Lab 3 Report
Arduino Ultrasonic distance-controlled Stoplight

Online Link:
This lab is available as part of my online portfolio at: https://github.com/DemonPiranha/ITC-441-labs.git

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:

- 2 x Arduino Wemos D1 Mini with ESP8266
- 2 x MicroUSB Cable with power source
- Wifi device, this lab uses a smartphone with a wifi hotspot
- 1 x SR04 Ultrasonic sensor
- 2 x breadboards
- 1 x Green LED
- 1 x Yellow LED
- 1 x Red LED
- 3 x 100 Ω Resistor (BrBlBrGl)
- 8 x Jumper Wires

References
I used the following resources in this lab:

- This lab assumes the hardware and software setup of the stoplight from the previous lab with slight modifications that will be explained.
- https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html - ESP8266 Wifi library documentation. This is where most of the wifi control code comes from.
- https://randomnerdtutorials.com/esp8266-nodemcu-http-get-post-arduino/ - Documentation that explains sending HTTP requests to an IP address.
- https://randomnerdtutorials.com/esp8266-nodemcu-static-fixed-ip-address-arduino/ - Documentation on how to set static IP addresses on Arduino.
- https://github.com/gamegine/HCSR04-ultrasonic-sensor-lib - Ultrasonic sensor library used to get distance value from the sensor.
- https://app.diagrams.net/ - Chart maker used to make the state diagram.
- https://1drv.ms/w/s!Ais81h0TsK5NhqEUuevEs8zMavzLag?e=hdaL2m - Todd Berrett's example write-up. Used for formatting.

Procedures:

1. Configure the Arduino IDE to interpret ESP8266 boards. Follow this guide to do that: https://arduino-esp8266.readthedocs.io/en/latest/installing.html
2. For the first Ardunio with the stoplight components use Lab3-1.ino code. For the second Arduino with the ultrasonic sensor use Lab3-2.ino code. Then follow the steps below.
3. Copy the Ardunio code in this project into an Arduino sketch. Adjust the global variables for SSID and password to match the wifi network you will be using.
4. Make sure the Arduino is properly connected to the computer with the computer via a USB cable.
5. Upload the code. Once the upload has completed the built-in Arduino LED should now be rapidly flashing. (This indicates awaiting connection to wifi)
6. Once a connection is established the built-in LED will stop blinking
7. Distance from the sensor will now be translated into color LED status.

System view

This project includes two primary systems that the user interacts with.

Stoplight module

This part includes the Ardunio and the three LEDs to model a stop light. The user is able to receive visual feedback from these lights determined by the ultrasonic distance module..

Ultrasonic distance module

This part includes the second Ardunio and an SR04 ultrasonic distance sensor. The module as a whole receives distance readings from the sensor and sends HTTP requests for the various stoplight colors depending on the distance from the sensor. With green being beyond 55cm, yellow between 50cm and 22cm, red between 20cm and 12cm, and flashing yellow and red warning below 10cm.

Component View

Model of the functionality, logical flow, and components of the system. Diagram provided in Appendix.

a. Functionality – This system consists of setup and primary loop, seen in Appendix 1:
    i. Setup:
        1. The system configures built-in outputs.
        2. Wifi system is enabled and the connection process begins. This sub-process will loop until a connection is established. The program is then able to continue.
        3. The primary loop then begins
    ii. Primary Loop:
        1. This loops primary job is to listen to check the distance from the sensor and perform actions based on this.
        2. If distance from sensor is beyond 55cm:
            a. Send http request of "/green" to the set ip address of the server
            b. If the light has already been green then no http request will be sent
        3. If distance from sensor is between 50cm and 22cm:
            a. Send http request of "/yellow" to the set ip address of the server
            b. If the light has already been yellow then no http request will be sent
        4. If distance from sensor is between 20cm and 12cm:
            a. Send http request of "/red" to the set ip address of the server
            b. If the light has already been red then no http request will be sent
        5. If distance from sensor is below 10cm:

a. Send http request of "/tooClose" to the set ip address of the server
b. This triggers the stoplight to start flashing red and yellow lights
c. If the light has already been tooClose then no http request will be sent

