# Lab 3 - Stoplight Controlled With HC-SR04 Ultrasonic Distance Sensor

#### Online Link

The details and code to this lab can be found at: <https://github.com/Rhemsley/Distance_Stoplight>

#### Objective

The purpose of this lab is to learn how to work with the HC-SR04 Ultrasonic Distance Sensor and then use that to control LEDs of another system based on the distance measured. This is accomplished as follows:

* 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 stoplight arduino found at with the addition of a Blink page as explained below. <https://github.com/Rhemsley/Arduino-Stoplight>. I also used the following materials to create the needed distance sensor system:

1 x D1 mini

4 x Male to Male Jumper Cables

1 x HC-SR04 Ultrasonic Distance Sensor

1 x Breadboard

#### References

I used the following resources to complete this lab:

<https://github.com/Rhemsley/Distance_Stoplight> Referencing for my Stoplight

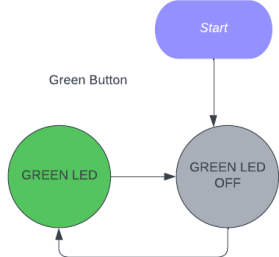
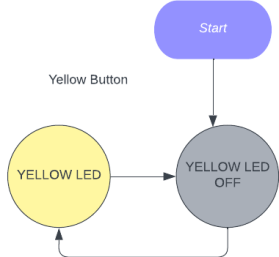
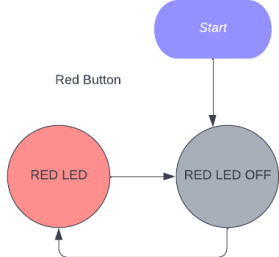
<https://arduinogetstarted.com/tutorials/arduino-http-request> Making HTTP Get Requests

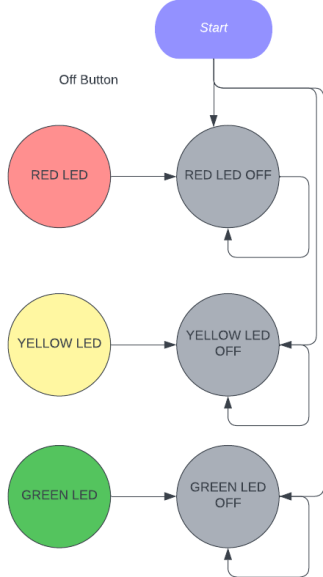
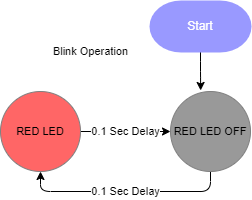
<https://esp8266-shop.com/blog/how-to-http-get-and-post-requests-with-esp8266/> Additional learning on HTTP Get Requests to maybe reduce the delay

