Lab 4 – MQTT Event Hub – Garage Door Sensor

Online Link

This lab is available as part of my online portfolio at: https://github.com/Brianhayden7/Arduino-MQTT-Hub

Objective

The purposes of this lab are to:

· Implement an Event Hub for publish/subscribe notifications between devices.

· Develop a communications protocol for devices across the event bus.

· Establish more complex conditions for the actuator involving multiple sensors.

Materials

I used the following materials to accomplish this lab:

- Personal computer w/ Mosquitto MQTT Broker
- 3 x Arduino wemos d1 mini
- 3 x MicroUSB Cable
- 1 x power brick
- 3 x breadboard
- 1 x HC-SR04 Distance Sensor
- 1 x Magnetic Sensor
- 1 x Red LED
- 1 x Yellow LED
- 1 x Green LED
- 3 x 100 Ω Resistor (BrBlBrGold)
- Jumper Wires

References

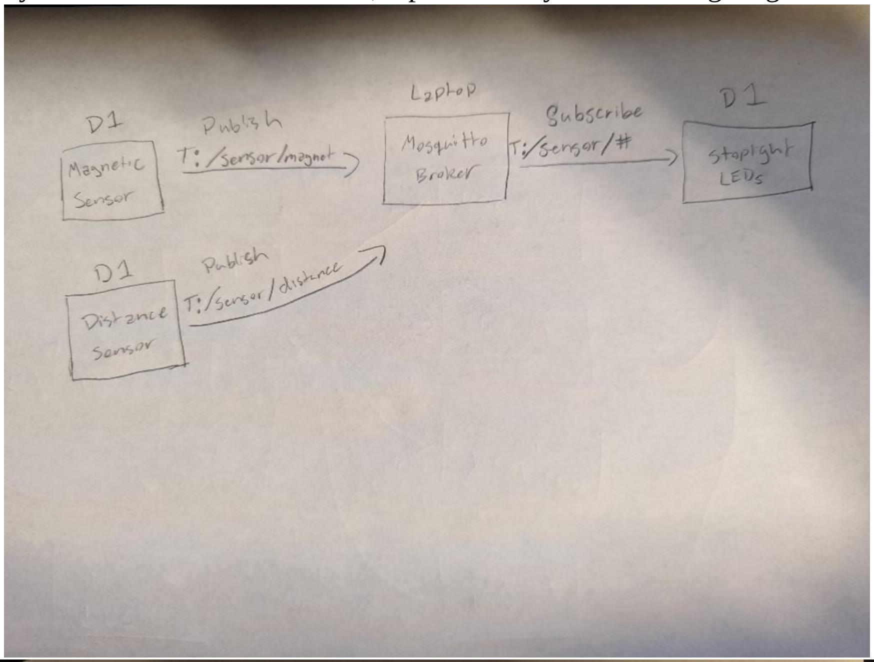
I used the following resources in this lab:

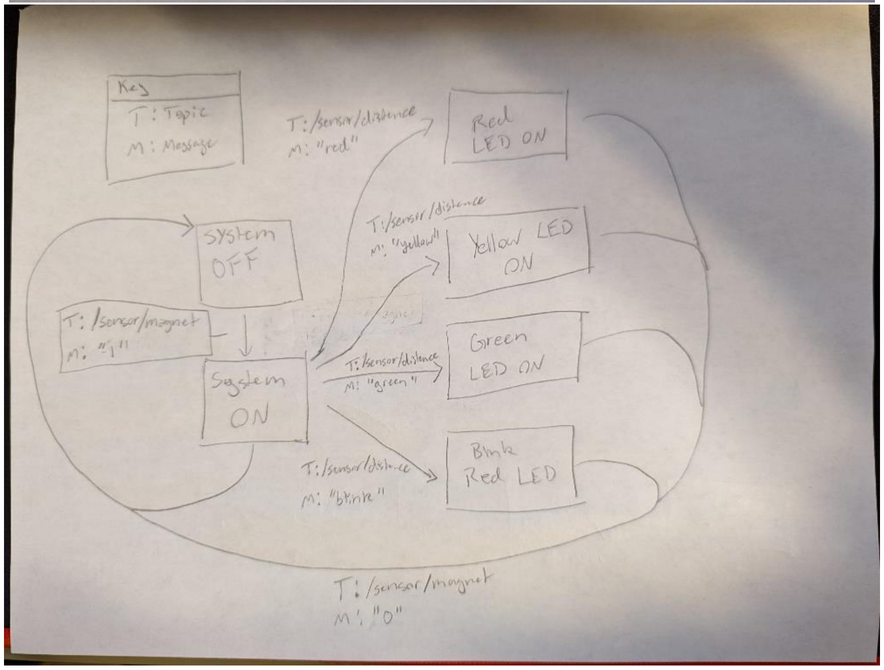
- https://github.com/Brianhayden7/Distance-Sensor My previous lab implementation of stoplight and distance sensor system
- https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device Guide on how to set up publishers and subscribers on Arduino using mqtt
- https://create.arduino.cc/projecthub/1NextPCB/how-to-use-a-magnetic-door-sensor-1f9439 Simple guide on how to wire up and code the magnetic sensor
- https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-10#1-overviewhttps://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-10#1-overview Guide for installing ubuntu/wsl (windows subsystem for linux) on windows
- https://www.vultr.com/docs/install-mosquitto-mqtt-broker-on-ubuntu-20-04-server/ Mosquitto install and usage tutorial

