# Multi-Agent IOT Door Sensor

## Online Link:

This Project is available on my online portfolio at: http://timothyjgow.com/

## Objective:

The purpose of this project is to learn how to set up a MQTT Broker (server) that will be able to pass notifications from a Arduino equipped with a door open sensor to turn on a distance sensor, which in turn will control the outputs of 3 led lights.

## Overview:

Message Queuing Telemetry Transport (MQTT) protocol is a lightweight subscribe/publish communicator. It runs over TCP/IP and has a small footprint over the network. It works by setting up a broker also know as a server to facilitate publishers and subscribers. Publishers publish content (messages) and anyone subscribed to the publisher will receive all content that the publisher publishes.

For this project we will be building of the previous project (Arduino Distance Indicator) which you can find here http://timothyjgow.com/ I suggest doing that project first if you haven't done it yet.

## Materials:

The materials needed for this project are the following:

- Personal Computer
- Arduino IDE
- 3x Wemos D1 Mini
- 3x MicroUSB cable
- 3x Breadboard
- HC-SR04 Ultrasonic Sensor
- 1x Red LED
- 1x Yellow LED
- 1x Green LED
- 8x Jumper wires

- 3x 220 Ω Resistors
- Reed Switch
- Raspberry Pi

## References:

https://en.wikipedia.org/wiki/MQTT description of MQTT

https://www.instructables.com/How-to-Use-MQTT-With-the-Raspberry-Pi-and-ESP8266/ Used to understand how to set up MQTT publisher on Arduino and Raspberry Pi Broker

https://github.com/knolleary/pubsubclient/blob/master/examples/mqtt_esp8266/mqtt_esp8266.ino Used to set up MQTT on Arduinos

https://escapequotes.net/esp8266-wemos-d1-mini-pins-and-diagram/ used for pin reference on the Arduinos

https://arduinogetstarted.com/tutorials/arduino-door-sensor This is were I got my code for the reed switch

## Procedure:

1. Let's start with setting up our Raspberry Pi. Any version will work. As long as you can connect it to your internet. Open up a terminal and ssh into your PI. Go ahead and run the following commands.

```
sudo apt-get update
sudo apt-get upgrade
```

Install mosquito which is what we will be using for our MQTT server

```
sudo apt-get install mosquitto -y
sudo apt-get install mosquitto-clients -y
```

Next we will change some items in the config file

```
sudo nano /etc/mosquitto/mosquitto.conf
```

At the bottom in the file delete this line

```
include_dir /etc/mosquitto/conf.d
```

Replace it with

```
listener 1883
```

This will be the port will listen on for subscription and publication events.

