# Lab 6 – User Interface – Garage Door

## Online Link

This lab is available as part of my online portfolio at: https://github.com/Brianhayden7/Home-Assistant-Google-Assistant

## Objective

The purposes of this lab are to:

- Build a controller to interface with the existing garage door opener. (You don't actually have to connect the relay to a garage door opener but should be able to show that it is performing the necessary action to toggle the garage door opener.)
- Integrate the garage system with at least one external service (IFTTT, Adafruit,io, etc) to introduce some additional functionality of your choice.
- Utilize a non-web interface for the garage door opener

## Materials

I used the following materials to accomplish this lab:

- Personal computer
- 4 x Arduino wemos d1 mini
- 4 x MicroUSB Cable
- 1 x power brick
- 3 x breadboard
- 1 x HC-SR04 Distance Sensor
- 1 x Magnetic Sensor
- 1 x Relay
- 1 x White LED
- 1 x Red LED
- 1 x Yellow LED
- 1 x Green LED
- 4 x 100 Ω Resistor (BrBlBrGold)
- Jumper Wires
- Home Assistant
- Google Assistant

## References

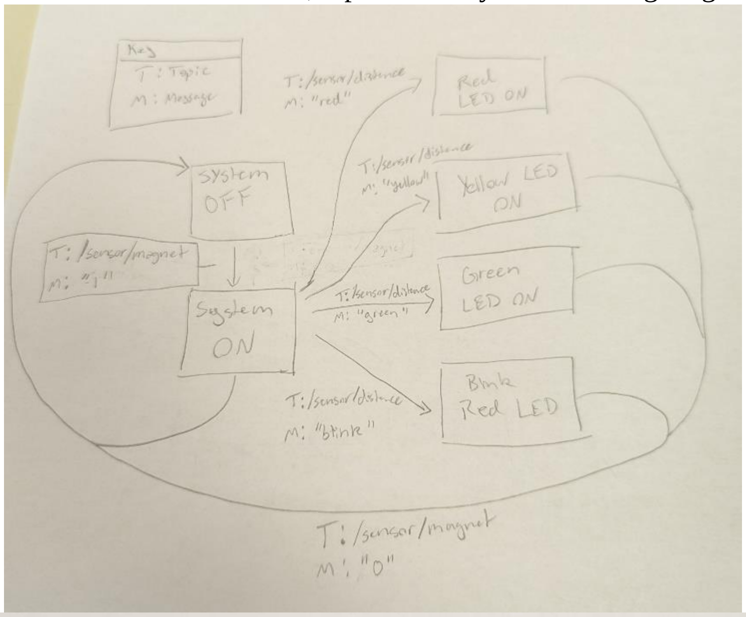
I used the following resources in this lab:

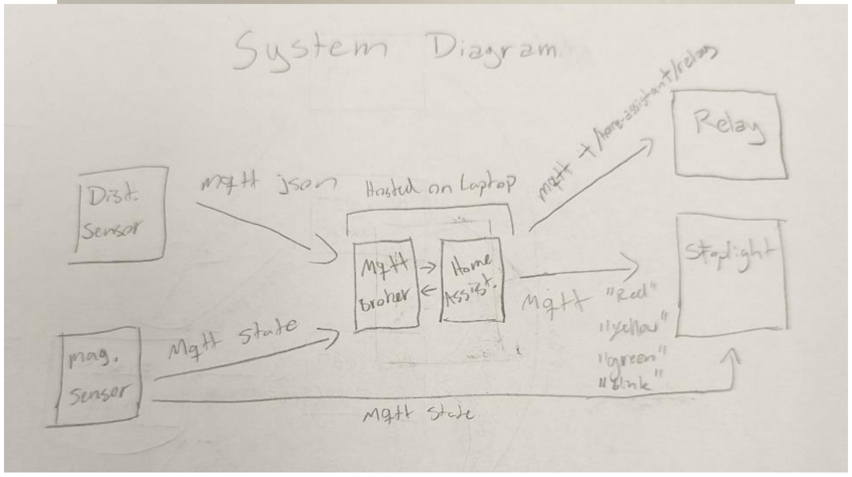
- https://github.com/Brianhayden7/Distance-Sensor My previous led stoplight implementation using the Arduino
- https://create.arduino.cc/projecthub/techvaler/basic-setup-for-arduino-with-relay-e162d5 Tutorial on how to implement a relay
- https://www.home-assistant.io/integrations/google_assistant/ Documentation on Google Assistant implamentation