Procedures:

1. Download and install the arduino IDE.
2. Search for the library that includes the esp8266 boards.
3. Configure the arduino IDE to use the wemos d1 mini and select the correct port.

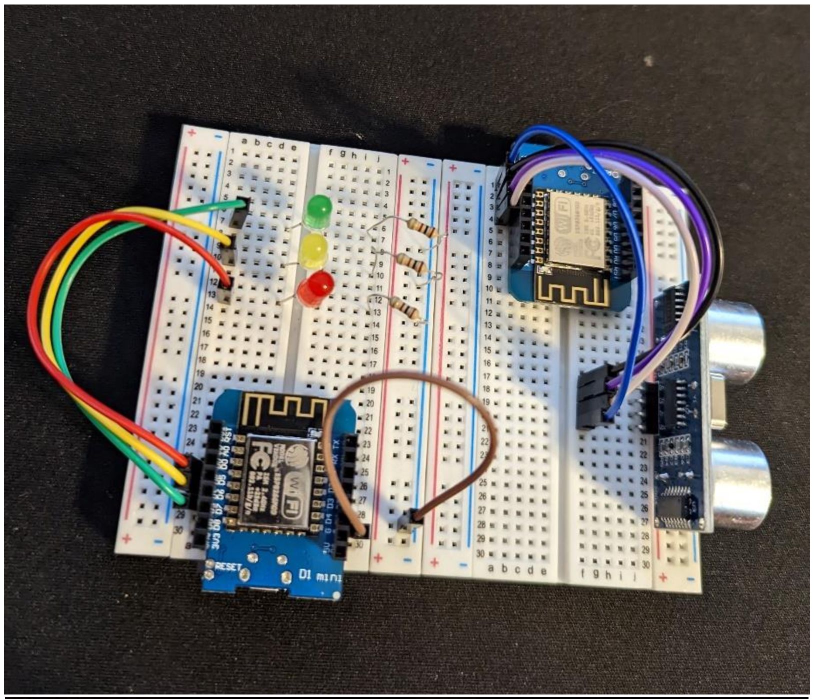
4. Using the magnetic sensor, set up the logic for when it is open or closed.
5. Install MQTT broker on laptop or pc
6. Adjust previous project code to communicate over MQTT instead of GET requests.
7. Set up topics to publish and subscribe to.
8. Set up the stoplight Arduino to subscribe to all sensor messages and configure logic branches appropriately.
9. Model the functionality, logical flow and components of the system. Include a schematic diagram.
    a. *Functionality* - This system is a basic state machine, represented by the following diagram:

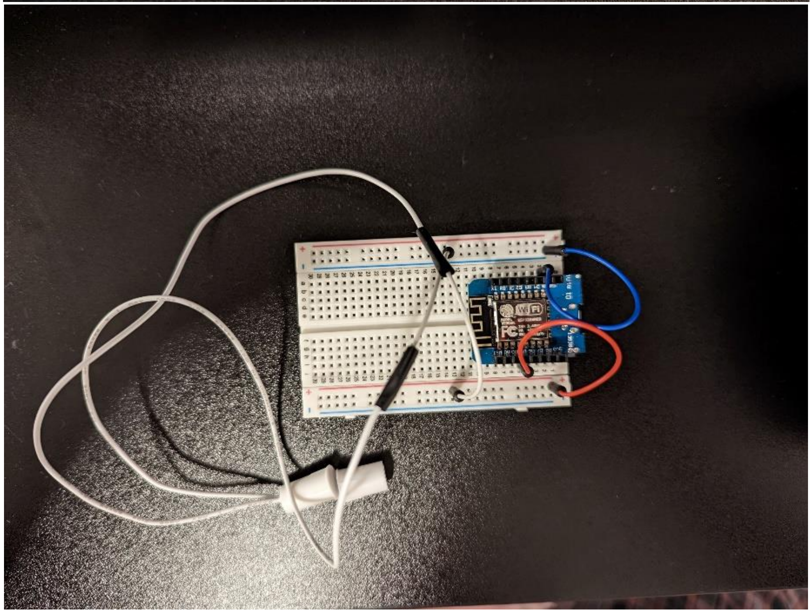




- Upon startup, it will start it the off position.

    b. *Component Diagram* and *Schematic Diagram* – This system is represented by the photos below.





Client Interactions

The system uses a central MQTT hub for managing sending information based on what data the sensors measure. This allows multiple devices to publish messages to the hub and then the stoplight Arduino can subscribe to all those messages and decide what to do depending on the message it receives. The topics and messages are as follows:

- Mosquitto_pub -t /sensor/magnet -m “0”
- Mosquitto_pub -t /sensor/magnet -m “1”
- Mosquitto_pub -t /sensor/distance -m “red”
- Mosquitto_pub -t /sensor/distance -m “yellow”
- Mosquitto_pub -t /sensor/distance -m “green”
- Mosquitto_pub -t /sensor/distance -m “blink”

- Mosquitto_sub -t /sensor/# *this subscribes the stoplight to all sensor messages

Observations

I actually liked using MQTT a lot more than the web routes and GET requests. I think that the system is more reliable this way and also more responsive as long as all the devices are properly connected to the broker. There are a lot more possibilities when using a system like this.

The ability to expand your network of devices from this point on seems a lot simpler now and also using data from the sensors is super easy now if there were other devices I wanted to add. All they need to do is subscribe to their messages and use those to do things. I don't need to go back and change the code on the sensor devices.

This helps my understanding of how large automated systems can easily and efficiently work together to make a whole system. Without doing things like this it can get a lot messier and add a lot of overhead onto a project, expansion, or making changes. This minimizes the downsides of the web system and adds a lot more functionality.

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?

This communication between devices is now being moderated by the MQTT broker which makes sending and receiving messages a lot more reliable and streamlined. With large amounts of messages being sent, it is necessary to have a dedicated device or service that is handling them all to prevent lag or backup or freezing. This makes it easier to add lots and lots of new devices without seeing these slow downs or breaks in the system. There are lots of great tutorials out there that helped me learn to implement MQTT on arduinos specifically which made it pretty easy.

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 think you can secure communication between a directly connected devices easier but it also makes coding and scalability harder when you have to go back and change code on all devices when you want to add connections. The event hub makes it easy to send and receive messages and adding a new device just needs to subscribe and it will have that connection without making any other changes. It does require another device which is a downside but it's not that bad.

3. What was the biggest challenge you overcame in this lab?

I had a hard time understanding how to parse out the MQTT messages on the subscriber end at first since I didn't realize that it only reads off one character at a time so you have to kind of build your message string before comparing them against your conditions. I was doing it each time a character was read basically so I was never getting the correct output I was expecting. This was mainly because I didn't fully understand some of the tutorial code I was using before actually implementing it which was a good learning experience I suppose.

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

I spent about 4 hours coding up this lab and then another 2 hours on this report.

Certification of Work

I certify that the solution presented in this lab represents my own work. In the case where I have borrowed code or ideas from another person, I have provided a link to the author's work in the references and included a citation in the comments of my code.

