Internet Of Things

Basics of HTTP and ThingSpeak

Prepared by:

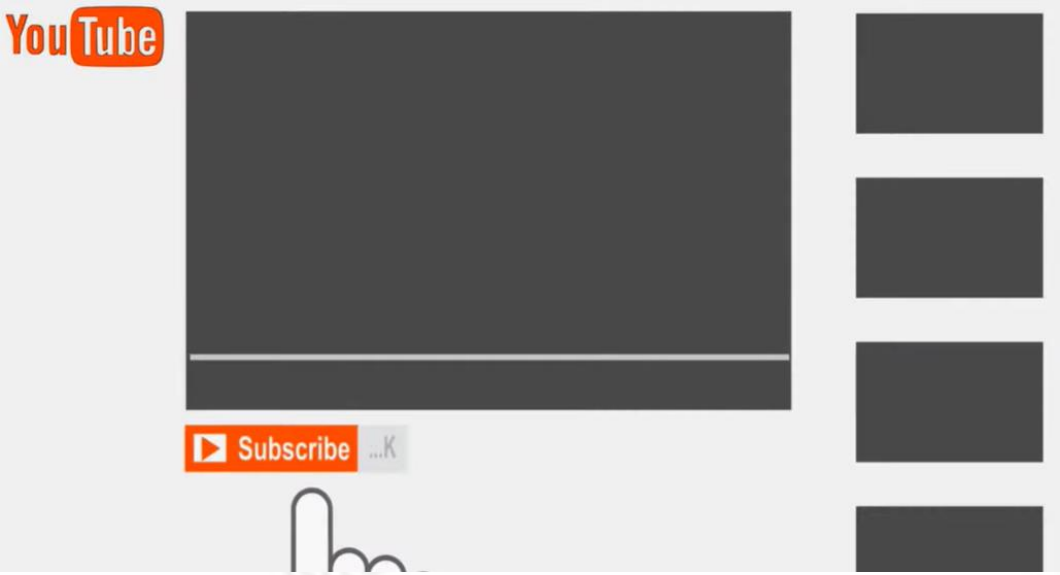
Dr. Murad Yaghi
Eng. Malek Al-Louzi

School of Computing and Informatics – Al Hussein Technical University

Fall 2024/2025

Introduction

- Let's try to understand MQTT protocol with the help of simple example.
- We will take a YouTube as an example
- YouTube working is similar to MQTT protocol.



Introduction

- There are more than 100 million YouTube channels available.
- If you are subscribed to a specific YouTube channel, whenever this channel publishes a video, you would get a notification on your YouTube home page.
- So, YouTube works on **publish** and **subscribe** mechanism.

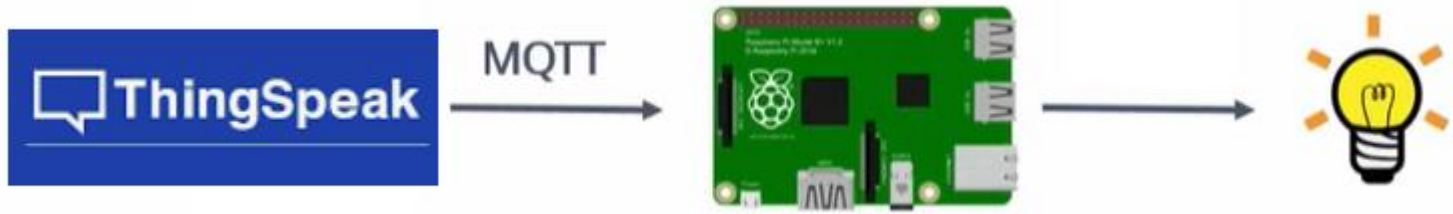
Introduction

- **MQTT stand for (Message Queuing Telemetry Transport)**
- It is a lightweight, **publish-subscribe** messaging protocol designed for constrained devices and low-bandwidth, where you can publish and receive messages.

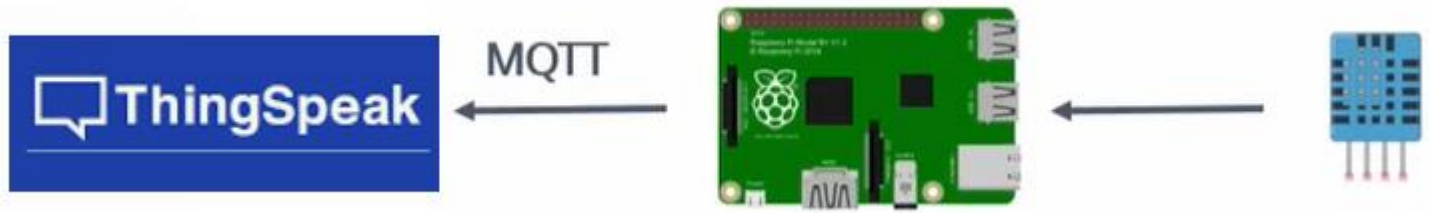
- It is widely used in **Internet of Things (IoT)** applications because of its simplicity and efficiency.