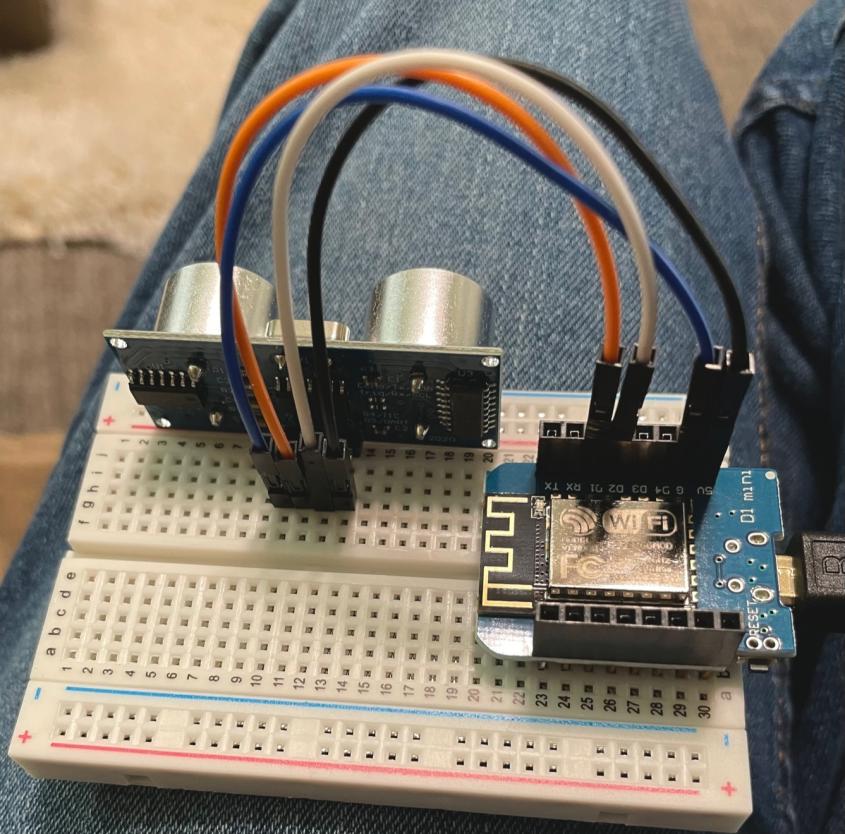
<https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/> HC-SR04 Distance sensor guide and implementation help

#### Baseline Information

Below are the Flow charts for the five different Web Server buttons or /Page Get requests that control the LEDs. As seen each of the individual LED buttons cycle the LED between on and off. The Off button turns off all three of the LEDs. Lastly, instead of the Loop operation, this distance sensor desired a blinking operation when too close. This was done with the Red LED as seen from the diagram.



To the right we have the view of the build with the HC-SR04 Distance Sensor plugged into the bread board and from this view from left to right of Ground (Blue), Echo (Orange), Trig (White), and Voltage(Black) pins. Those four wires are respectively going into the D1, D2, Ground, and Voltage pin slots on the D1-mini.

#### Procedures

I will assume most if not all of the steps given in the above Arduino-Stoplight Git repository have already been followed and I will work off of that baseline knowledge to write these Procedures. The main thing to note though is that to get the blink operation working for this project, make sure you are using the updated Stoplight code found within this projects Git repository to allow the /Blink Get request to work.

1. Modify code to match your Pin and WiFi setup.
   1. In the Arduino IDE, get all the code copied in and then in both of the arduino codes make sure your Pins are labeled correctly and the WiFi SSID and password is setup correctly as explained on step 3 of the Stoplight procedures.
   2. On top of this, in the Distance sensor code, you will need to set the “HOST\_NAME[]” to the IP address of the Stoplight server. This will likely mean running the stoplight server while connected to the laptop and grabbing the IP address it gives once connected to the server. If the correct IP address is not set here, the distance sensor will not be able to connect to the Stoplight.
2. Verify and then Upload code.
   1. Now verify and then upload the code to each of the D1 minis, making sure the correct D1 minis are receiving the correct code.
   2. I would recommend first uploading the Stoplight code and then watching the Serial Monitor to see what the IP address of the Stoplight is and record that somewhere.
   3. Then unplug the Stoplight from the computer, plug in the ultraviolet sensor board, modify the “HOST\_NAME[]” to the right IP address, and then upload.
3. Watch Serial Monitor to make sure the Distance Sensor is measuring properly
   1. Once the Distance Sensor board has connected to WiFi, it should start giving out the average distance measured over the last 5 measurements within the Serial Monitor.
   2. These sensors can be a bit touchy and so measuring distances to flat objects will give much more consistent results.
4. Start exploring the different distances to get your different LED lights.
   1. If you would rather use Centimeters than Inches, the code can easily be modified to show this instead by finding the “distance1 = duration \* 0.034 / 2 \* 0.393701;” line and then deleting the “\* 0.393701”. I would also recommend updating the “Serial.print("Average Distance (inch): ");” line to say “(cm)” instead of “(inch)”.
   2. Currently the Green LED should light up between Average distances of 120 inches to 20 inches.
   3. The Yellow LED should light up between Average distances of 20 inches to 12 inches.
   4. The Red LED should light up between Average distances of 12 inches to 5 inches.
   5. And the Red LED should start blinking when the Average distance is 5 inches or less.
   6. Any of the above distances can be modified by modifying the numbers within if/else if statements at the bottom of the code. Do note that both the bottom range of one distance and the top range of the following statement need to match.
5. Like the standard Stoplight instructions explained, these lights are changing through hitting the servers endpoints with Get/Post requests as listed below.
   1. [IP]/Red - Alternates the RED LED between Off and On
   2. [IP]/Yellow - Alternates the Yellow LED between Off and On
   3. [IP]/Green - Alternates the Green LED between Off and On
   4. [IP]/Off - Turns off all of the LEDs and stops the loop if looping
   5. [IP]/Blink - Turns off all the LEDs and then starts the Blink Red cycle