--Brian Hayden

Appendix

Appendix 1: Arduino Code

(available at: https://github.com/Brianhayden7/Arduino-MQTT-Hub)

Stoplight Code:

```
#include <ESP8266WiFi.h>

#include <ArduinoMqttClient.h>


const char* wifiSSID = "Brian's Pixel"; // In order for this to work, you MUST specify the SSID for your wifi

const char* wifiPSK = "00000007"; // And the preshared key (wifi password)


int mytime = 0;

int redLED = D5;

int yellowLED = D6;

int greenLED = D7;

bool isLOOP = false;

bool firstLoop = true;

bool isOpen = false;

int state = -1;

String msg;

char letter;

String red = "red";

String yellow = "yellow";

String green = "green";

unsigned long previousMillis = 0;

const long interval = 100;

int ledState = LOW;



WiFiClient wifiClient;
```

```
MqttClient mqttClient(wifiClient);


const char broker[] = "192.168.18.68";

int port = 1883;

const char topic[]  = "/sensor/#";


WiFiServer server(80); // This sets which port our server will listen on


//WiFiClient client = server.available(); // Create a new client object
for available connections


void setup() {

  // put your setup code here, to run once:

  Serial.begin(9600);

  // int myTime = 0;

  pinMode(redLED, OUTPUT);

  pinMode(yellowLED, OUTPUT);

  pinMode(greenLED, OUTPUT);

  digitalWrite(greenLED, LOW);

  digitalWrite(yellowLED, LOW);

  digitalWrite(redLED, LOW);


  // ** Connect to WiFi network - Adapted from
http://www.esp8266learning.com/wemos-webserver-example.php


  Serial.print("Connecting to "); // Display debugging connection info


  Serial.println(wifiSSID); // Print the configured SSID to the Serial
Monitor
```

```
  WiFi.begin(wifiSSID, wifiPSK); // Use the provided SSID and PSK to
connect


  while (WiFi.status() != WL_CONNECTED) { // If not connected to wifi


  delay(500); // Pause


  Serial.print("."); // Print a dot each loop while trying to connect


  }


  Serial.println("");


  Serial.println("WiFi connected"); // Print "connected" message to the
Serial Monitor


  server.begin(); // Start the web server


  Serial.println("Server started");


  Serial.print("Use this URL : "); // Print the connected IP address to
the Serial Monitor


  Serial.print("http://");


  Serial.print(WiFi.localIP());


  Serial.println("/");


  // ** End Adapted Code - This is the end of the code that was adapted
from www.esplearning.com
```

```
  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1);
  }

  Serial.println("You're connected to the MQTT broker!");
  Serial.println();

  // set the message receive callback
  mqttClient.onMessage(onMqttMessage);

  Serial.print("Subscribing to topic: ");
  Serial.println(topic);
  Serial.println();

  // subscribe to a topic
  mqttClient.subscribe(topic);



}

void loop() {
  // put your main code here, to run repeatedly:
```

```
  //WiFiClient client = server.available(); // Create a new client object 
for available connections

  mqttClient.poll();


  unsigned long currentMillis = millis();

  int myTime = millis();

  //if(firstLoop){

  //  myTime = 0;

  //}

  if (currentMillis - previousMillis >= interval && isOpen && state == 3) 
{

    // save the last time you blinked the LED

    previousMillis = currentMillis;


    // if the LED is off turn it on and vice-versa:

    if (ledState == LOW) {

      ledState = HIGH;

    } else {

      ledState = LOW;

    }


    // set the LED with the ledState of the variable:

    digitalWrite(redLED, ledState);

  }


  if(state == 0 && isOpen){

    digitalWrite(greenLED, LOW);

    digitalWrite(yellowLED, LOW);

    digitalWrite(redLED, HIGH);

  }
  if(state == 1 && isOpen){
    digitalWrite(greenLED, LOW);
    digitalWrite(yellowLED, HIGH);
    digitalWrite(redLED, LOW);
  }
  if(state == 2 && isOpen){
    digitalWrite(greenLED, HIGH);
    digitalWrite(yellowLED, LOW);
    digitalWrite(redLED, LOW);
  }

  /*
  if((myTime / 2000) % 3 == 0 && isLOOP){
    // Serial.println("0");
    digitalWrite(greenLED, HIGH);
    digitalWrite(yellowLED, LOW);
    digitalWrite(redLED, LOW);
    firstLoop = false;
  }
  if((myTime / 2000) % 3 == 1 && isLOOP){
    // Serial.println("1");
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, LOW);