Introduction

- For example, MQTT allows you to:
  - Send a command from ThingSpeak to control an LED



  - Publish a sensor data



MQTT Concepts

- Publish/Subscribe
- Messages
- Topics
- Broker

Publish/Subscribe

- Means that a device can publish messages to other devices, or your device can subscribe to a particular topic to receive messages

Publish/Subscribe

- For example:
  - 1

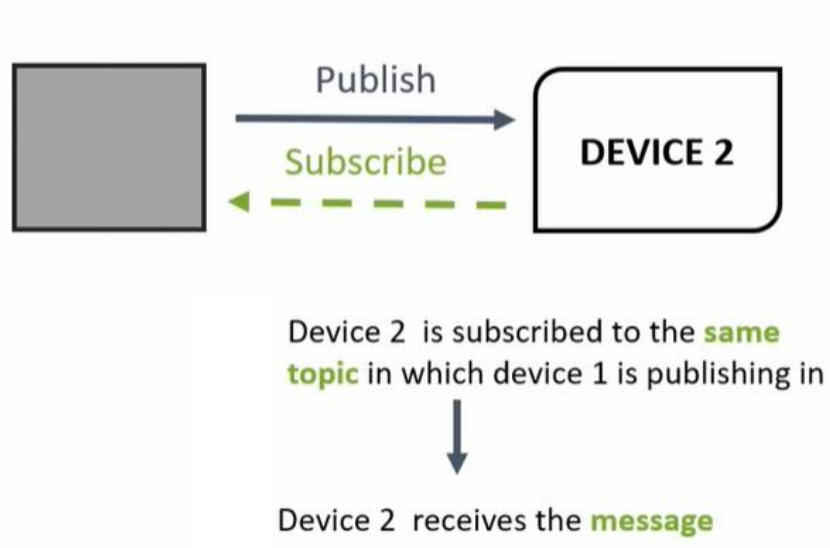
Device 1 publishes on a **topic**

  - 2

Device 2 is subscribed to the **same topic** in which device 1 is publishing in

Publish/Subscribe

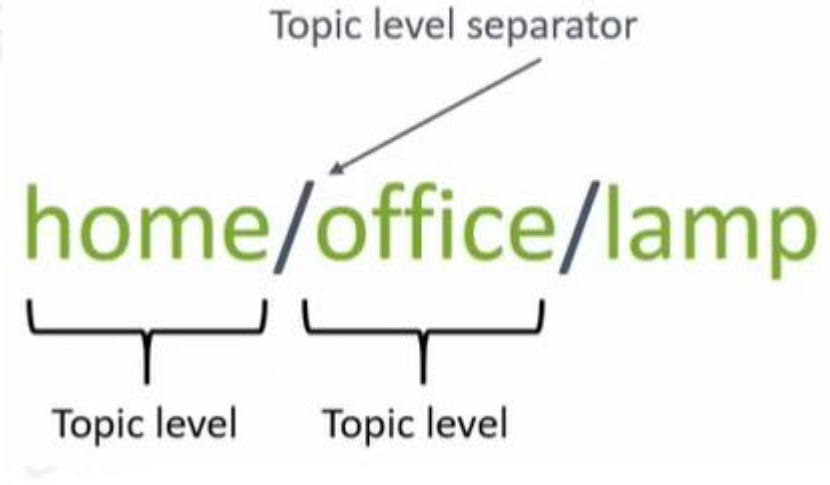
- 3



- **Message** or **Payload**: information exchanged between devices – **Command** or **Data**

Topics

- Topics are the way to register interest for incoming messages or how you specify where you want to publish your messages
- Topics are represented with **strings** separated by **slashes "/"**
- Slashes indicate the **topic level**
- Example:

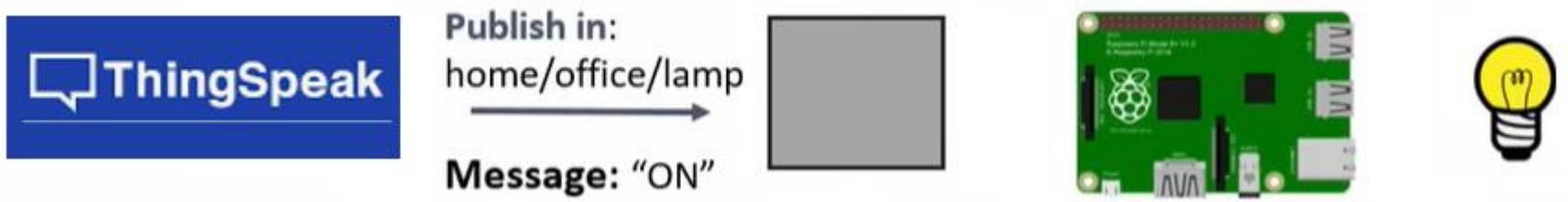


Topics

- For example, if we want to turn On an LED connected to a Raspberry Pi from ThingSpeak that represents a lamp in your home office, we will follow the following steps:
- 1