SR04 Distance Sensor Library
For ease of programming I used a SR04 library to provide the distance output from the SR04 sensor. This library essentially sends a pulse out on the trigger pin and reads the echo pin that comes back. The pulse is set for 10 microseconds and then stopped and read. This value is eventually returned so that it can be used in my program. Because I use this library some lag could be induced however it is within desired parameters for this application.

Distance Checking
I check for the distances with small gaps between each one to provide a bit of wiggle room between the distance points. This helps to reduce the twitching received from the sensor and smooths out the readings around the change-over points better. This induces some inaccuracies however and finding a better solution that addressed the twitching of the values would be better.

HTTP Requests
Using the ESP8266HTTPClient library an http client can be created which allows the arduino to send standard http requests to a set address. A wifi client is also configured so that the http request are sent through the wifi module. In my program I use http.GET request that doesn't have any response required from the server. This means there isn't any back and fourth communication however this functionality is not needed. The server is then able to read these requests like any other and performs it's set actions depending on those requests. Another feature I built is that of redundant http requests not being sent. If a color has already been sent then it will no longer send any more requests while the distance remains within the set parameter. Once the distance goes past this parameter a new http request for a different color or command may be sent and then the process repeats. This reduces unnecessary http requests being sent unless a legitimate update occurs.

Observations

This lab was one of there first where I felt I had grasped the concepts of Ardunio and was comfortable working my way around. I was quickly able to have code running on the sensor module returning distances. Several google searches later I was able to get HTTP requests set up creating the communication from the sensor to the stoplight.

I was really happy with how this all came together and how I understood how everything was interacting with eachother.

Thought Questions

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

In this lab, the client sensor device communicates with the server stoplight device via HTTP requests. Due to this any other client device could be added that would send those same HTTP requests. Adding another device that could function as an actuator would require configuring it as a server as well. This could get complicated if more devices are added. It would be better to switch to a more lightweight communication protocol to scale effectively. The advantage of this system is that anything that can send an HTTP request could send commands to the server module which allows for multiple different clients.

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?

Strengths are that it doesn't require power of any kind eliminating long-term costs. The tennis ball can be extremely accurate for the exact park location desired as the point of contact is the right spot. A weakness of this however is that there is not accuracy with distance before and after the correct spot. Another weakness is that the tennis ball only perfectly works for one vehicle. Any other vehicle that has different dimensions will no longer be able to use the tennis ball for an accurate parking location.

This is where the IoT system comes at an advantage. Since the warnings are based on pure distance any vehicle can park and do so within the same distance of any other. You are also able to know generally how far away the vehicle is while coming into the parking spot providing a user the feedback they need to be more careful as they get closer and faster when they are further away. The IoT device also has tooClose warning that informs the user when the vehicle is past the perfect spot.

One main enhancement I would make is to connect the device to an LED strip that would light up from the bottom to top with progressive colors from green, yellow, to red to provide a user even more precise feedback. Another enhancement could be to add a sound warning when you have reached the correct distance to provide another sensory warning. Finally, I would change the sensor for a laser distance sensor to get more accurate readings.

What was the biggest challenge you overcame in this lab?

One of the biggest difficulties for me was figuring out the tolerances of the sensor and finding the sweet spot to avoid changing the light too quickly with an erroneous distance reported. I then also had to figure out a way to reduce the HTTP request that needed to be sent so they were only sent when an update of the light needed to happen.

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

I spent about 8 hours on the project with 3 hours on this lab report.

