Lab 5 – Smart Home Controller

Online Link

This lab is available as part of my online portfolio at: https://github.com/Brianhayden7/home-assistant

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
- Virtual box running home assistant OS
- 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

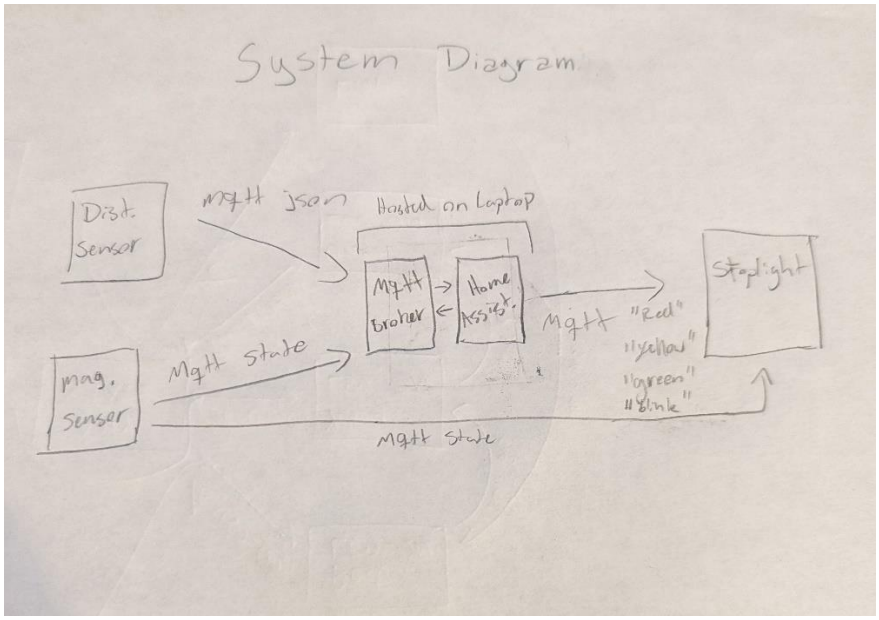
References

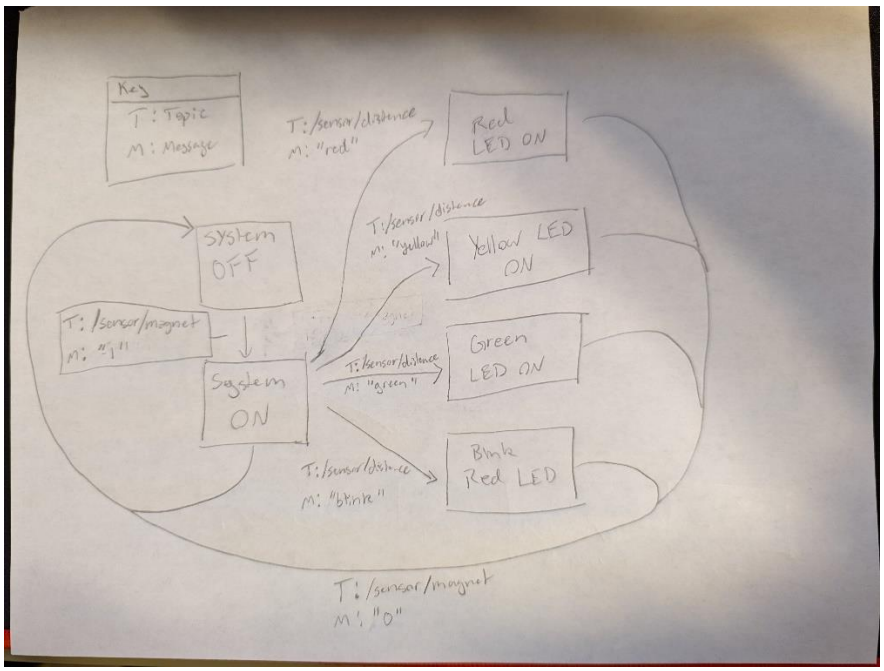
I used the following resources in this lab:

- https://github.com/Brianhayden7/Arduino-MQTT-Hub My previous lab implementation of stoplight and distance sensor system using mqtt
- 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://www.tutorialspoint.com/arduinojson-serialize-and-deserialize How to use json on arduino
- 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
- https://www.home-assistant.io/installation/windows Install walkthrough of Home Assistant on windows vm in virtual box
- https://www.home-assistant.io/integrations/sensor.mqtt How to create an mqtt sensor in home assistant

Procedures:

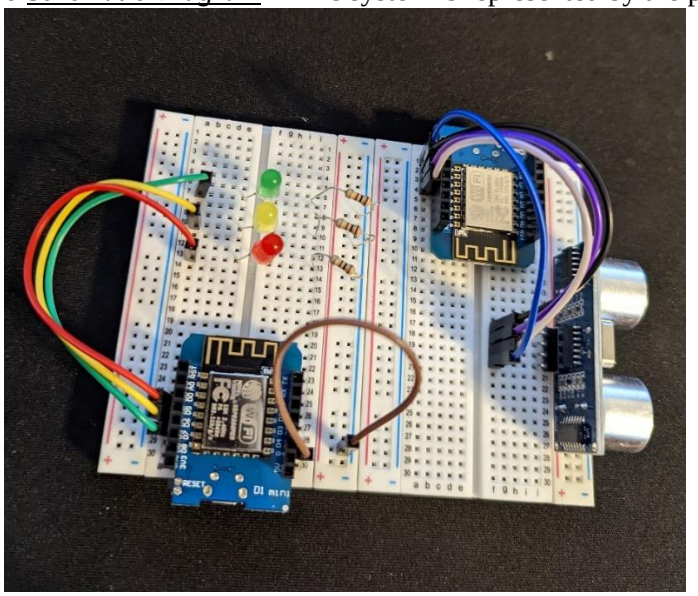
1. Download and install the arduino IDE.
2. Search for the library that includes the esp8266 boards.
3. Configure the arduino IDE to use the wemos d1 mini and select the correct port.
4. Using the magnetic sensor, set up the logic for when it is open or closed.
5. Install MQTT broker on laptop or pc
6. Install home assistant on virtual box. I had to use bridged network adapter. Under advanced options I changed promiscuous mode to allow all. This allows it to pass through on wifi I believe.
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. Set up home assistant.
10. Change distance sensor to send json object with distance readings over mqtt.
11. Display distance in home assistant.
12. Use home assistant to handle distance logic and publish mqtt messages for the stoplight accordingly.
13. Model the functionality, logical flow and components of the system. Include a schematic diagram.
    a. ***Functionality*** - This system is a basic system diagram and state machine, represented by the following diagrams:



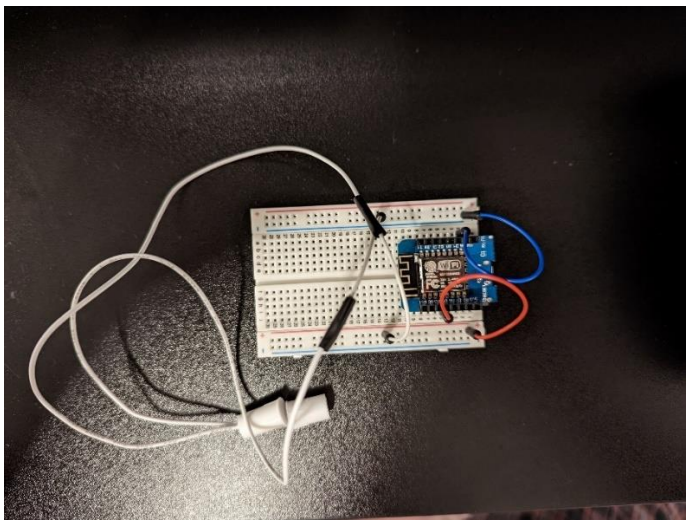


- Upon startup, it will start it the off position.

