Introduction

How do you send data from an ESP32 to a Laptop?

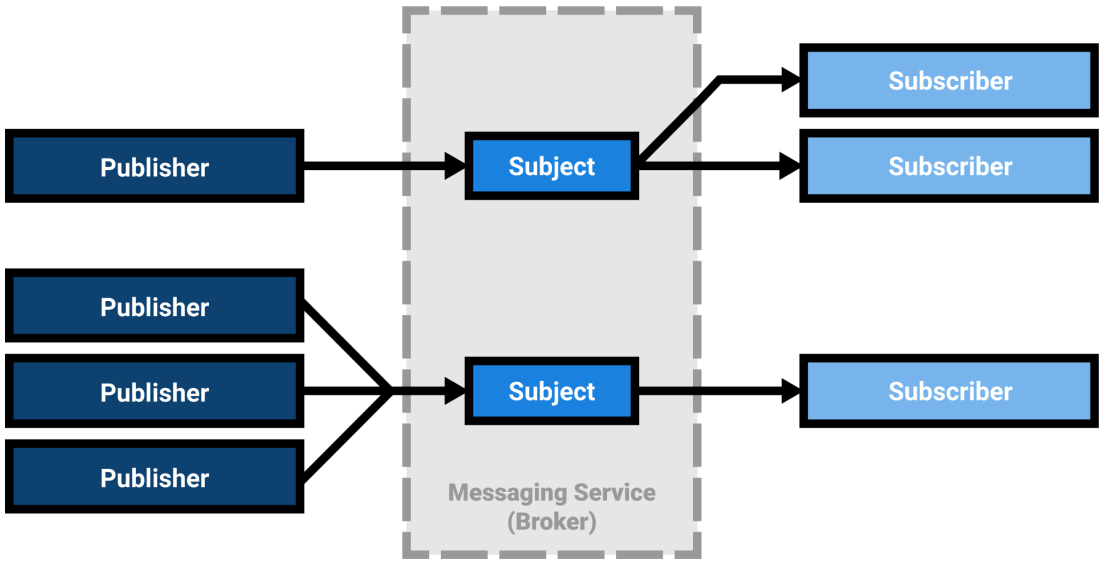
There are several ways to establish a communication between the two. For instance, most of the time, a wired connection in the form of USB (Universal Serial Bus) is used to be able to upload code from the Arduino software in our laptop to our microcontrollers, and the same wired connection can also be used to send data from our microcontroller to the Serial Monitor of the Arduino in our laptop.

But, what if we want our microcontrollers to be wirelessly separated from our Laptops?

Say we want to use our ESP32 microcontroller with an IMU sensor as a gaming input device like a hand-held tennis racket that hits an incoming ball from a tennis game implemented in a laptop. There is the need to send data from the ESP32 and IMU to the laptop to be able to know if the player's swing orientation correctly hits the ball on the laptop's screen, but how would we send the data from our microcontroller to the game implemented in our laptop?

One answer is over wireless communication, and this can be easily done by a messaging protocol called MQTT which stands for Message Queuing Telemetry Transport. MQTT is a standard messaging protocol of IoT (Internet of Things) devices (MQTT.org, 2022). The said protocol is utilized in machine-to-machine communication using a simple publish/subscribe communication pattern (Bernstein et al., 2021).

Publish/Subscribe Communication Pattern of MQTT



Rothuis, A. (2018). https://www.arothuis.nl/posts/messaging-pub-sub/

In the above diagram, it can be seen that there are publishers sending subjects, also known as topics, into a broker. After being sent to the broker, these topics are sent to the subscribers of those topics, respectively. This is a high-level description of how MQTT works. The major component of MQTT is the broker which allows the publisher and subscriber to communicate to each other. The MQTT broker takes the messages published by the publisher, filters these messages according to their subjects/topics, and then proceeds to send them to the subscribers (Chaurasia, 2020).

Tutorial

In this tutorial, we will enable an ESP32 microcontroller to send data to a laptop using MQTT. The data that will be sent for this tutorial is a simple integer, but these data can easily be replaced by sensor or telemetry values.

There are two MQTT broker that can be used in these tutorial. They are both open-source – test.mosquitto.org or mqtt.eclipseprojects.io.

At the end of this tutorial, one must be able to establish an MQTT connection between an ESP32 microcontroller and a Laptop.

Materials Needed (Hardware and Software)

1. A Laptop
2. ESP32 Microcontroller
3. Python3 Installed in your Laptop
4. Arduino IDE
5. WiFi with known SSID and Password (not 5G)
6. Directory in your Laptop where you want to store the contents of this tutorial call it MQTT_Basic

Setting up the Laptop

1. Install and download Miniconda in accordance with your Operating System.
2. Create an environment in your Laptop and install the MQTT library for Python. Execute the following commands:

```
$ cd [path to MQTT_Basic directory]
$ conda update conda
```

```
$ conda create -n yourenvname python
$ conda activate yourenvname
$ pip install paho-mqtt
$ conda deactivate
```

Setting up the Arduino IDE

1. Add ESP32 boards in your Arduino. Follow this tutorial.
2. Install the PubSubClient by Nick O'Leary

Codes for MQTT Communication between ESP32 and Laptop

In the codes below, make sure that

1. The WiFi credentials written in the Arduino code is the same as the WiFi that the Laptop is connected to.

```
// WiFi
const char *ssid = ""; // Enter your WiFi name
const char *password = "";  // Enter WiFi password
```

2. The MQTT broker should be the same.
3. The topic by the Publisher must match the topic of the Subscriber

    a. Arduino Code (ESP32 Microcontroller)