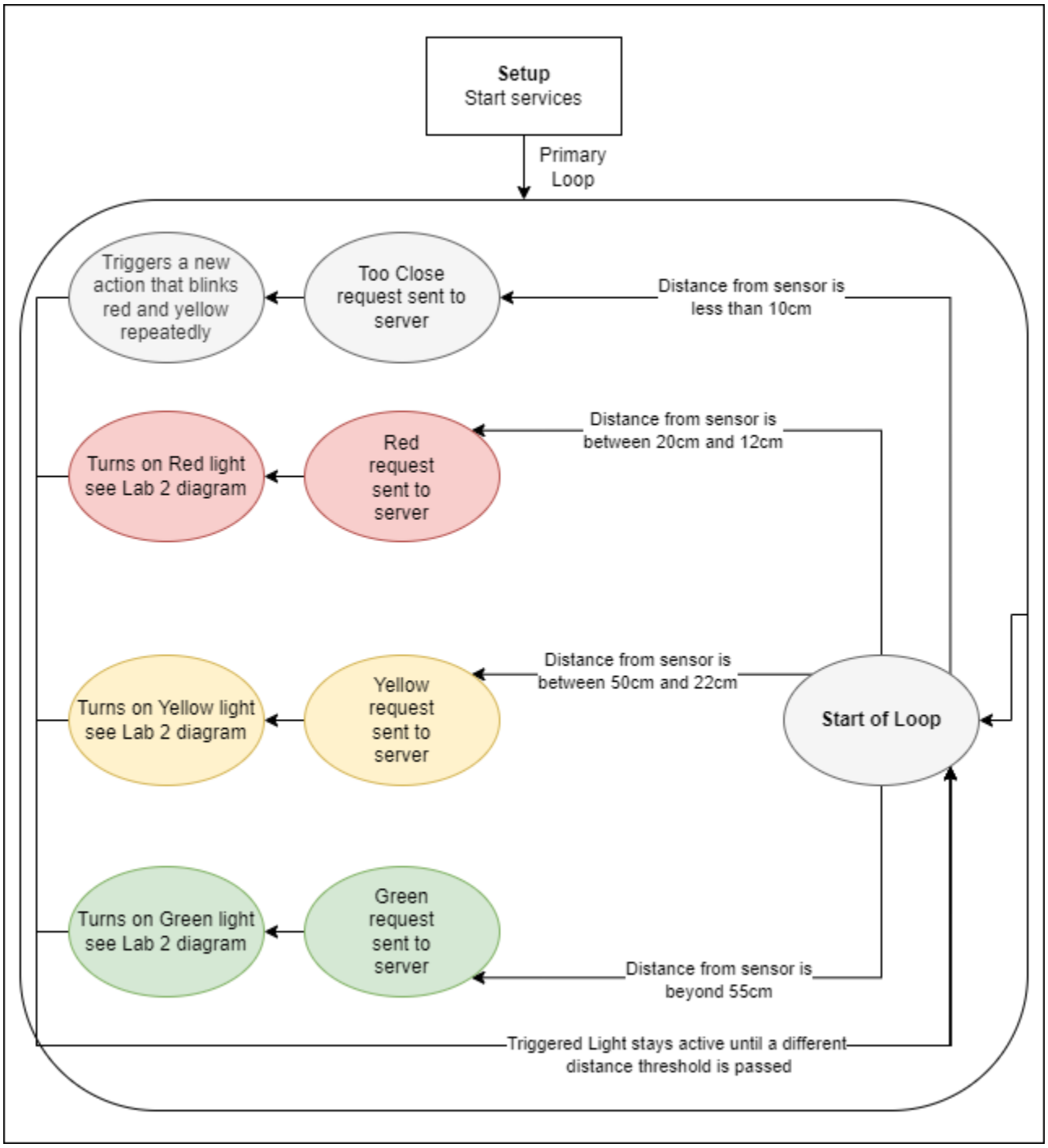
Certification of Work

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.

