MQTT (Message Queuing Telemetry Transport) protocol

MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe based messaging protocol designed to provide reliable transmission of data packets of all types for devices. It's a popular mechanism for devices constrained by power and computing. MQTT is commonly used for IoT applications to connect to and communicate with low-power devices such as sensors, actuators, and home appliances, efficiently managing the payload of data transmitted.

MQTT uses TCP.

TCP/IP port 1883 is reserved with IANA for use with MQTT. TCP/IP port 8883 is also registered, for using MQTT over SSL.

Uses a Broker/Server model.

MQTT Clients

Because MQTT clients don't have addresses like email addresses, phone numbers etc. you don't need to assign addresses to clients like you do with most messaging systems.

For MQTTv3.1.1 there is client software available in almost all programming languages and for the main operating systems Linux, Windows, Mac from the Eclipse Paho project.

- Paho Python client.
- Node.js MQTT Client-Starting Guide
- JavaScript Websockets Client
- Client for ESP32

https://test.mosquitto.org/

Programming Challenge 3

1) Upload and test the two MQTT demo sender and client programs on two ESP32 boards. These two demo programs use the free **test.mosquitto.org** MQTT server.

2) Modify the MQTT demo sender and receiver programs to turn on and off the built-in LED. The sender will send either an "ON" or "OFF" message under the same topic. Upon receiving the message, the receiver will act accordingly. Program any user interface you like for the sender.

   Sample Arduino code to turn on and off the led and to flash the led.

```
#define statusLed 2

void setup() {
  pinMode(statusLed, OUTPUT);     // init the port for output
```

```
  digitalWrite(statusLed, HIGH);  // turn on led
  digitalWrite(statusLed, LOW);   // turn off led
}

void loop() {
  // flash the led every 1000 millisecond
  digitalWrite(statusLed, HIGH);  // to turn on
  delay(1000);
  digitalWrite(statusLed, LOW);   // to turn off
  delay(1000);
}
```

Sample Arduino code to read bytes from the Serial console.

```
int incomingByte;

void setup() {
  Serial.begin(115200); // opens serial port, sets data rate to 115200 bps
  Serial.println("Enter something and press Send");
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read(); // read the incoming byte
    Serial.print("Received ");
    Serial.print(incomingByte);
    Serial.print(", [");
    Serial.print((char)incomingByte);
    Serial.println("]");
  }
}
```

3) Instead of using the free **test.mosquitto.org** server, install and configure your own MQTT server on any computer platform you like. See instructions below.

4) Do number 2) again but using your MQTT server.

5) Instead of running the demo sender program from number 2) on an ESP32, write your own sender/publisher program for your computer. Test it with the receiver program from number 2).

   Here's a nice C example for both the sender/publisher and receiver/subscriber that you can use and follow.

   https://github.com/LiamBindle/MQTT-C

Setting up your own MQTT broker

Mosquitto is a free open source MQTT broker that runs on Windows and Linux.