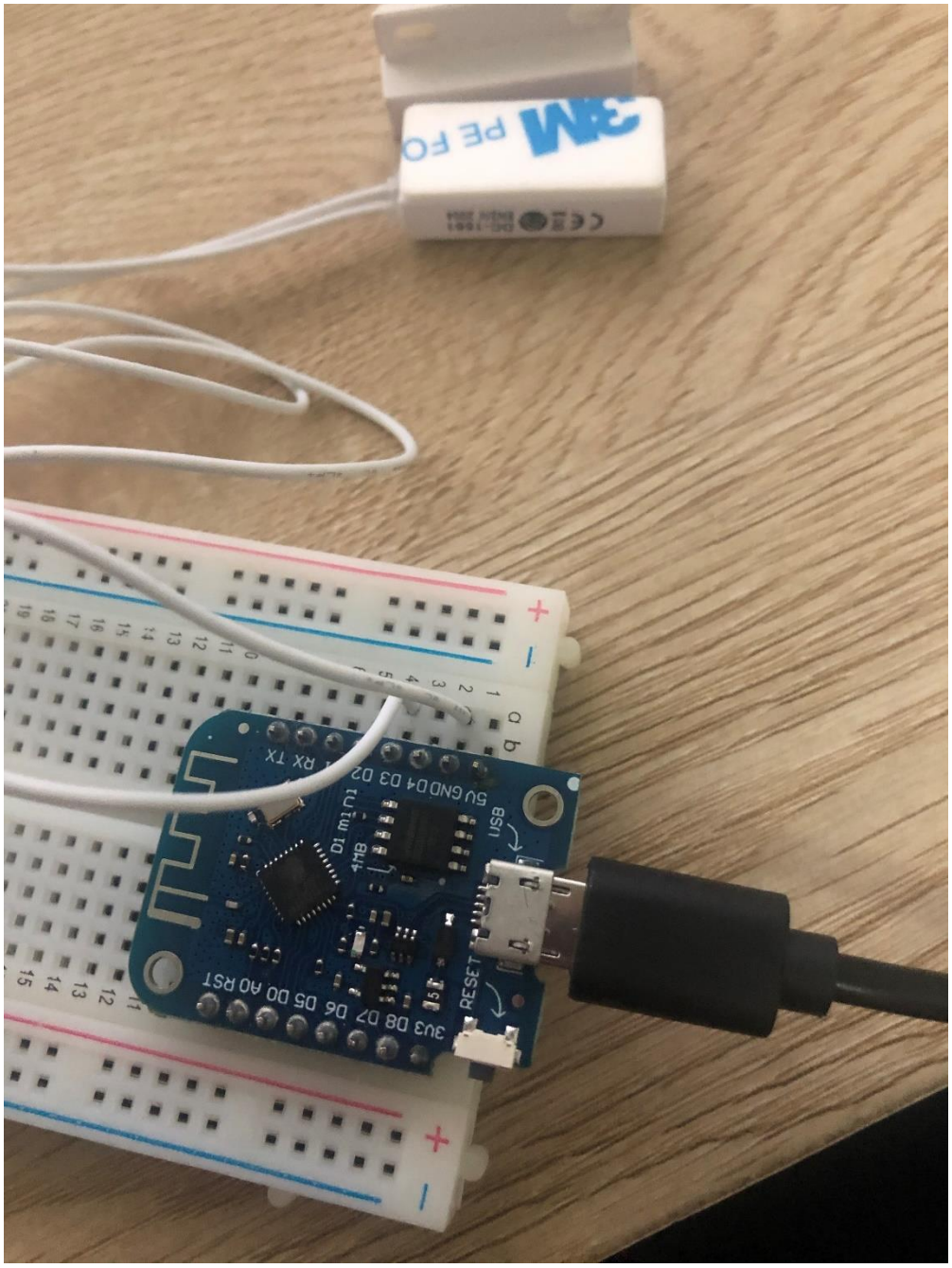
Now you will reboot your PI

```
sudo reboot
```

2. The second step is to now setup the Arduino that will be publishing the door open and close events

The wiring is pretty easy on this device. Connect one end of the Reed switch to GND and the other to D3. D3 is a 10kohm pullup pin that we will be using to know whether it is closed or not.

Use the image below to set up wiring.




3. Begin coding for the Arduino reed switch sensor. This Arduino will publish to a topic called door_status. The messages will be either 1 for closed or 0 for open.

4. You will need to add PubSubClient library in the Arduino IDE before you run the code.

5. The code for the door sensor is below. Make sure to replace any variables with your specific values for your setup.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "your network ssid";
const char* wifi_password = "your password";
const char* mqtt_server = "ip address on the pi";
const char* clientID = "door_sensor";
const char* mqtt_topic = "door_status";

const int DOOR_SENSOR_PIN = 0;  // Arduino pin connected to door sensor's pin

int currentDoorState; // current state of door sensor
int lastDoorState;    // previous state of door sensor

WiFiClient wifiClient;
PubSubClient client(mqtt_server, 1883, wifiClient);


void setup() {
  Serial.begin(9600);                    // initialize serial
  pinMode(DOOR_SENSOR_PIN, INPUT_PULLUP); // set arduino pin to input pull-
up mode
  pinMode(LED_BUILTIN, OUTPUT);