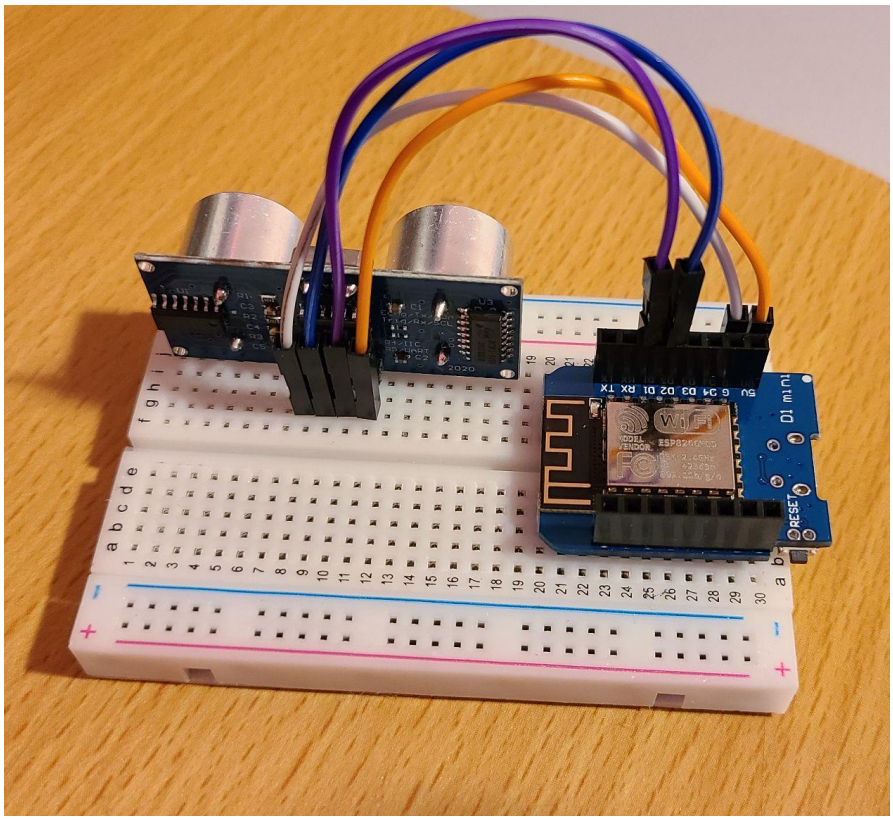
--Blake Porter

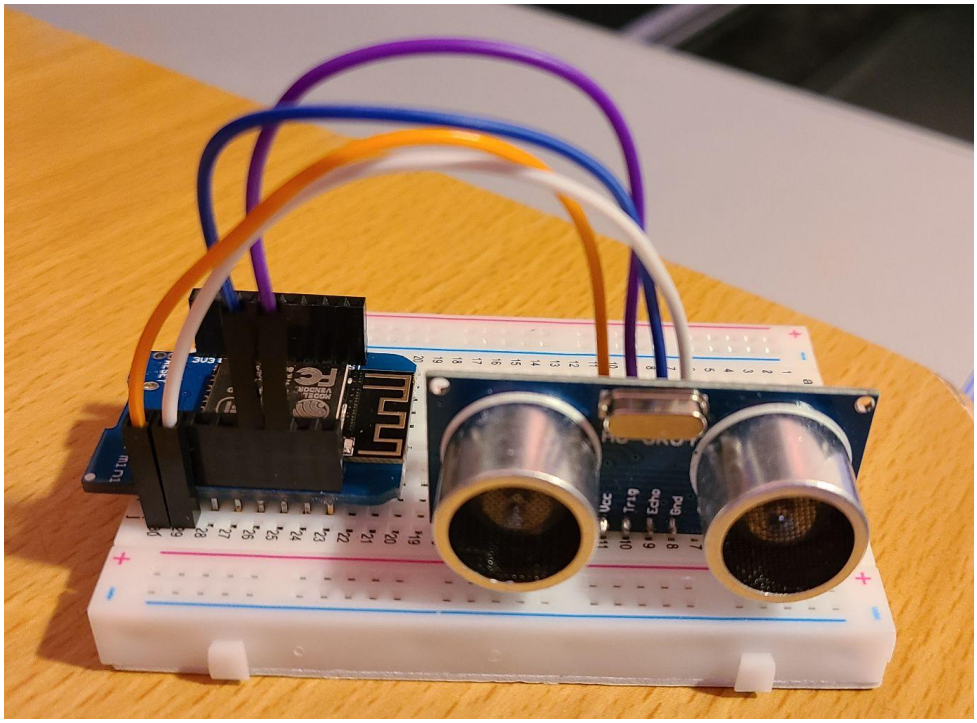
Appendix

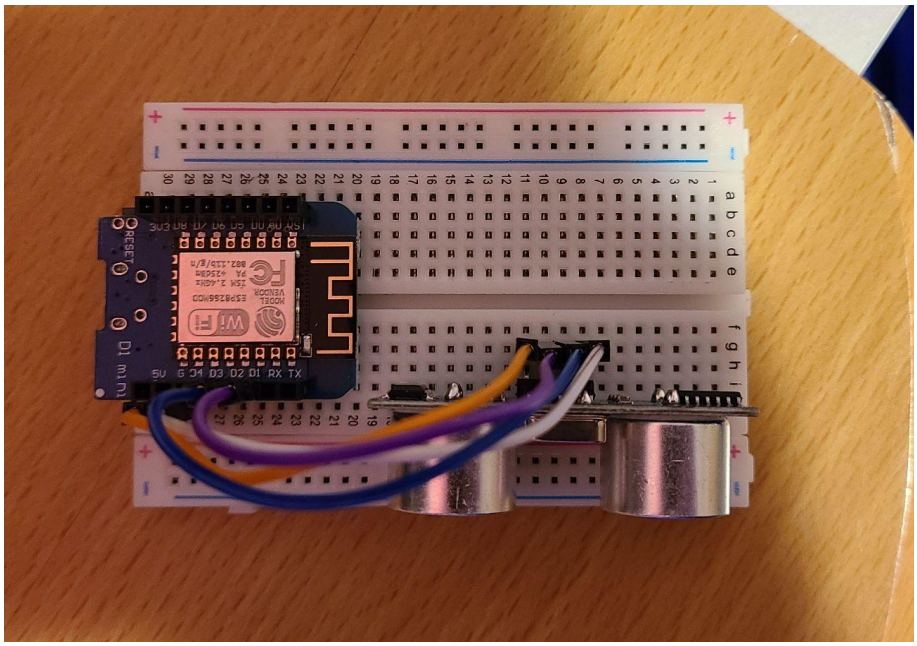
Appendix 1: State Diagram



Appendix 2: Physical Components Layout of Ultrasonic sensor assembly






RESET
D1 mini
5V G D4 D3 D2 D1 RX TX

Appendix 3: Ardunio Code

(available at https://github.com/DemonPiranha/ITC-441-labs/tree/main/Lab%203)