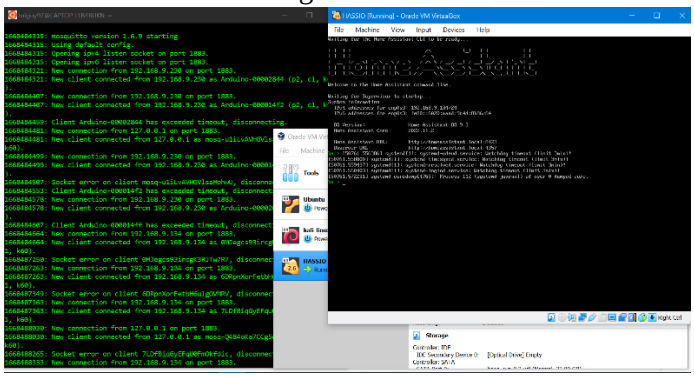
b. *Component Diagram* and *Schematic Diagram* – This system is represented by the photos below.



Stoplight (left) and distance sensor (right)



Magnet Sensor



Laptop running mosquitto broker and home assistant in virtual box

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” (sent from magnet sensor d1)
- Mosquitto_pub -t /sensor/magnet -m “1” (sent from magnet sensor d1)
- Mosquitto_pub -t /sensor/distance -m ‘json object with distance value’ (captured by home assistant)
- Mosquitto_pub -t /sensor/distance -m “red” (sent from home assistant automation)
- Mosquitto_pub -t /sensor/distance -m “yellow” (sent from home assistant automation)
- Mosquitto_pub -t /sensor/distance -m “green” (sent from home assistant automation)
- Mosquitto_pub -t /sensor/distance -m “blink” (sent from home assistant automation)
- Mosquitto_sub -t /sensor/# *this subscribes the stoplight to all sensor messages (captured by stoplight d1)

Observations

Home assistant definitely has a lot of options. At first glance it is kind of intimidating but learning it little by little helps the process a bit. There is also tons of documentation so finding answers isn’t too hard plus there’s a pretty big community around the platform. The only difficult thing is understanding the documentation sometimes.

I like how you can abstract out all the logic from the devices if you want to and run all the logic from within home assistant. The automations let you do any logic that you would otherwise need to do which is nice.

One thing I didn't dive too deep into was trying to run the mqtt broker that you can add onto home assistant. From what I read it wouldn't allow unauthenticated connections, so I decided to continue with the mosquitto broker being run on wsl.

It's cool how you can visualize the whole system from within home assistant. There is a lot you can customize and see and I'm excited to learn more about how to really make it my own.

Thought Questions

1. Which version of Home Assistant did you choose to install? (Home Assistant Operating System, Home Assistant Container or one of the more experienced versions) Why did you choose this version?

I installed home assistant operating system onto a vm using virtual box on my laptop. I wanted to do it this was to make it easier to manage, spin up, and shut down easy and quick. It reduces the number of physical devices I need to be powering as well.

2. How should you decide which logic to perform in Home Assistant versus coding the logic directly into the devices? What guiding principles would you establish for future devices?

I think that almost all logic should be done in home assistant so that sensors can be used for other systems if necessary. This means that a device that needs commands, light the stoplight, should only need one command per state that it has. You should only need to send 4 different commands to the stoplight for it to function and it just needs to interpret those as far as logic goes.

3. What features do you like the most about Home Assistant?

I like the visualizations a lot. I saw a video where they did a lot of custom cards that can change colors and everything depending on device states and I would love to be able to jump more into those. I also really like the automations because they really open up all the doors for making whatever systems you want.

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

I spent about 6 hours on the system and 2 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)

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 <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[] = "192.168.9.68";

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[] = "192.168.18.68";
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);
}
```