    digitalWrite(yellowLED, HIGH);
    firstLoop = false;
  }
  if((myTime / 2000) % 3 == 2 && isLOOP){
    // Serial.println("2");
    digitalWrite(yellowLED, LOW);
    digitalWrite(redLED, HIGH);
    digitalWrite(greenLED, LOW);

    firstLoop = false;

  }

  */

/*

  if (client) { // If a client is connected, wait until it sends some data

    //while (!client.available()) { // If the client hasn't sent info,
wait for it

    //  delay(10);

    //}


    String request = client.readStringUntil('\r'); // read the first line
of the request

    Serial.println(request); // Echo the request to the Serial Monitor for
debug


    client.flush(); // Wait until the buffers are clear


    if (request.indexOf("/loop") != -1) { // If the request is for the
page "/led=on"

      isLOOP = true;

      firstLoop = true;

    }


    if (request.indexOf("/led=OFF") != -1) { // If the request is for the
page "/led=off"

      isLOOP = false;

      digitalWrite(yellowLED, LOW);

      digitalWrite(redLED, LOW);

      digitalWrite(greenLED, LOW);

    }
    if (request.indexOf("/ledGREEN=ON") != -1) { // If the request is for
the page "/led=off"

      isLOOP = false;

      digitalWrite(yellowLED, LOW);

      digitalWrite(redLED, LOW);

      digitalWrite(greenLED, HIGH);

    }

    if (request.indexOf("/ledYELLOW=ON") != -1) { // If the request is for
the page "/led=off"

      isLOOP = false;

      digitalWrite(yellowLED, HIGH);

      digitalWrite(redLED, LOW);

      digitalWrite(greenLED, LOW);

    }

    if (request.indexOf("/ledRED=ON") != -1) { // If the request is for
the page "/led=off"

      isLOOP = false;

      digitalWrite(yellowLED, LOW);

      digitalWrite(redLED, HIGH);

      digitalWrite(greenLED, LOW);

    }

    // ** End Adapted Code - This is the end of the code that was adapted
from www.esplearning.com


    // Return the response


    client.println("HTTP/1.1 200 OK");

    client.println("Content-Type: text/html");

    client.println("");

    client.println("<!DOCTYPE HTML>");

    client.println("<html>");
    client.println("<head></head>");
    client.println("<body>");
    client.println("<h1>Stoplight Controller</h1>");
    client.println("<br>");
    client.println("<a href=\ledGREEN=ON><button>Green</button></a><br>");
    client.println("<a 
href=\ledYELLOW=ON><button>Yellow</button></a><br>");
    client.println("<a href=\ledRED=ON><button>Red</button></a><br>");
    client.println("<a href=\led=OFF><button>OFF</button></a><br>");
    client.println("<a href=\loop><button>Loop</button></a>");
    client.println("</body>");
    client.println("</html>");
  }

  //delay(100); // This introduces a little pause in each cycle. Probably 
helps save some power.
*/
}

void onMqttMessage(int messageSize) {
  // we received a message, print out the topic and contents
  Serial.println("Received a message with topic '");
  Serial.print(mqttClient.messageTopic());
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");
  msg = "";
  // use the Stream interface to print the contents
  while (mqttClient.available()) {
    letter = (char)mqttClient.read();
    msg = msg + letter;
    Serial.print(letter);
  }
  Serial.println();
  Serial.print(msg);
  Serial.println();
  if(msg == red){
      state = 0;
      Serial.println(state);
      //digitalWrite(greenLED, LOW);
      //digitalWrite(yellowLED, LOW);
      //digitalWrite(redLED, HIGH);
    }
    else if(msg.equals(yellow)){
      state = 1;
      Serial.println(state);
      //digitalWrite(greenLED, LOW);
      //digitalWrite(yellowLED, HIGH);
      //digitalWrite(redLED, LOW);
    }
    else if(msg.equals(green)){
      state = 2;
      Serial.println(state);
      //digitalWrite(greenLED, HIGH);
      //digitalWrite(yellowLED, LOW);
      //digitalWrite(redLED, LOW);
    }
    else if(msg.equals("blink")){
      Serial.println("in blink loop");
      state = 3;
    }
    else if(msg.equals("1")) {
      Serial.println("in open state");
      isOpen = true;
    }
    else if(msg.equals("0")){
      Serial.println("in closed state");
      isOpen = false;
      digitalWrite(greenLED, LOW);
      digitalWrite(yellowLED, LOW);
      digitalWrite(redLED, LOW);
    }
}
```