    ```
    const char *topic = "test/topic"; // Enter your topic name
    here
    ```

    b. Python Code (Laptop)

    ```
    client.subscribe("ece180d/test", qos=1)
    ```

MQTT Code for ESP32 as a Publisher (Arduino)

```
// Most code is provided to us in the Lab 4 Prompt

#include <WiFi.h>
#include <PubSubClient.h>
#include "arduino_secrets.h"
```

```
// WiFi
const char *ssid = ""; // Enter your WiFi name
const char *password = "";  // Enter WiFi password

// MQTT Broker
// const char *mqtt_broker = "broker.emqx.io";
// const char *topic = "ece180d/test";
// const char *mqtt_username = "emqx";
// const char *mqtt_password = "public";
// const int mqtt_port = 1883;

// MQTT Broker
const char *mqtt_broker = "mqtt.eclipseprojects.io";
const char *topic = "test/topic"; // Enter your topic name here
// const char *mqtt_username = "emqx";
// const char *mqtt_password = "public";
const int mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
// Set software serial baud to 115200;
Serial.begin(115200);
// connecting to a WiFi network
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
}
Serial.println("Connected to the WiFi network");
// Connecting to a MQTT broker
client.setServer(mqtt_broker, mqtt_port);
client.setCallback(callback);
while (!client.connected()) {
    String client_id = "esp32-client-";
    client_id += String(WiFi.macAddress());
    Serial.printf("The client %s connects to the public mqtt broker\n",
client_id.c_str());
    if (client.connect(client_id.c_str())) { //, mqtt_username,
mqtt_password)) {
        Serial.println("mqtt broker connected");
    } else {
```

```
        Serial.print("failed with state ");
        Serial.print(client.state());
        delay(2000);
    }
}
// Publish and Subscribe
client.publish(topic, "Hi I'm ESP32 ^^");
client.subscribe(topic);

}

void callback(char *topic, byte *payload, unsigned int length) {
 Serial.print("Message arrived in topic: ");
 Serial.println(topic);
 Serial.print("Message:");
 for (int i = 0; i < length; i++) {
   Serial.print((char) payload[i]);
 }
 Serial.println();
 Serial.println("-----------------------");
}

void loop()
{
   // Publish to subscriber inside the loop
   client.publish(topic, 1);
   client.loop();
}
```

MQTT Code for laptop as a Subscriber (Python)

```
import paho.mqtt.client as mqtt


# 0. define callbacks - functions that run when events happen.
# The callback for when the client receives a CONNACK response from the
server.
def on_connect(client, userdata, flags, rc):
 print("Connection returned result: " + str(rc))

 # Subscribing in on_connect() means that if we lose the connection and
 # reconnect then subscriptions will be renewed.
 client.subscribe("ece180d/test", qos=1)
```

```
# The callback of the client when it disconnects.
def on_disconnect(client, userdata, rc):
 if rc != 0:
   print('Unexpected Disconnect')
 else:
   print('Expected Disconnect')


# The default message callback.
# (you can create separate callbacks per subscribed topic)
def on_message(client, userdata, message):
 print('Received message: "' + str(message.payload) + '" on topic "' +
       message.topic + '" with QoS ' + str(message.qos))


# 1. create a client instance.
client = mqtt.Client()
# add additional client options (security, certifications, etc.)
# many default options should be good to start off.
# add callbacks to client.
client.on_connect = on_connect
client.on_disconnect = on_disconnect
client.on_message = on_message

# 2. connect to a broker using one of the connect*() functions.
# client.connect_async("test.mosquitto.org")
client.connect_async('mqtt.eclipseprojects.io')
# client.connect("test.mosquitto.org", 1883, 60)
# client.connect("mqtt.eclipse.org")

# 3. call one of the loop*() functions to maintain network traffic flow
with the broker.
client.loop_start()
# client.loop_forever()

while True:  # perhaps add a stopping condition using some break or
something.
 pass  # do your non-blocked other stuff here, like receive IMU data or
something.
# use subscribe() to subscribe to a topic and receive messages.
```

```
# use publish() to publish messages to the broker.

# use disconnect() to disconnect from the broker.
client.loop_stop()
client.disconnect()
```

Expected Output

Once each respective code is uploaded or executed. The integer 1 should be received on the Laptop side. An example output is below:

```
Received message 1 on topic test/topic with QoS 1
Received message 1 on topic test/topic with QoS 1
Received message 1 on topic test/topic with QoS 1
```

Conclusion

This tutorial showed how to send a simple integer from an IoT device like the ESP32 to a laptop using a message protocol called MQTT. The codes shown in this tutorial can be extended in such a way that sensor/telemetry values like those outputted by IMU sensors can be sent from the ESP32 to the laptop for processing. At the same time, the tutorial above can also be extended to send data from a laptop to an ESP32 microcontroller.

References:

Bernstein, C., Brush, K., & Gillis, A. S. (2021, January 27). *What is MQTT and how does it work?* IoT Agenda. Retrieved February 8, 2023, from https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport

MQTT.org. (2022). *The standard for IOT messaging*. MQTT. Retrieved February 8, 2023, from https://mqtt.org/

Rothuis, A. (2018, October 7). *Messaging Pattern: Publish-Subscribe*. A. ROTHUIS. Retrieved February 8, 2023, from https://www.arothuis.nl/posts/messaging-pub-sub/

Chaurasia, A. (2020, November 15). *Wired communication protocols in IOT - Pianalytix - machine learning*. Pianalytix. Retrieved February 10, 2023, from https://pianalytix.com/wired-communication-protocols-in-iot/