#### Observations

Wow, the power that comes from combining two Arduino ran devices or more particularly combining a sensor system with a working Web Server and LED system. Figuring out how to get one to connect to the other successfully and then send the desired requests of information truly was an adventure.

The first hump I encountered was figuring out how to get one device to connect to another. Thankfully, there are many good resources online on how to get an Arduino to make a client and connect to something. After some failed attempts and learning, I got it working and was able to get it to hit an end point and change an LED status. The extra tough and weird thing I ran into with this was the delay I often saw when doing so. I would see in the Serial Monitor that the device connected successfully and even sent the GET request but I wouldn’t see the LED light up until a second to multiple seconds later. Never quite figured out what this large delay was for but over time it went down to less than a second, though there was definitely still much more of a delay than a normal browser experienced when clicking on a button.

Once that hump was overcame, the next was getting the distance sensor running how I wanted it to. I had a tough time getting it to show consistent numbers at first with many reads above 1000 cm (this was before I switched it to inches) indicating to me that it was not receiving back its signal it sent out. Maybe this was just my environment but I ended up coming up with some good code systems to help deal with this. First I implemented a rolling average of the last 5 measurements which helped but not as much as I was hoping as the huge measurements would still throw it off. Then I implemented an if statement where if the measurement was too large it would ignore that measurement and move on. This meant I only kept accurate measurements but could still see when it had bad measurements. These combined lead to very consistent results, especially since I decided to take measurements every 100 ms or so, and much quicker that that if it hit bad measurements.

#### Lastly, I just overall enjoyed being able to mess with a sensor and directly see accurate measurements or results being displayed digitally or even through the air to the other device to be displayed visually with the LED. It helped truly light the fire that has started for Internet of Things devices.

#### 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?

I think my system actually could scale to an extent to more devices as it only sent a request upon a change in distance threshold. With that, I think it would for sure hit a limit, especially of a limit of what incoming traffic the D1-mini Stoplight Web Server would be able to handle. I think it would be doable to add a single actuator to the system though it would not be the most efficient way and would likely be extremely difficult to continue to scale past another actuator.

1. 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?

I think the huge strength to me of the tennis-ball system is it does not rely on any power or technology to be up and running. It can just hang there and if its not hanging there then you know you have to be careful. A weakness to both systems could be the issue of tampering though that should not be too big of an issue in ones own home. A strength of the IOT system would be not only does it look cool but the different LEDs give accurate measurements to how close you are getting instead of trying to rely on depth perception through the windshield of the tennis ball. As for an enhancement, first I would recommend only having it run when applicable, like the garage being open as currently it would just constantly be running no mater if the car is there or not and that would take constant power.

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

I think I hit quite a bit on this question in my observation thoughts so make sure to check there for details but this would be the connection delay or distance reading errors that I had to overcome. Again check the observation for specifics but something interesting with the connection delay is I ended up deciding to make a /Blink end point to handle the Blinking instead of trying to send multiple connection frequently to get one of the LEDs to blink. This was quite easy to implement since I had a good grasp on the Stoplight system but sure did add another step that if there was little to no delay I would have just found a good balance of sending LED status changes instead.