https://mosquitto.org/

For Linux

Install Mosquitto on Ubuntu

sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa

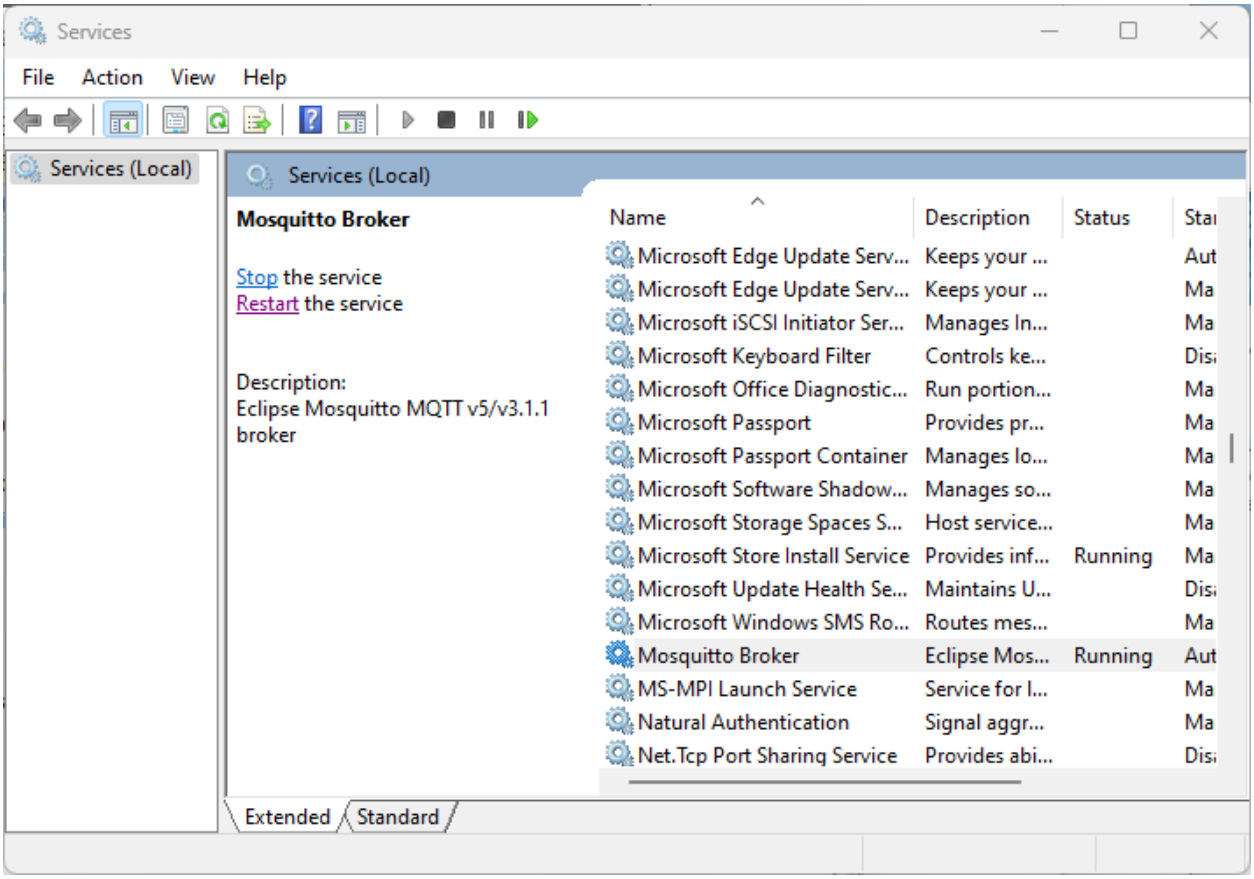
sudo apt-get update

For Windows

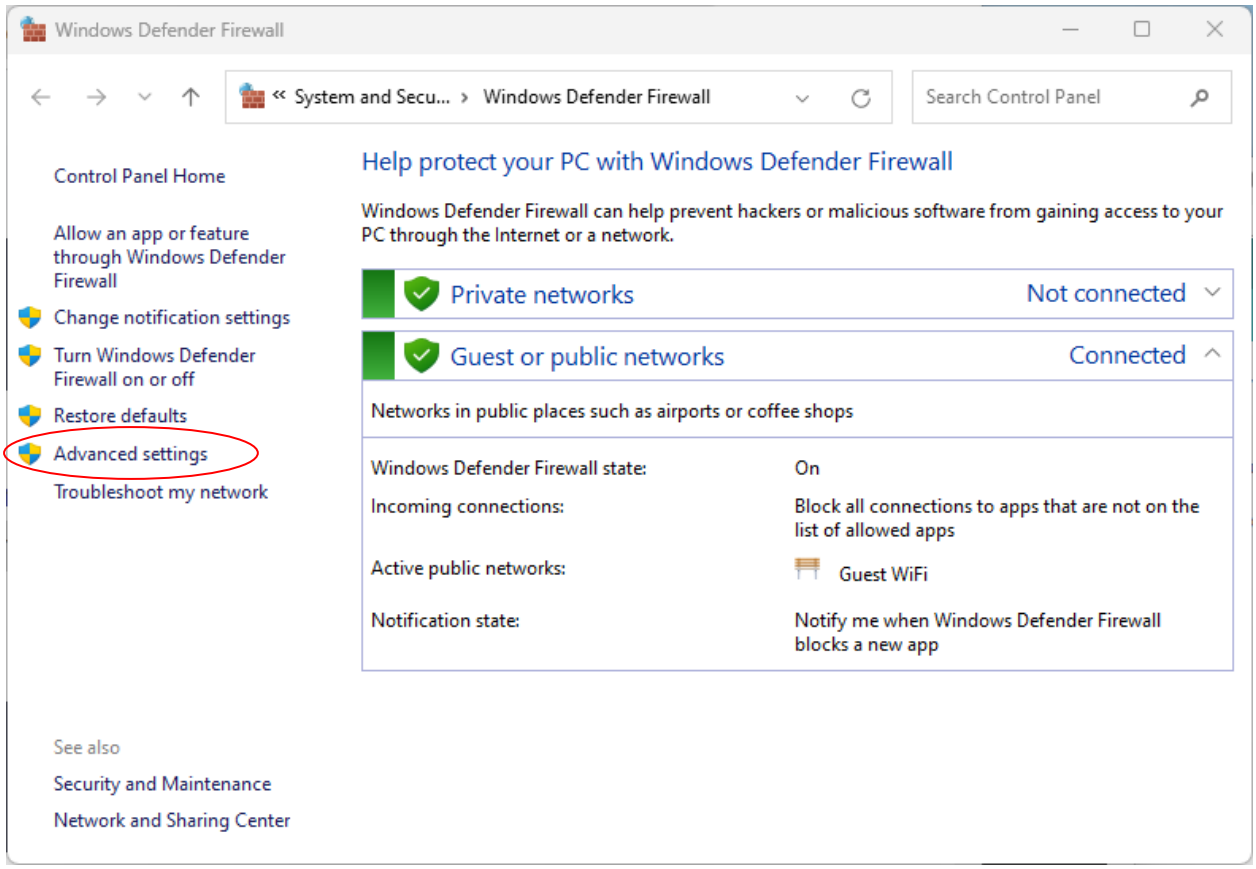
https://cedalo.com/blog/how-to-install-mosquitto-mqtt-broker-on-windows/#How_to_install_Mosquitto_MQTT_Broker_on_Windows