## 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.

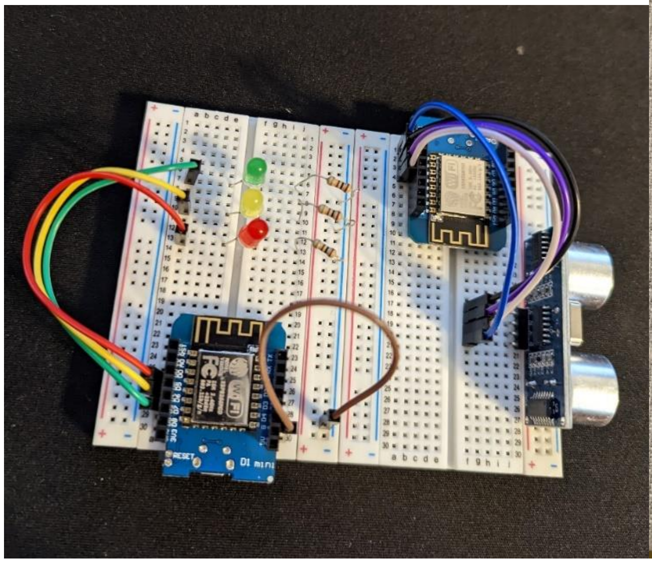
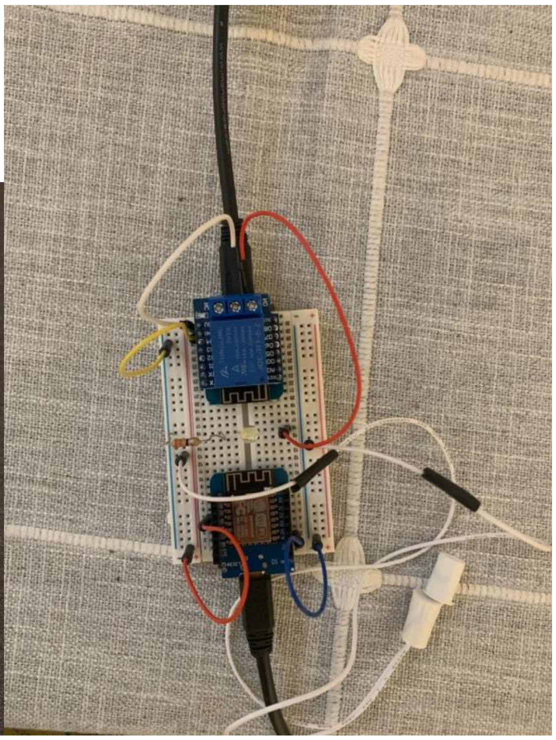
4. Using the ultrasonic Distance sensor guide, configure the logic for continuously pulling and reporting distances to the serial monitor.
5. Then create logical branches depending on 3 different distance ranges.
6. Use these logical branches and the guide for http requests to sent those requests to the web server hosted on the other d1 mini.
7. 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.  You essentially can get to any state from any other state in the diagram so the logic needs to account for this.  On top of that, the second D1 mini will make GET requests depending on the distance the sensor reports.  This is shown by the 3 boxes on the left hand side.

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

8. Program the logic behind your webpage
   a. Set the GPIO pins you want to use to output so you can send them power.
   b. Define each route you will use and make sure the lights do what you want.
   c. Upload code to arduino and it will start to run automatically when powered
   d. See last report in the references on mor information on this.
9. To get the lights to change, dependent on distance
   a. Set up a wifi client that will be used to connect to the wifi server on the second D1 mini.
   b. Set up if statements based on distance and use those 3 conditionals to send http requests depending on which one the distance falls into.