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

I would estimate I spent about 6 hours coding/building the project and then 3-4 hours creating the lab write up.

#### Certification of Work

I certify that the results and solution to this lab were my own work. For the resources of information I found through exploring the internet, I referenced the website and what I pulled from it. All code written was of my own writing.

-Rylan Hemsley

#### Appendix

Code for reference:

------------------Distance Stoplight code-------------------

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

const char\* ssid = "RYLANSLAPTOP"; // Set Wifi Name

const char\* password = "Onyourleft100"; // Set Password

WiFiClient client;

int HTTP\_PORT = 80; // Set to Port 80

String HTTP\_METHOD = "GET"; // or "POST" // Using GET for this. Either would work

char HOST\_NAME[] = "192.168.137.195"; // hostname/ip of web server

unsigned char echoPin = D1; // Set Echo Pin

unsigned char trigPin = D2; // Set Trig Pin

long duration; // sensor duration measurement

int distance1 = 0; // Current distance measurement

int distance2 = 0; // Previous distance measurement 2

int distance3 = 0; // Previous distance measurement 3

int distance4 = 0; // Previous distance measurement 4

int distance5 = 0; // Previous distance measurement 5

int avgDistance = 0; // Sensor duration measurement

bool greenOn = false; // Green status

bool yellowOn = false; // Yellow status

bool redOn = false; // Red status

bool blinkOn = false; // Blink status

// Open Green page to change green LED

void cycleGreen() {

// If a successful connection, open page

if(client.connect(HOST\_NAME, HTTP\_PORT)) {

Serial.println("Connected to arduino server");

// send HTTP request header of /Green

client.println(HTTP\_METHOD + " " + "/Green" + " HTTP/1.1");

client.println("Host: " + String(HOST\_NAME));

client.println("Connection: close");

client.println(); // end HTTP header

Serial.println("Navigated to Green Page");

} else {

Serial.println("connection for Green failed");

}

}

// Open Yellow page to change yellow LED

void cycleYellow() {

// If a successful connection, open page

if(client.connect(HOST\_NAME, HTTP\_PORT)) {

Serial.println("Connected to arduino server");

// send HTTP request header of /Yellow

client.println(HTTP\_METHOD + " " + "/Yellow" + " HTTP/1.1");

client.println("Host: " + String(HOST\_NAME));

client.println("Connection: close");

client.println(); // end HTTP header

Serial.println("Navigated to Yellow Page");

} else {

Serial.println("connection for Yellow failed");

}

}

// Open Red page to change red LED

void cycleRed() {

// If a successful connection, open page

if(client.connect(HOST\_NAME, HTTP\_PORT)) {

Serial.println("Connected to arduino server");

// send HTTP request header of /Red

client.println(HTTP\_METHOD + " " + "/Red" + " HTTP/1.1");

client.println("Host: " + String(HOST\_NAME));

client.println("Connection: close");

client.println(); // end HTTP header

Serial.println("Navigated to Red Page");

} else {

Serial.println("connection for Red failed");

}

}

// Open Blink page to start red LED blink cycle

void startBlink() {

// If a successful connection, open page

if(client.connect(HOST\_NAME, HTTP\_PORT)) {

Serial.println("Connected to arduino server");

// send HTTP request header of /Blink

client.println(HTTP\_METHOD + " " + "/Blink" + " HTTP/1.1");

client.println("Host: " + String(HOST\_NAME));

client.println("Connection: close");

client.println(); // end HTTP header

Serial.println("Navigated to Blink Page");

} else {

Serial.println("connection for Blink failed");

}

}

// Open Off page to turn off all LEDs

void lightsOff() {

// If a successful connection, open page

if(client.connect(HOST\_NAME, HTTP\_PORT)) {

Serial.println("Connected to arduino server");