  Serial.print("Connecting to ");
  Serial.println(ssid);

  // Connect to the WiFi
  WiFi.begin(ssid, wifi_password);

  // Wait until the connection has been confirmed before continuing
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // Debugging - Output the IP Address of the ESP8266
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

```
  currentDoorState = digitalRead(DOOR_SENSOR_PIN); // read state

    // Connect to MQTT Broker
  // client.connect returns a boolean value to let us know if the connection was
successful.
  if (client.connect(clientID)) {
    Serial.println("Connected to MQTT Broker!");
  }
  else {
    Serial.println("Connection to MQTT Broker failed...");
  }

}

void loop() {


  lastDoorState = currentDoorState;              // save the last state
  currentDoorState  = digitalRead(DOOR_SENSOR_PIN); // read new state

  if (lastDoorState == LOW && currentDoorState == HIGH) { // state change: LOW -
> HIGH
    Serial.println("The door-opening event is detected");
    digitalWrite(LED_BUILTIN, LOW);
    if (client.publish(mqtt_topic, "1")) {
      Serial.println("door closed and message sent!");
    }
    // Again, client.publish will return a boolean value depending on whether it
succeded or not.
    // If the message failed to send, we will try again, as the connection may ha
ve broken.
    else {
      Serial.println("Message failed to send. Reconnecting to MQTT Broker and try
ing again");
      client.connect(clientID);
      delay(10); // This delay ensures that client.publish doesn't clash with the
 client.connect call
      client.publish(mqtt_topic, "1");
    }
  }
  else
  if (lastDoorState == HIGH && currentDoorState == LOW) { // state change: HIGH -
> LOW
    Serial.println("The door-closing event is detected");
    digitalWrite(LED_BUILTIN, HIGH);
```

```
    if (client.publish(mqtt_topic, "0")) {
      Serial.println("door open and message sent!");
    }
    // Again, client.publish will return a boolean value depending on whether it
succeded or not.
    // If the message failed to send, we will try again, as the connection may ha
ve broken.
    else {
      Serial.println("Message failed to send. Reconnecting to MQTT Broker and try
ing again");
      client.connect(clientID);
      delay(10); // This delay ensures that client.publish doesn't clash with the
 client.connect call
      client.publish(mqtt_topic, "0");
    }
  }
}
```

6. The next step is to setup the other two boards with the correct code to connect to the server. For wiring you can refer to the previous project mentioned above. Wiring won't change this round. The only thing that will be changing is we will be no longer using http for our communication protocol.

7. Code for the distance indicator:

```


#include <ESP8266WiFi.h>
#include <PubSubClient.h>


// Update these with values suitable for your network.

const char* ssid = "Your ssid";
const char* wifi_password = "your password";
const char* mqtt_server = "your pi IP address";
const char* clientID = "distance_sensor";

int Echo = 12;
int Trig = 14;
long duration;
int distance;
```

```
String state;


WiFiClient wifiClient;
PubSubClient client(mqtt_server, 1883, wifiClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

void callback(char* topic, byte* payload, unsigned int length) {
  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the vo
ltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
    state = "open";
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage H
IGH
    state = "closed";
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    if (client.connect(clientID)) {
      Serial.println("connected");
      client.subscribe("door_status");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void setup() {
  pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an outpu
t
  pinMode(Trig, OUTPUT);
```

```
  pinMode(Echo, INPUT);
  Serial.begin(9600);


  Serial.print("Connecting to ");
  Serial.println(ssid);

  // Connect to the WiFi
  WiFi.begin(ssid, wifi_password);

  // Wait until the connection has been confirmed before continuing
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // Debugging - Output the IP Address of the ESP8266
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  client.setCallback(callback);
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if (state == "open") {
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(200);
  digitalWrite(Trig, LOW);

  duration = pulseIn(Echo, HIGH);

  distance = duration * 0.034/2;
  if (distance >= 20 ) { //far away still
    client.publish("distance_status", "0"); // tell client to turn on its green l
ed.
  }
```

```
  else if (distance > 10 && distance < 20) { //getting close but still not there
    client.publish("distance_status", "1"); // tell client to turn on its yellow
led.
  }
  else if (distance > 7 && distance <= 10) { // good distance between 4cm and 7cm
    client.publish("distance_status", "2"); // tell clien to turn on its yellow l
ed.
  }
  else if (distance <= 7) { // too close back up;
    client.publish("distance_status", "3");
  }
  }

}
```