Download from mosquitto.org and run the installation program.

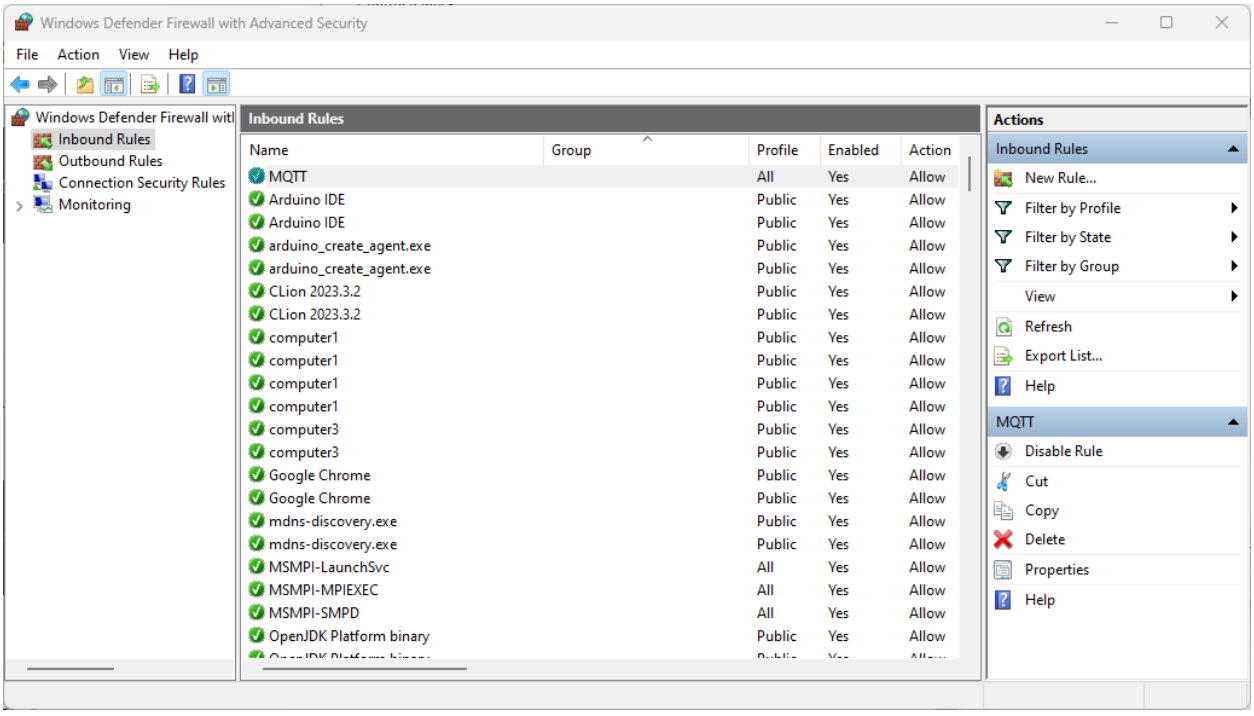
Need to start the service on Windows. Search for ***services***



Need to punch hole through Windows' Defender Firewall for port 1883



Create a new Inbound Rule to allow incoming traffic through port 1883



Edit the mosquito.conf file in

C:/Program Files/mosquito/mosquitto.conf

And add the two uncommented lines shown in red below

```
# affected:
#
# acl_file
allow_anonymous true
listener 1883 0.0.0.0
# allow_zero_length_clientid
# auto_id_prefix
# password_file
# plugin
```

Stop the Mosquitto service and restart it again. The publisher and subscriber should work now. Make sure you're connecting to the correct Mosquitto Broker IP address.