Lab 2 Report (2022)
Arduino Webpage 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:

- Reinforce enumerating requirements from use cases and user stories
- Become familiar with the Arduino platform and its programming methodologies.
- Program an Arduino-based microcontroller to use the GPIO pins to control LEDs.
- Develop iteratively, beginning with a minimum viable product and add functionality until the requirements are met.

Materials
I used the following materials to accomplish this lab:

- Arduino Wemos D1 Mini with ESP8266
- MicroUSB Cable with power source
- External device to view webpage (This lab uses a smartphone with a wifi hotspot to simplify the process)
- 1 x breadboard
- 1 x Green LED
- 1 x Yellow LED
- 1 x Red LED
- 3 x 100 Ω Resistor (BrBlBrGl)
- 4 x Jumper Wires

References
I used the following resources in this lab:

- 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://www.w3schools.com/html/ - Adjustments to the HTML code were needed to fit into Arduino and W3 helped.
- https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-hosting-a-webserver - This along with some other Arduino.cc tutorials helped to explain how to create a webserver and display HTML through it.
- 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. 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.
3. Make sure the Arduino is properly connected to the computer with the computer via a USB cable.
4. Upload the code. Once the upload has completed the built-in Arduino LED should now be rapidly flashing. (This indicates awaiting connection to wifi)
5. Once a connection is established the built-in LED will stop blinking and the colored LEDs should cycle from Red through to Green to indicate a good status of server start.
6. The webpage should now be accessible at the Arduino's IP address.

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 their actions with the web interface.

Web interface

This part is a simple HTML website that has all of the buttons for manual usability of the stoplight as well as an auto cycle button for hands free operation. Not feedback is provided in this interface as it is seen on the stoplight. If an error occurs the webpage will most likely no longer show.

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. While in the “Setup” state:
      1. The system configures GPIO outputs and clears all lights off.
      2. Red light is activated to indicate non-connection status.
      3. 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.
      4. Yellow light is then activated to indicate connection achieved.
      5. Web server is then started followed by a Green status light update with all lights being cleared following this.
      6. The primary loop then begins
   ii. While in the “Loop” state.
      1. This loops primary job is to listen to web requests and perform actions based on this.
      2. If /off is called:
         a. Cycle is turned off.
         b. allLightsOff function is called which switches all GPIO pins to LOW
      3. If /green is called:
         a. Cycle is turned off.
         b. greenlight function is called which switches the green GPIO pin to HIGH and the rest LOW
      4. If /yellow is called:

    a. Cycle is turned off.
    b. yellolight function is called which switches the yellow GPIO pin to HIGH and the rest LOW
5. If /red is called:
    a. Cycle is turned off.
    b. red function is called which switches the red GPIO pin to HIGH and the rest LOW
6. If /cycle is called:
    a. Cycle is turned on.
    b. The millisecond timer is reset.
    c. Additional if statements follow checking for how much time has pass and displaying the appropriate light ending with a timer reset starting the cycle over.

Observations

I had a lot of fun with this lab once I got things rolling. Keeping things simple with less “moving parts” in a way was a welcome change from R Pi. I am excited to see how I can use this system in our future labs.

One of the biggest difficulties for me was just getting started. I spent a while trying to find resources that would point me in the right direction without leading me down a path that ultimately would be too complicated. Once I found the resource to use the right modules it was far more straightforward for me to get it going. In this starting phase as well I also struggled with getting the Ardunio IDE to connect properly to my device which led to much frustration from needing to use another system to program on and upload from.

Once I was up and running learning the ins and outs of Ardunio was fun. Leveraging the usability of the quickly running loop was very cool to work with and made more sense to me that the per request operation of python code previously.

Because there is only a single device, there isn’t any communication between internal components. The single device does the following:

- Listening on TCP port 80 for web responses
- Parsing input through IF statements
- Controlling the LEDs
- Responding to the client with fresh HTML

Thought Questions

What are some key differences between developing this lab on a Raspberry Pi, and developing on Arduino?

The biggest difference is the simplification of the system. Working in a Raspberry Pi requires more setup and other systems to fully function as intended. Ardunio is able to all of this in a single program and can run indefinitely all on it's own. This can be difficult to figure out programmatically but results in a more independent system.

What are the strengths and trade-offs of each of these platforms?

Arduino is a potentially smaller and lower-powered system than Rasberry Pi is. Anything programmed in Ardunio can easily be used on a plethora of other Ardunio-based devices allowing for more flexibility. Additional systems can easily be integrated into the Ardunio architecture natively.