Distance Sensor Code:

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


//#include <NewPing.h>  //https://www.makerguides.com/hc-sr04-arduino-
tutorial/


const char* wifiSSID = "Brian's Pixel"; // In order for this to work, you
MUST specify the SSID for your wifi
const char* wifiPSK = "00000007"; // And the preshared key (wifi password)


// Define pins and max distance:
int trigPin = D3;
int echoPin = D4;
//#define MAX_DISTANCE 350 // Maximum distance we want to ping for (in
centimeters). Maximum sensor distance is rated at 400-500cm.
```

```
//NewPing sonar(trigPin, echoPin, MAX_DISTANCE); // NewPing setup of pins
and maximum distance.
long duration;
int distance;


int state = 0;


WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);


const char broker[] = "192.168.18.68";
int port = 1883;
const char topic[]  = "/sensor/distance";


//Last 3 readings for averages
int dis1 = 0;
int dis2 = 0;
int dis3 = 0;
int avgDis = 0;


int    HTTP_PORT   = 80;
String HTTP_METHOD = "GET"; // or "POST"
char   HOST_NAME[] = "192.168.34.141"; // hostname of web server:



WiFiClient client;



void setup() {
  Serial.begin(9600); // Open serial monitor at 9600 baud to see ping
results.


  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  delay(1000);


  // ** Connect to WiFi network - Adapted from
http://www.esp8266learning.com/wemos-webserver-example.php


  Serial.print("Connecting to "); // Display debugging connection info


  Serial.println(wifiSSID); // Print the configured SSID to the Serial
Monitor


  WiFi.begin(wifiSSID, wifiPSK); // Use the provided SSID and PSK to
connect


  while (WiFi.status() != WL_CONNECTED) { // If not connected to wifi


    delay(500); // Pause


    Serial.print("."); // Print a dot each loop while trying to connect


  }


  Serial.println("");


  Serial.println("WiFi connected"); // Print "connected" message to the
Serial Monitor
```

```
  Serial.print("Use this URL : "); // Print the connected IP address to
the Serial Monitor


  Serial.print("http://");


  Serial.print(WiFi.localIP());


  Serial.println("/");


  // ** End Adapted Code - This is the end of the code that was adapted
from www.esplearning.com
  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1);
  }

  Serial.println("You're connected to the MQTT broker!");
  Serial.println();
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should
be the shortest delay between pings.
```

```
  //distance = sonar.ping_in();

  mqttClient.poll();

  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);

  // Trigger the sensor by setting the trigPin high for 10 microseconds:
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin, pulseIn() returns the duration (length of the
pulse) in microseconds:
  duration = pulseIn(echoPin, HIGH);
  // Calculate the distance:
  distance = duration * 0.034 / 2;

  Serial.print("Distance = ");
  Serial.print(distance); // Distance will be 0 when out of set max range.
  Serial.println(" cm");

  if(distance > 150){
    distance = dis1;
  }
  dis3 = dis2;
  dis2 = dis1;
  dis1 = distance;

  avgDis = (dis1 + dis2 + dis3) / 3;
  Serial.print(avgDis);
  Serial.println(" cm");


/*
  if(avgDis >= 13){
    Serial.println("in green if");
    if(client.connect(HOST_NAME, HTTP_PORT)) {
      // if connected:
      Serial.println("Connected to server");
      client.println(HTTP_METHOD + " " + "/ledGREEN=ON" + " HTTP/1.1");
      client.println("Host: " + String(HOST_NAME));
      client.println("Connection: close");
      client.println(); // end HTTP header
    }
  }
  else if(avgDis < 13 && avgDis > 4){
    Serial.println("in yellow if");
    if(client.connect(HOST_NAME, HTTP_PORT)) {
      // if connected:
      Serial.println("Connected to server");
      client.println(HTTP_METHOD + " " + "/ledYELLOW=ON" + " HTTP/1.1");
      client.println("Host: " + String(HOST_NAME));
      client.println("Connection: close");
      client.println(); // end HTTP header
    }
  }
  else if(avgDis < 4){
    Serial.println("in red if");
    if(client.connect(HOST_NAME, HTTP_PORT)) {
      // if connected:
```