8. Next add the code for the LEDs:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "your ssid";           // SSID of your home WiFi
const char* password = "your password";           // password of your home WiFi
int GreenLED = 15;
int YellowLED = 13;
int RedLED = 12;
int flashing = 0;


const char* mqtt_server = "your pi IP address";
const char* clientID = "lights";

WiFiClient wifiClient;
PubSubClient client(mqtt_server, 1883, wifiClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

void callback(char* topic, byte* payload, unsigned int length) {

  if ((char)payload[0] == '0'){
    digitalWrite(GreenLED, HIGH);
    digitalWrite(YellowLED, LOW);
    digitalWrite(RedLED, LOW);
```

```
    flashing = 0;
  }
  else if ((char)payload[0] == '1') {
    digitalWrite(GreenLED, LOW);
    digitalWrite(YellowLED, HIGH);
    digitalWrite(RedLED, LOW);
    flashing = 0;
  }
  else if ((char)payload[0] == '2') {
    digitalWrite(GreenLED, LOW);
    digitalWrite(YellowLED, LOW);
    digitalWrite(RedLED, HIGH);
    flashing = 0;
  }
  else if ((char)payload[0] == '3') {
    flashing = 1;
    digitalWrite(GreenLED, LOW);
    digitalWrite(YellowLED, LOW);
    digitalWrite(RedLED, LOW);
  }
  if (flashing == 1) {
    digitalWrite(RedLED, LOW);
    delay(300);
    digitalWrite(RedLED, HIGH);
    delay(300);
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    if (client.connect(clientID)) {
      Serial.println("connected");
      client.subscribe("distance_status");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

```
void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);             // connects to the WiFi router
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
 Serial.println("Connected to wifi");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
  pinMode(GreenLED, OUTPUT);
  pinMode(YellowLED, OUTPUT);
  pinMode(RedLED, OUTPUT);

  client.setCallback(callback);
}

void loop () {
    if (!client.connected()) {
    reconnect();
  }

  client.loop();

}
```

9. After you have added and uploaded the code you can test out the functionality of the devices. If the door is closed the distance measure will do anything or send any commands to change the LEDs until the door is open. The distance sensor will then start reading distances and send commands via MQTT to the LED Arduino.

## Thought Questions:

1. How does the communication between devices change with the shift to using an event hub? How does this facilitate greater scalability? What things did you do to learn to use MQTT?

   The raspberry pi as the event hub makes communication much faster. There is not as much of a delay when using MQTT. This makes communicating to devices much easier. To learn MQTT I looked up some tutorials and read some documentation on how it works.

2. What are strengths and weaknesses of the direct communication between the sensor and actuator? What are strengths and weaknesses of the event hub? Which do you feel is better for this application?

   I can really easily Identify weaknesses with MQTT it is really lightweight and has its purpose for messaging that does not have to be really complex. I feel like MQTT is much better suited than using HTTP.
3. What was the biggest challenge you overcame in this lab?

   Figuring out that I needed to flash the whole Arduino device.

4. Please estimate the total time you spent on this lab and report

   10 hrs

## Appendix:

Code for the Arduino devices can be found at my github.

https://github.com/Choyero/Multi-Agent-MQTT-Hub.git