with ♥ by GitHub

[view raw](#)

In the console, it will look like this.



```
Just published 6 to topic TEMPERATURE
Just published 4 to topic TEMPERATURE
Just published 6 to topic TEMPERATURE
Just published 6 to topic TEMPERATURE
Just published 4 to topic TEMPERATURE
Just published 2 to topic TEMPERATURE
Just published 7 to topic TEMPERATURE
```

Subscribing to a Topic

Now its time to consume the messages which are published to the Temperature topic. Let's see how we can set up a subscriber with Python Paho-MQTT.

```
1 import paho.mqtt.client as mqtt
2 import time
3
4 def on_message(client, userdata, message):
5     print("received message: " ,str(message.payload.decode("utf-8")))
6
7 mqttBroker = "mqtt.eclipseprojects.io"
8
9 client = mqtt.Client("Smartphone")
10 client.connect(mqttBroker)
11
12 client.loop_start()
13
14 client.subscribe("TEMPERATURE")
15 client.on_message=on_message
16
17 time.sleep(30)
18 client.loop_stop()
```

MQTT_subscribe.py hosted with ♥ by [GitHub](#)

[view raw](#)

Some steps are quite similar compared to the implementation of the publishing clients...

- Again we need to import Python MQTT client paho-mqtt first(*line 1*). Additionally, we are importing time as a helper library (*line 2*).
- we need to define the location of the MQTT Broker (*line 7*) and set up a new MQTT client (*line 9*) which connects to the broker (*line 10*)

... but the following steps are specific to the subscribing MQTT clients

- the command `client.subscribe` (*line 14*) defines the Topic to connect to and already subscribes the client to the broker's topic. Actually we will already receive published messages, but we will not see any yet

- Whenever the subscribed client receives messages it generates an **on_message** callback. To view the received messages we need to specify the on_message callback function (*lines 7–8*) and start it within a loop (*lines 12–18*).

In the console, it will look like this.

```
received message: 4
received message: 4
received message: 20.011309541318152
received message: 0
received message: 20.169592869618793
received message: 1
received message: 20.013313198960883
received message: 9
received message: 20.380534030271615
received message: 8
received message: 20.67437407488137
```

Summary

MQTT is an easy to implement & lightweight protocol with a small footprint. The publish/subscribe implementation makes it a perfect match for IoT scenarios in which many devices are interested in information from many other devices.

You can follow along with the above examples or simply [download my code from GitHub](#). I hope you liked my short introduction. If you want to learn more about MQTT, HTTP or a comparison between the protocols just let me know :-)

If you want to follow along with all my stories & **support me**, you can [register on Medium](#). If something is unclear or you need help, just drop a comment. I will answer it for sure.

code-and-dogs/mqtt-python

Contribute to code-and-dogs/mqtt-python development by creating an account on GitHub.

github.com

Sign up for Python Point Newslette

 190

|

 5

By Python Point

Best recent stories recently published in Python Point. [Take a look.](#)

Your email



We couldn't process your request. Try again, or contact our support team.

Open in app ↗

Sign up

Sign In



 Search Medium



Get the Medium app