```
      Serial.println("Connected to server");
      client.println(HTTP_METHOD + " " + "/ledRED=ON" + " HTTP/1.1");
      client.println("Host: " + String(HOST_NAME));
      client.println("Connection: close");
      client.println(); // end HTTP header
    }
  }
  */
  if(avgDis >= 15 && state != 1){
    Serial.println("in green if");
    mqttClient.beginMessage(topic);
    mqttClient.print("green");
    mqttClient.endMessage();
    state = 1;
  }
  else if(avgDis < 15 && avgDis > 8 && state != 2){
    Serial.println("in yellow if");
    mqttClient.beginMessage(topic);
    mqttClient.print("yellow");
    mqttClient.endMessage();
    state = 2;
  }
  else if(avgDis < 8 && avgDis > 4 && state != 3){
    Serial.println("in red if");
    mqttClient.beginMessage(topic);
    mqttClient.print("red");
    mqttClient.endMessage();
    state = 3;
  }
  else if(avgDis < 4 && state != 4){
    Serial.println("in red blink if");
    mqttClient.beginMessage(topic);
    mqttClient.print("blink");
    mqttClient.endMessage();
    state = 4;
  }


}
```

Magnetic Sensor Code:

```
/*
  Button
  Turns on and off a light emitting diode(LED) connected to digital pin
13,
  when pressing a pushbutton attached to pin 2.
  The circuit:
  - LED attached from pin 13 to ground through 220 ohm resistor
  - pushbutton attached to pin 2 from +5V
  - 10K resistor attached to pin 2 from ground
  - Note: on most Arduinos there is already an LED on the board
    attached to pin 13.
  created 2005
  by DojoDave <http://www.0j0.org>
  modified 30 Aug 2011
  by Tom Igoe
  This example code is in the public domain.
  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Button
*/
#include <ESP8266WiFi.h>
#include <ArduinoMqttClient.h>
```

```
const char* wifiSSID = "Brian's Pixel"; // In order for this to work, you
MUST specify the SSID for your wifi

const char* wifiPSK = "00000007"; // And the preshared key (wifi password)


// constants won't change. They're used here to set pin numbers:

const int buttonPin = D5;     // the number of the pushbutton pin

//const int ledPin =  13;      // the number of the LED pin


WiFiClient wifiClient;

MqttClient mqttClient(wifiClient);


const char broker[] = "192.168.18.68";

int port = 1883;

const char topic[]  = "/sensor/magnet";


// variables will change:

int buttonState = 0;         // variable for reading the pushbutton status

int state = 0;


int count = 0;


void setup() {

  // initialize the LED pin as an output:

  // pinMode(ledPin, OUTPUT);

  // initialize the pushbutton pin as an input:

  pinMode(buttonPin, INPUT_PULLUP);

  Serial.begin(9600);


// ** Connect to WiFi network - Adapted from
http://www.esp8266learning.com/wemos-webserver-example.php
```

```
  Serial.print("Connecting to "); // Display debugging connection info

  Serial.println(wifiSSID); // Print the configured SSID to the Serial
Monitor

  WiFi.begin(wifiSSID, wifiPSK); // Use the provided SSID and PSK to
connect

  while (WiFi.status() != WL_CONNECTED) { // If not connected to wifi

  delay(500); // Pause

  Serial.print("."); // Print a dot each loop while trying to connect

  }

  Serial.println("");

  Serial.println("WiFi connected"); // Print "connected" message to the
Serial Monitor



  Serial.println("Server started");

  Serial.print("Use this URL : "); // Print the connected IP address to
the Serial Monitor

  Serial.print("http://");

  Serial.print(WiFi.localIP());
  Serial.println("/");

  // ** End Adapted Code - This is the end of the code that was adapted
from www.esplearning.com

  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1);
  }

  Serial.println("You're connected to the MQTT broker!");
  Serial.println();
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  // call poll() regularly to allow the library to send MQTT keep alive
which
  // avoids being disconnected by the broker
  mqttClient.poll();

  // check if the pushbutton is pressed. If it is, the buttonState is
HIGH:
  if (buttonState == HIGH && state != 1) {
    // turn LED on:
   // digitalWrite(ledPin, HIGH);
    Serial.println("on");
    mqttClient.beginMessage(topic);
    mqttClient.print(1);
    mqttClient.endMessage();
    state = 1;
  } else if (buttonState == LOW && state != 2) {
    // turn LED off:
   // digitalWrite(ledPin, LOW);
    Serial.println("off");
    mqttClient.beginMessage(topic);
    mqttClient.print(0);
    mqttClient.endMessage();
    state = 2;
  }
  delay(500);
}
```