## Client Interactions

The first d1 mini uses routes in order to initiate procedures coded into the arduino. The second D1 mini that has the ultrasonic distance sensor will then use those routes via mqtt in order to change the led color depending on how close something gets to the sensor. The routes are as follows:

Sent from Distance sensor:

- Current distance reading in json object - topic - /sensor/distance

Sent from Magnet Sensor:

- 1 or 0 depending on state – topic - /sensor/magnet

Home Assistant:

Listens:

- Topic - /sensor/#

Sends:

- Blink, red, yellow, green – on topic
- Empty message – on topic - /home-assistant/relay

Home assistant does all the logic with automations. With the home assistant cloud integration as well as a google assistant integration, I could add in google assistant commands to control devices added in home assistant and then run automations on that to add more control.

## Observations

Adding in the relay itself wasn't too hard after I found out how to wire up the relay in series with the LED I was using as my indicator that it was working. I thought this was super cool because it opens a lot of doors for adding automations to things that don't already have them. I love thinking about all the things I could add this new technology to.

Getting google assistant to connect with home assistant is a little tricky. I was going to try to do it without the home assistant cloud integration but there were some complications with security tokens that I didn't want to burn too much time on and the cloud method was fairly simple.

## Thought Questions

1. What services did you consider integrating into your project and why?

I considered using google assistant for this project mainly because it opens up the door for voice control. This makes opening up a garage door while driving hands free which is really nice.

2. What services would you like to integrate in the future?

I would like to learn how to integrate IFTTT into home assistant. Their platform already has a lot of integrations to lots of other services that would make automating everything that much easier.

3. Consider internal and external security threats. Identify 3 likely attack vectors for someone to compromise this system? How did you mitigate these?

The garage door opener is now accessible on the web interface of home assistant. Someone could get in if they found out the login. Luckily you can use strong credentials and not give out the IP to combat this.

There is also another service connected to the system that has access to the devices. It is Google though so I'm not too worried about their security.

Lastly, if someone were to get into the garage, they could mess with the actual wiring of the garage door. Not sure if there's any way to stop this other that build a nice enclosure for it.

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

The biggest challenge for me was figuring out how to use the relay. I probably spent at least an hour just on that trying to figure out how to wire it properly, maybe because I got a bad diagram I was referencing at first. But now that I got it, I understand how it works a lot better now.

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

I spent about 4 hours on the lab and then 1 hour 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/Home-Assistant-Google-Assistant)

**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[] = "192.168.75.68";
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 <ArduinoJson.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 = D1;
int echoPin = D2;
```

```
//#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[] = "192.168.75.68";
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;
  }
  */
  Serial.println("sending json");
  StaticJsonDocument<200> doc;
  doc["distance"] = (String)avgDis;
  serializeJsonPretty(doc, Serial);
  Serial.println("");
  char data[200];
  serializeJson(doc, data);
  mqttClient.beginMessage(topic);
  mqttClient.print(data);
  mqttClient.endMessage();
  delay(200);

}
```

## 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[] = "192.168.75.68";
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);
}
```

## Relay 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)
const int RELAY_PIN = D1;

int mytime = 0;
int redLED = D5;
int yellowLED = D6;
int greenLED = D7;
bool isLOOP = false;
bool firstLoop = true;
bool isOpen = false;
bool pulse = false;
int state = -1;
```

```
String msg;
char letter;
String red = "red";
String yellow = "yellow";
String green = "green";
unsigned long previousMillis = 0;
const long interval = 500;
int ledState = LOW;



WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

const char broker[] = "192.168.75.68";
int port = 1883;
const char topic[]  = "/home-assistant/relay";

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(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, 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 (currentMillis - previousMillis >= interval && pulse) {
    // 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(RELAY_PIN, ledState);
    if(ledState == LOW){
      pulse = false;
    }
  }
}

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();
  pulse = true;
}
```