Raspberry Pi functions great as a self-contained mini-computer with many of the basic abilities of one. Onboard programming and system integration makes the Pi an amazing dev platform to perform many different kinds of testing. Since it has the ability to run a basic operating system many different languages can be used and run on it allowing for system connections to happen within the device. It is also able to do this due to the much larger processing and system resources that are available.

How familiar were you with the Arduino platform prior to this lab?

I previously used the Arduino platform to control a stepper motor as part of a larger project that involved several other systems. Other than this I haven't used Ardunio before or after.

What was the biggest challenge you overcame in this lab?

The biggest challenge for me was figuring out a new implementation for the auto cycle functionality. In the previous project, I allowed some liberties namely that of leaving the webpage hang while the cycle was running. Figuring out a new way to do this without the previous issues took sometime. With the help of other classmates I was able to put together an implementation that I was happy with.

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

I spent about 10 hours on the project with 3 hours on this lab report.

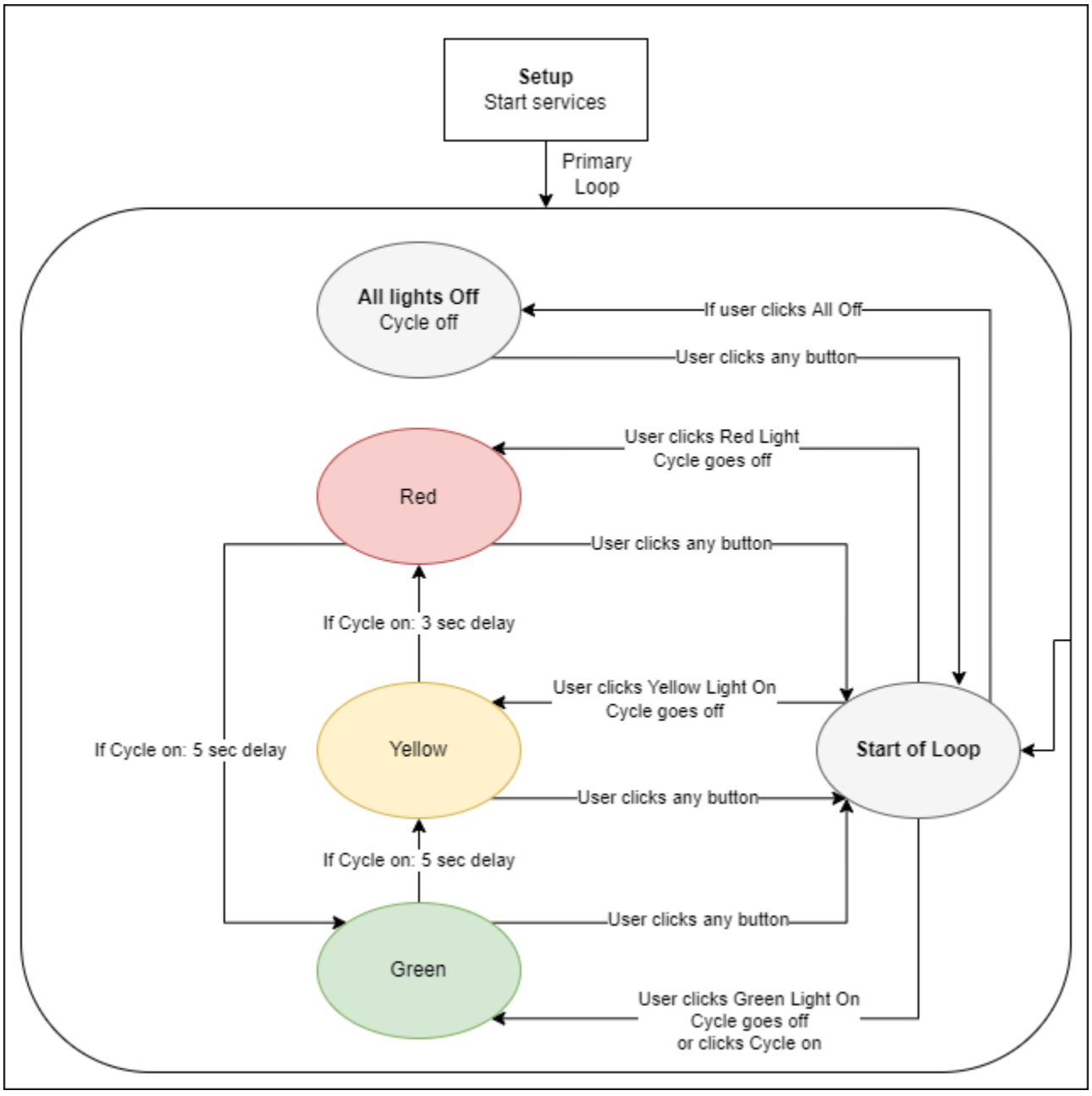
Certification of Work

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.