// send HTTP request header of /Off

client.println(HTTP\_METHOD + " " + "/Off" + " HTTP/1.1");

client.println("Host: " + String(HOST\_NAME));

client.println("Connection: close");

client.println(); // end HTTP header

Serial.println("Navigated to Off Page");

} else {

Serial.println("connection for Off failed");

}

}

void setup() {

// Setup Serial Monitor with 9600

Serial.begin(9600);

// Create Wifi connection and delay until connected

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

Serial.println("Connecting to WiFi");

while(WiFi.status() != WL\_CONNECTED) { // Loop until connected to WiFi

delay(500);

Serial.print(".");

}

// If connection successful show IP address in serial monitor

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP()); // Show your assigned IP address

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

lightsOff(); // Makes sure all the lights are off to start

}

void loop() {

// Sensor distance checking

digitalWrite(trigPin, LOW); // Clears the trigPin

delayMicroseconds(2); // Small delay

digitalWrite(trigPin, HIGH); // Sets the trigPin on HIGH state

delayMicroseconds(10); // For 10 microseconds

digitalWrite(trigPin, LOW); // trigPin back to LOW state

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the measured distance in inches. Remove "\* 0.393701" to get cm.

distance1 = duration \* 0.034 / 2 \* 0.393701;

// If a bad distance read, just print a ` and don't add the new distance

if (distance1 >= 120) {

Serial.print("`");

}

else {

// add delay to be measuring roughly 9-10 times a second

delay(100);

// Average the previous 5 distances together to get a rolling average

avgDistance = (distance1 + distance2 + distance3 + distance4 + distance5) / 5;

// Shift all of the measured distances once

distance5 = distance4;

distance4 = distance3;

distance3 = distance2;

distance2 = distance1;

// Prints the avgDistance on the Serial Monitor

Serial.print("Average Distance (inch): ");

Serial.println(avgDistance);

}

// Now to check the distance and set the appropriate LEDs with Get requests

// Distance between 120 and 20 inches, set LED to green if not already green

if ((avgDistance <= 120) && (avgDistance > 20) && greenOn == false) {

// Lights Off then Green on

lightsOff();

cycleGreen();

// All status false but green

greenOn = true;

yellowOn = false;

redOn = false;

blinkOn = false;

}

// Distance between 20 and 12 inches, set LED to yellow if not already yellow

else if ((avgDistance <= 20) && (avgDistance > 12) && yellowOn == false) {

// Lights Off then Yellow on

lightsOff();

cycleYellow();

// All status false but yellow

greenOn = false;

yellowOn = true;

redOn = false;

blinkOn = false;

}

// Distance between 12 and 5 inches, set LED to red if not already red

else if ((avgDistance <= 12) && (avgDistance > 5) && redOn == false) {

// Lights Off then Red on

lightsOff();

cycleRed();

// All status false but red

greenOn = false;

yellowOn = false;

redOn = true;

blinkOn = false;

}

// Distance between 5 and 0 inches, set LED to blinking red if not already blinking

else if ((avgDistance <= 5) && (avgDistance > 0) && blinkOn == false) {

// Lights Off then Blink Red on

lightsOff();

startBlink();

// All status false but blink

greenOn = false;

yellowOn = false;

redOn = false;

blinkOn = true;

