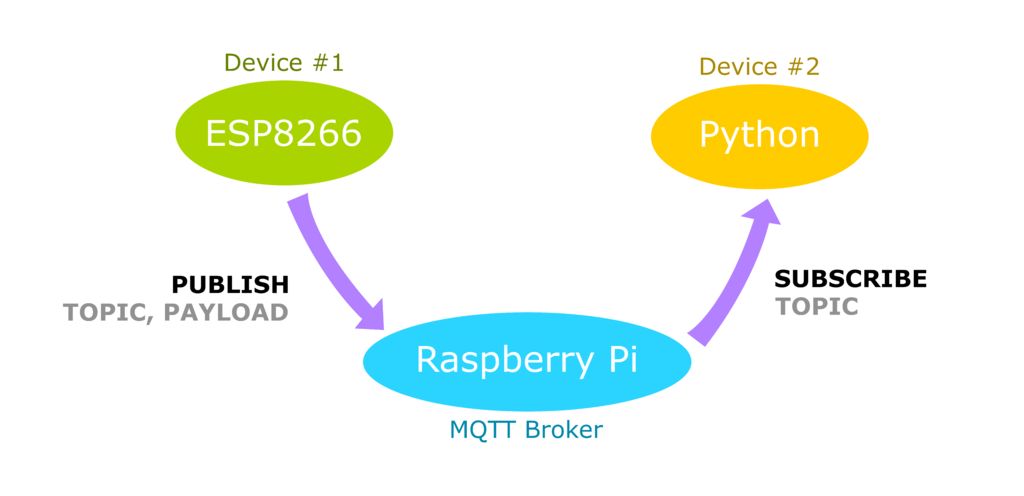
**MQTT With the Raspberry Pi and ESP8266**

MQTT is based around the idea that devices can publish or subscribe to topics. So, for example, if Device #1 has recorded the temperature from one of its sensors, it can publish a message which contains the temperature value it recorded, to a topic (e.g. "Temperature"). This message is sent to an MQTT Broker, which you can think of as a switch/router on a local area network. Once the MQTT Broker has received the message, it will send it to any devices (in this case, Device #2) which are subscribed to the same topic.



## STEP 1. Installing the MQTT Broker on the Raspberry Pi

For the Raspberry Pi, we will be using the "Mosquitto" MQTT broker. Before we install this, it is always best to update our Raspberry Pi.

sudo apt-get update

sudo apt-get upgrade

Once you've done this, install **mosquitto**and then the **mosquitto-clients** packages.

sudo apt-get install mosquitto -y

sudo apt-get install mosquitto-clients -y

When you've finished installing these two packages, we are going to need to configure the broker. The mosquitto broker's configuration file is located at **/etc/mosquitto/mosquitto.conf.**

Open a Text Editor, In this case , Nano Editor

sudo nano /etc/mosquitto/mosquitto.conf

At the bottom of this file, you should see the line:

include\_dir /etc/mosquitto/conf.d

Delete this line. Add the following lines to the bottom of the file.

allow\_anonymous false

password\_file /etc/mosquitto/pwfile

listener 1883

By typing those lines, we've told mosquitto that we don't want anyone connecting to our broker who doesn't supply a valid username and password (we'll get on to set these in a second) and that we want mosquitto to listen for messages on port number 1883.

If you don't want the broker to require a username and password, don't include the first two lines that we added (i.e. allow\_anonymous... and password\_file...). If you have done this, then skip to rebooting the Raspberry Pi.

Now close (and save) that file. If you are following along with the nano example, press **CTRL+X**, and type **Y** when prompted.

Because we've just told mosquitto that users trying to use the MQTT broker need to be authenticated, we now need to tell mosquitto what the username and password are! So, type the following command - replacing **username** with the username that you would like - then enter the password you would like when prompted (Note: if, when editing the configuration file, you specified a different **password\_file** path, replace the path below with the one you used).

sudo mosquitto\_passwd -c /etc/mosquitto/pwfile username

As we've just changed the mosquitto configuration file, we should reboot the Raspberry Pi.

sudo reboot

Once the Raspberry Pi has finished rebooting, you should have a fully functioning MQTT broker.

## Step 2: Testing the Broker

Once you've installed mosquitto on the Raspberry Pi, you can give it a quick test - just to make sure everything is working correctly. For this purpose, there are two commands that we can use on the command line. **mosquitto\_pub** and **mosquitto\_sub**. In this step, I will guide you through using each of these to test our broker.

In order to test the broker, you will need to open two command line windows. If you are using Putty or another SSH client, this is as simple as opening another SSH window and logging in as usual. If you are accessing your Pi from a UNIX terminal, this is exactly the same. If you are using the Raspberry Pi directly, you will need to open two terminal windows in the GUI mode (the command **startx**can be used to start the GUI).For this case, we are using Putty.

Now that you have opened two windows, we can get started on the testing. In one of the two terminals, type the following command, replacing **username** and **password**with the ones you setup in the previous step.

mosquitto\_sub -d -u username -P password -t test

If you decided not to set a username and password in the previous step, then from now on, ignore the -u and -P flags in the commands. So, as an example, the mosquitto\_sub command would now be:

mosquitto\_sub -d -t test

The mosquitto\_sub command will subscribe to a topic, and display any messages that are sent to the specified topic in the terminal window. Here, **-d** means **debug mode**, so all messages and activity will be output on the screen. **-u** and**-P**should be self-explanatory. Finally, **-t** is the name of the **topic**we want to subscribe

## Step 3: Programming the ESP8266

Now we will begin to program the ESP8266, but before we can start, we will need to install the following libraries in the Arduino Library manager (Sketch->Include Libraries->Manage Libraries)