Ultrasonic sensor code:

```
#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include "HCSR04.h" // include the Ultrasonic sensor library "HCSR04 ultrasonic sensor by gamegine v2.0.3"


//Sets some global variables

const char* ssid = "Blake-Hotspot";

const char* password = "blakepublic";

WiFiServer server(80);

IPAddress local_IP(192, 168, 91, 185);

IPAddress dns(192, 168, 91, 254);

IPAddress gateway(192, 168, 91, 254);

IPAddress subnet(255, 255, 255, 0);

String serverName = "http://192.168.91.184";   // IP address path of server device

long ultraDistance;

int change = 0;

HCSR04 ultraSonic(D1, D2);  //Class for HCSR04 sets (trig pin , echo pin)



//Setup for pins and clears all of them

void setup() {


  //Setup serial and all LED outputs then clears the LEDs

  Serial.begin(9600);

  pinMode(LED_BUILTIN, OUTPUT);
```

```
  //Sets the static IP to the set variables
  if (!WiFi.config(local_IP, dns, gateway, subnet)) {
    Serial.println("STA Failed to configure");
  }


  //Using the Wifi library, begin connecting to the set network, start webserver and provide status updates.
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(100);
    Serial.print(".");
    digitalWrite(LED_BUILTIN, LOW);
    delay(100);
  }
  delay(100);
  digitalWrite(LED_BUILTIN, HIGH);
  Serial.println();
  Serial.println("(( WiFi connected ))");
  Serial.println(WiFi.localIP());
  Serial.println(WiFi.gatewayIP());
  Serial.println(WiFi.dnsIP());


  server.begin();
  Serial.println("((( Server Started )))");
```

```
}


void loop() {


  // Sets up a client object that sets this device as a client. http is also set as a HTTPClient
  WiFiClient client;
  HTTPClient http;


  //Serial.println(change);
  Serial.print(ultraDistance);
  Serial.println(" cm");
  int httpResponseCode;


  //each time the loop repeats ultraDistance will be set to the distance received from the sensor
  ultraDistance = ultraSonic.dist();
  delay(60);


  //If the distance is farther than 55cm
  if (ultraDistance > 55) {
    //If the distance is still within the range then nothing will happen so that http requests are not sent unnecessarily.
    if(change == 1){
    }
    else {
      Serial.print(ultraDistance);
      Serial.println(" cm");
```

```
      http.begin(client, serverName + "/green");
      httpResponseCode = http.GET();
      change = 1;
    }
  }
  //If the distance is between 50cm and 22cm
  else if (ultraDistance <= 50 && ultraDistance >= 22) {
    //If the distance is still within the range then nothing will happen so that http
requests are not sent unnecessarily.
    if(change == 2){
    }
    else {
      Serial.print(ultraDistance);
      Serial.println(" cm");
      http.begin(client, serverName + "/yellow");
      httpResponseCode = http.GET();
      change = 2;
    }
  }
  //If the distance is between 20cm and 12cm
  else if (ultraDistance <= 20 && ultraDistance >= 12) {
    //If the distance is still within the range then nothing will happen so that http
requests are not sent unnecessarily.
    if(change == 3){
    }
    else {
      Serial.print(ultraDistance);
      Serial.println(" cm");
```

```
      http.begin(client, serverName + "/red");
      httpResponseCode = http.GET();
      change = 3;
    }
  }
  //If the distance is closer than 10 cm
  else if (ultraDistance <= 10) {
    //If the distance is still within the range then nothing will happen so that http requests are not sent unnecessarily.
    if(change == 4){
    }
    else {
      http.begin(client, serverName + "/tooClose");
      httpResponseCode = http.GET();
      change = 4;
    }
  }

}
```

Stoplight Arduino code additions:

```
//Variable to be used later in program
IPAddress local_IP(192, 168, 91, 184);
IPAddress dns(192, 168, 91, 254);
IPAddress gateway(192, 168, 91, 254);
IPAddress subnet(255, 255, 0, 0);
bool tooClose = false;



void setup() {

//Takes the set values for IP configuration and sets them to the arduino before connection to wifi occurs. This effectively sets a static IP so that other devices can communicate to a known address.
  if (!WiFi.config(local_IP, dns, gateway, subnet)) {
    Serial.println("STA Failed to configure");
  }
}

void loop() {

// This is another web request check for a new command that will trigger the too close warning
  if (req.indexOf("/tooClose") != -1) {
    cycle = false;
    tooClose = true;
  }

  if (tooClose == true){
    yellowLight();
    delay(150);
    redLight();
    delay(150);
  }
}
```