}

}

------------------Updated Web Server Code------------------------

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

// Needed variables for LED Pin association

unsigned char greenLed = D3; // Create greenLed Label for pin 3

unsigned char yellowLed = D2; // Create yellowLed Label for pin 2

unsigned char redLed = D1; // Create redLed Label for pin 1

// Needed variables for looping

unsigned long previousTime = 0; // Create previousTime int for looping

unsigned long startTime = 0; // Create startTime int for looping

bool runStoplight = false; // Create runStoplight bool for looping

bool runBlink = false;

bool redStatus = false; // Create Red bool for cycling red

bool yellowStatus = false; // Create Yellow bool for cycling yellow

bool greenStatus = false; // Create Green bool for cycling green

// wifi and server config

const char\* ssid = "RYLANSLAPTOP"; // Set Wifi Name

const char\* password = "Onyourleft100"; // Set Password

ESP8266WebServer server(80); // Set server to port 80

// Simple HTML saved in a string with the different page references in buttons

const char MAIN\_page[] PROGMEM = R"=====(

<html>

<head>

<title>Home Webpage</title>

<h1>Home Webpage</h1>

<h2>Stoplight</h2>

</head>

<body>

<a href="/Off"><button style="background-color:grey">Off</button></a>

<a href="/Red"><button style="background-color:red">Red</button></a>

<a href="/Yellow"><button style="background-color:yellow">Yellow</button></a>

<a href="/Green"><button style="background-color:lightGreen">Green</button></a>

<a href="/Loop"><button style="background-color:lightGrey">Loop</button></a>

</body>

</html>

)=====";

// Handle default page call

void handleRoot() {

String page = MAIN\_page; // Store HTML in a string

server.send(200, "text/html", page); // Send HTML to user

}

// Handle Off call and reset needed variables and LEDs

void handleOff() {

String page = MAIN\_page; // Store HTML in a string

server.send(200, "text/html", page); // Send HTML to user

digitalWrite(greenLed, LOW); // Turn Green LED off

digitalWrite(yellowLed, LOW); // Turn Yellow LED off

digitalWrite(redLed, LOW); // Turn Red LED off

greenStatus = false; // Indicate green is off

yellowStatus = false; // Indicate yellow is off

redStatus = false; // Indicate red is off

runStoplight = false; // Stop stoplight looping

runBlink = false; // Stop blink looping

previousTime = 0; // Reset previousTime

startTime = 0; // Reset startTime

Serial.println("Turning Off LEDs");

}

// Handle Red call and cycle red LED depending on status

void handleRed() {

String page = MAIN\_page; // Store HTML in a string

server.send(200, "text/html", page); // Send HTML to user

// If off turn on and if on turn off

if (redStatus == false) {

digitalWrite(redLed, HIGH); // Turn Red LED on

redStatus = true; // Flip status

}

else {

digitalWrite(redLed, LOW); // Turn Red LED off

redStatus = false; // Flip status

}

Serial.println("Cycling Red");

}

// Handle Yellow call and cycle yellow LED depending on status

void handleYellow() {

String page = MAIN\_page; // Store HTML in a string

server.send(200, "text/html", page); // Send HTML to user

// If off turn on and if on turn off

if (yellowStatus == false) {

digitalWrite(yellowLed, HIGH); // Turn Yellow LED on

yellowStatus = true; // Flip status

}

else {

digitalWrite(yellowLed, LOW); // Turn Yellow LED off

yellowStatus = false; // Flip status

}

Serial.println("Cycling Yellow");

}

// Handle Green call and cycle green LED depending on status

void handleGreen() {

String page = MAIN\_page; // Store HTML in a string

server.send(200, "text/html", page); // Send HTML to user

// If off turn on and if on turn off

if (greenStatus == false) {

digitalWrite(greenLed, HIGH); // Turn Green LED on

greenStatus = true; // Flip status

}

else {

digitalWrite(greenLed, LOW); // Turn Green LED off

greenStatus = false; // Flip status

}

Serial.println("Cycling Green");

}

// Handle Loop call and setup needed variables

void handleLoop() {

String page = MAIN\_page; // Store HTML in a string

server.send(200, "text/html", page); // Send HTML to user

digitalWrite(greenLed, LOW); // Turn Green LED off

digitalWrite(yellowLed, LOW); // Turn Yellow LED off

digitalWrite(redLed, LOW); // Turn Red LED off

greenStatus = false; // Indicate green is off

yellowStatus = false; // Indicate yellow is off

redStatus = false; // Indicate red is off

runStoplight = true; // Start stoplight looping

previousTime = millis(); // Set previousTime to current runtime

startTime = millis(); // Then set startTime to current runtime

Serial.println("Starting Stoplight Loop");

