Lab 3 – Distance Sensor

Online Link
This lab is available as part of my online portfolio at: https://github.com/Brianhayden7/Distance-Sensor

Objective
The purposes of this lab are to:

· Build an IoT sensor using GPIO pins for inputs

· Establish a machine-to-machine (M2M) communication protocol

· Design an IoT interaction between a sensor and an actuator

Materials
I used the following materials to accomplish this lab:

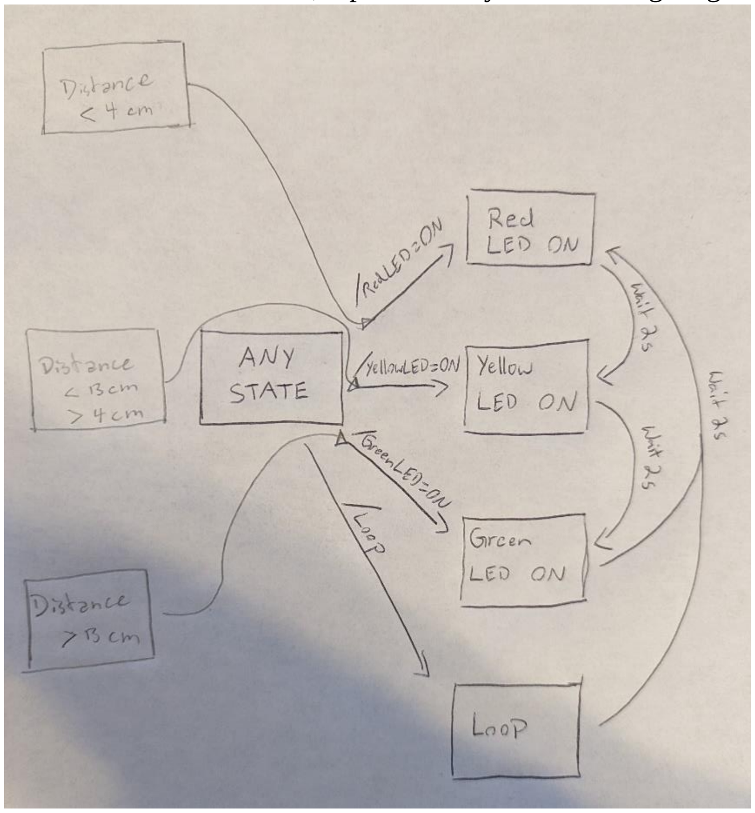
- Personal computer
- 2 x Arduino wemos d1 mini
- 2 x MicroUSB Cable
- 1 x power brick
- 1 x breadboard
- 1 x HC-SR04 Distance Sensor
- 1 x Red LED
- 1 x Yellow LED
- 1 x Green LED
- 3 x 100 Ω Resistor (BrBlBrGold)
- Jumper Wires

References
I used the following resources in this lab:

- https://github.com/Brianhayden7/arduino-led-stoplight My previous led stoplight implementation using the Arduino
- https://arduinogetstarted.com/tutorials/arduino-http-request Guide on How to make HTTP GET requests on the Arduino
- https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6 Guide on using the ultrasonic distance sensor

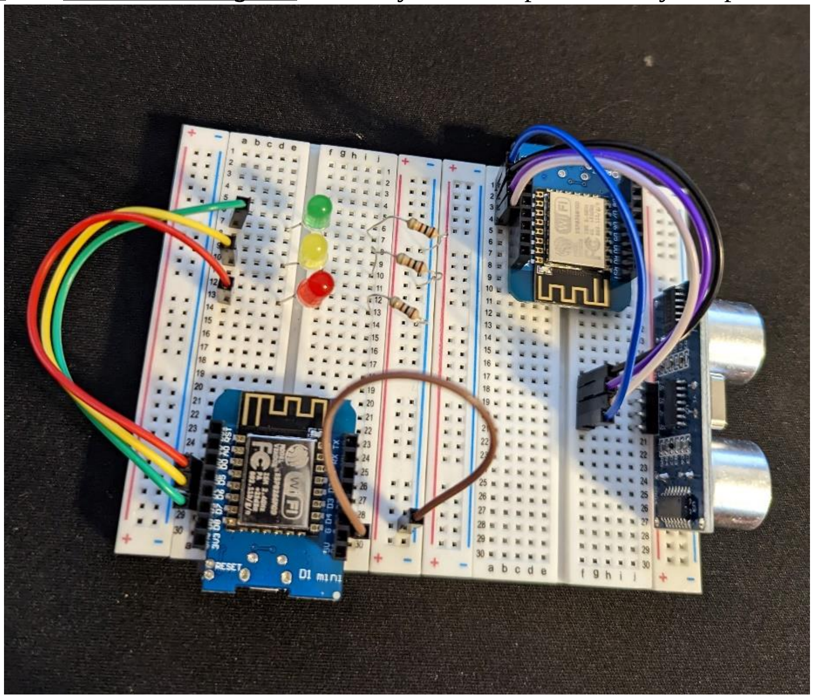
Procedures:
1. Download and install the arduino IDE.
2. Search for the library that includes the esp8266 boards.
3. Configure the arduino IDE to use the wemos d1 mini and select the correct port.
4. Using the ultrasonic Distance sensor guide, configure the logic for continuously pulling and reporting 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 get requests in order to change the led color depending on how close something gets to the sensor. The routes are as follows:

/redLED=ON at less than 4 cm

/yellowLED=ON between 4 and 13 cm

/greenLED=ON further than 13 cm

Observations

Luckily most of this lab was already done since nothing needed to be changed with the web server and stoplight code on the first D1 mini. The web paths were already set up as well, so all the work needed done with the distance sensor and making get requests.

I had some issues with uploading code to the D1 mini while the distance sensor was connected, and I couldn’t find out why for a while. Turns out the issue was connecting to the serial port while the 5v connector was attached to the sensor. It was mostly smooth sailing after that.

I had some difficulty figuring out how to properly format and make the http get requests from one machine to the other, but I found a good general guide that ended up working mostly right. I just coded in the paths for each one and that was basically it.

Overall, I thought that this wasn’t that bad of a lab. Most of my issues stemmed from me making dumb mistakes and just being able to troubleshoot those. I really enjoyed being able to connect two devices over wifi though since that opens up a lot of doors in the future which is super cool.

Thought Questions

1. Think of the interaction between your devices. How well would this scale to multiple devices? Is it easy to add another sensor? Another actuator?

This would not scale very well. Currently the one sensor and second device are sending so many requests, adding more devices that are all trying to connect with the one D1 mini on the same port and have to parse all those requests, it would get overwhelmed and not be able to handle all the requests.

2. What are strengths and weaknesses of the tennis-ball-on-a-string system that Don had originally? What are strengths and weaknesses of the IoT system that he developed? What enhancements would you suggest?

The tennis ball on the string is reliable and as long as you see it there, it is always working. Unfortunately, it doesn’t give any kind of feedback until you hit it. The light system is much better at giving you feedback on how close you are getting to the parking spot. Technology can run into bugs though and need a reset so there are times when maybe it isn't working like it should. One improvement I can think of is adding in some kind of audio feedback as well to the system to help for when you get too close. Pretty cool system though.

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

Making sure all the connections to wifi and also to each other using the web server was the hardest part here. Getting the distance sensor working was actually easier than I thought it would be. I had some trouble getting the http request formatted correctly so that the web server both recognized it and also recognized the correct routes I was trying to pass it. Minor spelling errors really messed me up. I used a '\' at first instead of '/' and that kept it from working for a while until I fixed it.

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

I probably spent about 3 hours on the coding up of the lab and another 2 hours on the report.

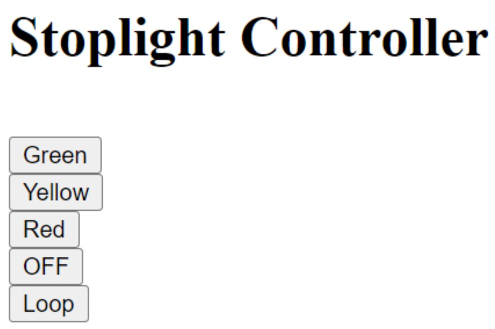
Certification of Work

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: System Interface - Web Page



Appendix 2: Arduino Code

(available at: https://github.com/Brianhayden7/Distance-Sensor)

First D1 Mini Code:

```
#include <ESP8266WiFi.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;


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
}


void loop() {
  // put your main code here, to run repeatedly:
  WiFiClient client = server.available(); // Create a new client object for available connections


  int myTime = millis();
  //if(firstLoop){
  //  myTime = 0;
  //}
  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.


}
```

Second D1 Code:

```
#include <ESP8266WiFi.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;

//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
}

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();

  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
    }
  }

}
```