Topics

- 2


ThingSpeak
Publish in:
home/office/lamp
Message: "ON"

- 3


ThingSpeak
Publish in:
home/office/lamp
Message: "ON"

- 4


ThingSpeak
Lamp is ON

Broker

- Broker responsible to receive all messages
- Filtering the messages
- Publishes the messages to all subscribed clients

Example 1

- To enable **two Raspberry Pi devices** communicate using MQTT with the help of a **public MQTT broker**, you can follow these steps

Example 1

- **Steps overview:**
- Set up MQTT on both Raspberry Pis using the **paho-mqtt** library
- Use a public MQTT broker (**broker.hivemq.com**) or any other public broker
- One Raspberry Pi will act as a **publisher** (sending data)
- The other Raspberry Pi will act as a **subscriber** (receiving data)

Example 1

- Install **paho-mqtt** on Both Raspberry Pis
- On both Raspberry Pi devices, install the paho-mqtt which is a library to implement mqtt clients

```
pip install paho-mqtt
```

Example 1

- Write Python Code for Publisher (Sending Data)

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

# Public MQTT Broker (Mosquitto Test Broker)
MQTT_BROKER = "broker.hivemq.com"
MQTT_PORT = 1883
MQTT_TOPIC = "test/topic"  # Topic to publish

# Callback when message is published
def on_publish(client, userdata, mid):
    print(f"Message Published: {mid}")

# Setup MQTT client
client = mqtt.Client()
client.on_publish = on_publish
client.connect(MQTT_BROKER, MQTT_PORT, 60)

# Start the loop
client.loop_start()

# Publish data
while True:
    payload = "Hello from Raspberry Pi!"
    print(f"Publishing: {payload}")
    client.publish(MQTT_TOPIC, payload)
    time.sleep(5)  # Publish every 5 seconds
```

Example 1

- Write Python Code for Subscriber (Receiving Data)

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

# Public MQTT Broker (Mosquitto Test Broker)
MQTT_BROKER = "broker.hivemq.com"
MQTT_PORT = 1883
MQTT_TOPIC = "test/topic"  # Same topic as the publisher

# Callback when connected to the broker
def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT Broker")
    client.subscribe(MQTT_TOPIC)  # Subscribe to the topic

# Callback when a message is received
def on_message(client, userdata, msg):
    print(f"Received message: {msg.payload.decode()}")

# Setup MQTT client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

# Connect to the broker
client.connect(MQTT_BROKER, MQTT_PORT, 60)

# Start the loop to listen for messages
client.loop_forever()
```

Example 2

- Two Raspberry Pis, each one connected directly to a push button and LED. The push button of the first Raspberry Pi will control (ON/OFF) the LED on the second Raspberry Pi and vice versa.

Example 2

```python
import RPi.GPIO as GPIO
import paho.mqtt.client as mqtt
import time

BROKER = "broker.hivemq.com"
PORT = 1883
TOPIC = "htu/iot/sec2"
BUTTON_PIN = 17
LED_PIN = 27

GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.output(LED_PIN, GPIO.LOW)

client = mqtt.Client()

def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT broker")
    client.subscribe(TOPIC)

def on_message(client, userdata, msg):
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(LED_PIN, GPIO.LOW)

client.on_connect = on_connect
client.on_message = on_message
client.connect(BROKER, PORT, 60)
client.loop_start()

while True:
    if GPIO.input(BUTTON_PIN) == GPIO.HIGH:
        print("Button pressed!")
        client.publish(TOPIC, "Button Pressed")
        time.sleep(0.5)
```

SERVO MOTOR SG90

- The SG90 servo motor is a small, lightweight, and cost-effective servo motor commonly used in robotics
- Operating Voltage: 5V
- Rotation Range: 180 degrees
- Controlled via Pulse Width Modulation

(PWM) at 50 Hz

- Active Duty Cycle range: 2% to 12%

SERVO MOTOR SG90

- **Pinout:**

SERVO MOTOR SG90

- **Example:**

```python
# We imports the GPIO module
import RPi.GPIO as GPIO
import time

# We name all the pins on BOARD mode
GPIO.setmode(GPIO.BOARD)

# Set pin 16 as an output for PWM
GPIO.setup(16, GPIO.OUT)

# Set up the PWM on pin #16 at 50Hz
pwm = GPIO.PWM(16, 50)

pwm.start(2) # Start the servo with 0 duty cycle ( at 0 deg position )
time.sleep(0.5) # Tells the servo to Delay for 0.5sec

# rotating to 180 degrees in 10 steps
duty = 2
while duty<=12:
    pwm.ChangeDutyCycle(duty)
    time.sleep(0.5)
    duty = duty + 1

time.sleep(2) # Tells the servo to Delay for 2sec
pwm.ChangeDutyCycle(7) # Tells the servo to turn back to 90 degrees

time.sleep(2) # Tells the servo to Delay for 2sec
pwm.stop() # Stop the servo with 0 duty cycle ( at 0 deg position )
```

Any Questions???