}

void handleBlink() {

String page = MAIN\_page; // Store HTML in a string

server.send(200, "text/html", page); // Send HTML to user

digitalWrite(greenLed, LOW); // Turn Green LED off

digitalWrite(yellowLed, LOW); // Turn Yellow LED off

digitalWrite(redLed, LOW); // Turn Red LED off

greenStatus = false; // Indicate green is off

yellowStatus = false; // Indicate yellow is off

redStatus = false; // Indicate red is off

runBlink = true; // Start blink looping

}

void setup() {

// Setup Serial Monitor with 9600

Serial.begin(9600);

// Create Wifi connection and delay until connected

WiFi.begin(ssid, password);

Serial.println("Connecting to WiFi");

while(WiFi.status() != WL\_CONNECTED) { // Loop until connected to WiFi

delay(500);

Serial.print(".");

}

// If connection successful show IP address in serial monitor

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP()); // Show your assigned IP address

// Setup all 3 LED pins and make sure they are off

pinMode(greenLed, OUTPUT); // Set Green LED to Output pin

pinMode(yellowLed, OUTPUT); // Set Yellow LED to Output pin

pinMode(redLed, OUTPUT); // Set Red LED to Output pin

digitalWrite(greenLed, LOW); // Turn Green LED off

digitalWrite(yellowLed, LOW); // Turn Yellow LED off

digitalWrite(redLed, LOW); // Turn Red LED off

// Setup the needed Handles for the client operations

server.on("/", handleRoot);

server.on("/Off", handleOff);

server.on("/Red", handleRed);

server.on("/Yellow", handleYellow);

server.on("/Green", handleGreen);

server.on("/Loop", handleLoop);

server.on("/Blink", handleBlink);

// Start server

server.begin();

Serial.println("HTTP server started");

}

void loop() {

// Run server client requests

server.handleClient();

// Stoplight Controlling only runs when set to true

if (runStoplight == true) {

// compare last run to starting time, turn to seconds, then

// check if within first second of three second loop

if ((((previousTime - startTime) / 1000)%3) == 0) {

digitalWrite(greenLed, HIGH); // Turn Green LED on

digitalWrite(yellowLed, LOW); // Turn Yellow LED off

digitalWrite(redLed, LOW); // Turn Red LED off

Serial.println("Loop Green ON");

previousTime = millis(); // Set previous time to current time

}

// compare last run to starting time, turn to seconds, then

// check if within second second of three second loop

else if ((((previousTime - startTime) / 1000)%3) == 1) {

digitalWrite(greenLed, LOW); // Turn Green LED off

digitalWrite(yellowLed, HIGH); // Turn Yellow LED on

digitalWrite(redLed, LOW); // Turn Red LED off

Serial.println("Loop Yellow ON");

previousTime = millis(); // Set previous time to current time

}

// compare last run to starting time, turn to seconds, then

// check if within third second of three second loop

else if ((((previousTime - startTime) / 1000)%3) == 2) {

digitalWrite(greenLed, LOW); // Turn Green LED off

digitalWrite(yellowLed, LOW); // Turn Yellow LED off

digitalWrite(redLed, HIGH); // Turn Red LED on

Serial.println("Loop Red ON");

previousTime = millis(); // Set previous time to current time

}

}

// Blink operation only runs when set to true

if (runBlink == true) {

// If off turn on and if on turn off

if (redStatus == false) {

digitalWrite(redLed, HIGH); // Turn Red LED on

redStatus = true; // Flip status

}

else {

digitalWrite(redLed, LOW); // Turn Red LED off

redStatus = false; // Flip status

}

Serial.println("Cycling Red");

delay(100); // Delay for 100ms

}

}