* Bounce2
* PubSubClient

Once you've installed those libraries, you will be able to run the code.Install the mosquitto python client, you just need to type the following into the command line (Linux/Mac) or even command prompt (Windows).

pip install paho-mqtt

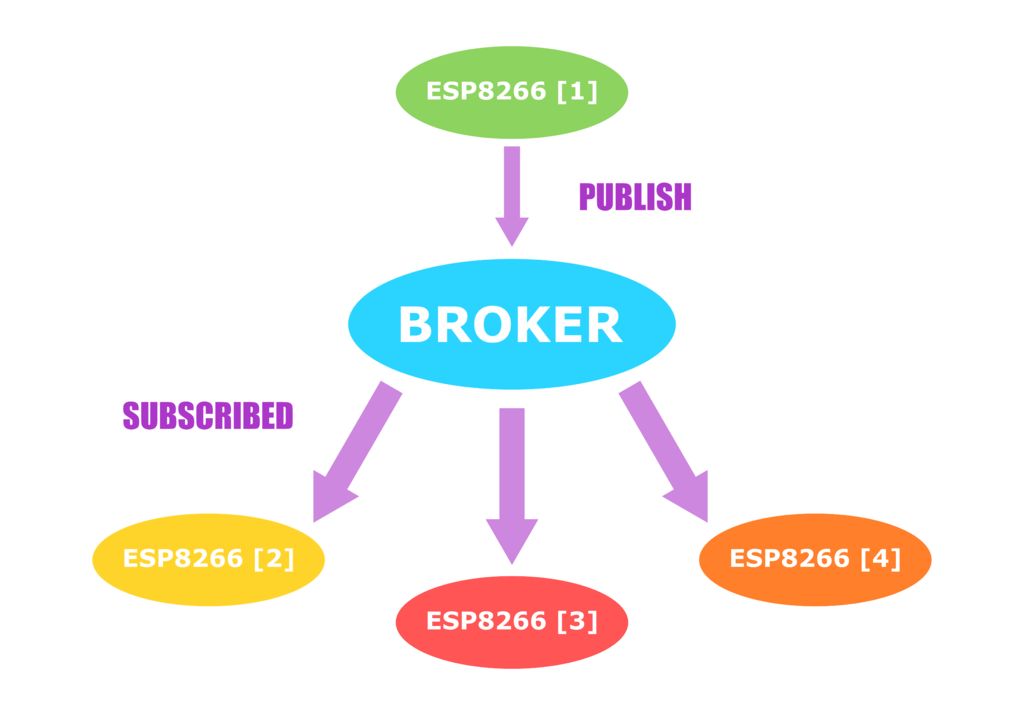
## Step 4: Python Client - Subscribing

In this step, must setup the Python script (either on the Raspberry Pi itself or on another computer connected to the network) to handle all of the messages that are sent (published) by the ESP8266 to the MQTT topic.

## Mosquito\_sub –h 169.254.152.165 –u \_\_\_\_\_ –P \_\_\_\_\_ –t “\_\_\_\_\_\_”

## Step 5: Communicating Between ESP8266 Devices

For one ESP to send data to another, the first ESP will need to **publish**to the topic, and the second ESP will need to **subscribe**to that topic. This setup will allow for a one-way conversation - ESP(1) to ESP(2). If we want ESP(2) to talk back to ESP(1), we can create a new topic, to which ESP(2) will publish, and ESP(1) will subscribe. Thankfully, we can have multiple subscribers on the same topic, so if you want to send data to a number of systems, you will only need one topic (to which they all subscribe, except the device which is sending the data, as that will be publishing).



**Code:**

#include<ESP8266WiFi.h>

#include<PubSubClient.h>

const char\* ssid="123";

const char\* password="lasya123";

const char\* mqttServer="192.168.43.180";

const int mqttPort=1883;

const char\* mqttUser="pi";

const char\* mqttPassword="root";

WiFiClient espClient;

PubSubClient client(espClient);

void setup() {

Serial.begin(115200);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid,password);

while(WiFi.status()!=WL\_CONNECTED){

delay(500);

Serial.println("connecting to wifi network...................");

}

Serial.println("Connected to wifi");

client.setServer(mqttServer,mqttPort);

client.setCallback(callback);

while(!client.connected()){

Serial.println("connecting to mqtt..................");

if(client.connect("ESP8266Client",mqttUser,mqttPassword)){

Serial.println("connected...");

} else

{

Serial.print("failed with state ");

Serial.println(client.state());

Serial.println();

delay(2000);

}

}

}

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() {

float temp\_0;

float temp\_1;

DS18B20.requestTemperatures();

temp\_0 = 85.00; //DS18B20.getTempCByIndex(0); // Sensor 0 will capture Temp in Celcius

dtostrf(temp\_0,4,2,temp\_0\_str);

client.publish("2\_esp8266",temp\_0\_str);

client.subscribe("2\_esp8266");

temp\_1 = 75.00; //DS18B20.getTempFByIndex(0); // Sensor 0 will capture Temp in Fahrenheit

dtostrf(temp\_1,4,2,temp\_1\_str);

client.publish("2\_esp8266",temp\_1\_str);

client.subscribe("2\_esp8266");

// read the input on analog pin 0:

float moist =12.34; //analogRead(A0);

dtostrf(moist,6,3,moist\_str);

client.publish("2\_esp8266",moist\_str);

client.subscribe("2\_esp8266");

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration\_ = pulseIn(echoPin, HIGH);

// Calculating the distance

float dist=100; //duration\_\*0.034/2;

dtostrf(dist,4,0,dist\_str);

client.publish("2\_esp8266",dist\_str);

client.subscribe("2\_esp8266");

Serial.println();

}