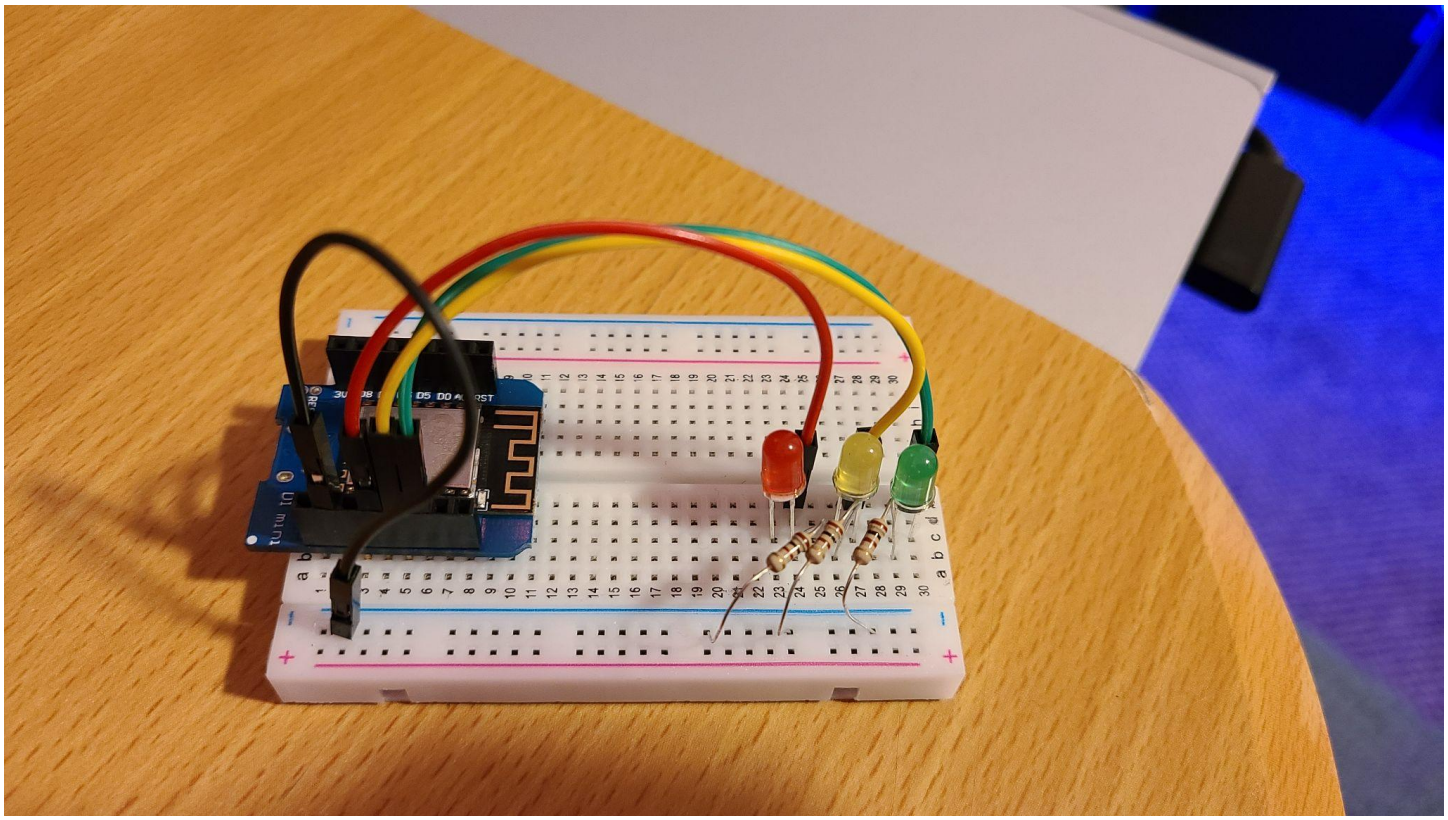
--Blake Porter

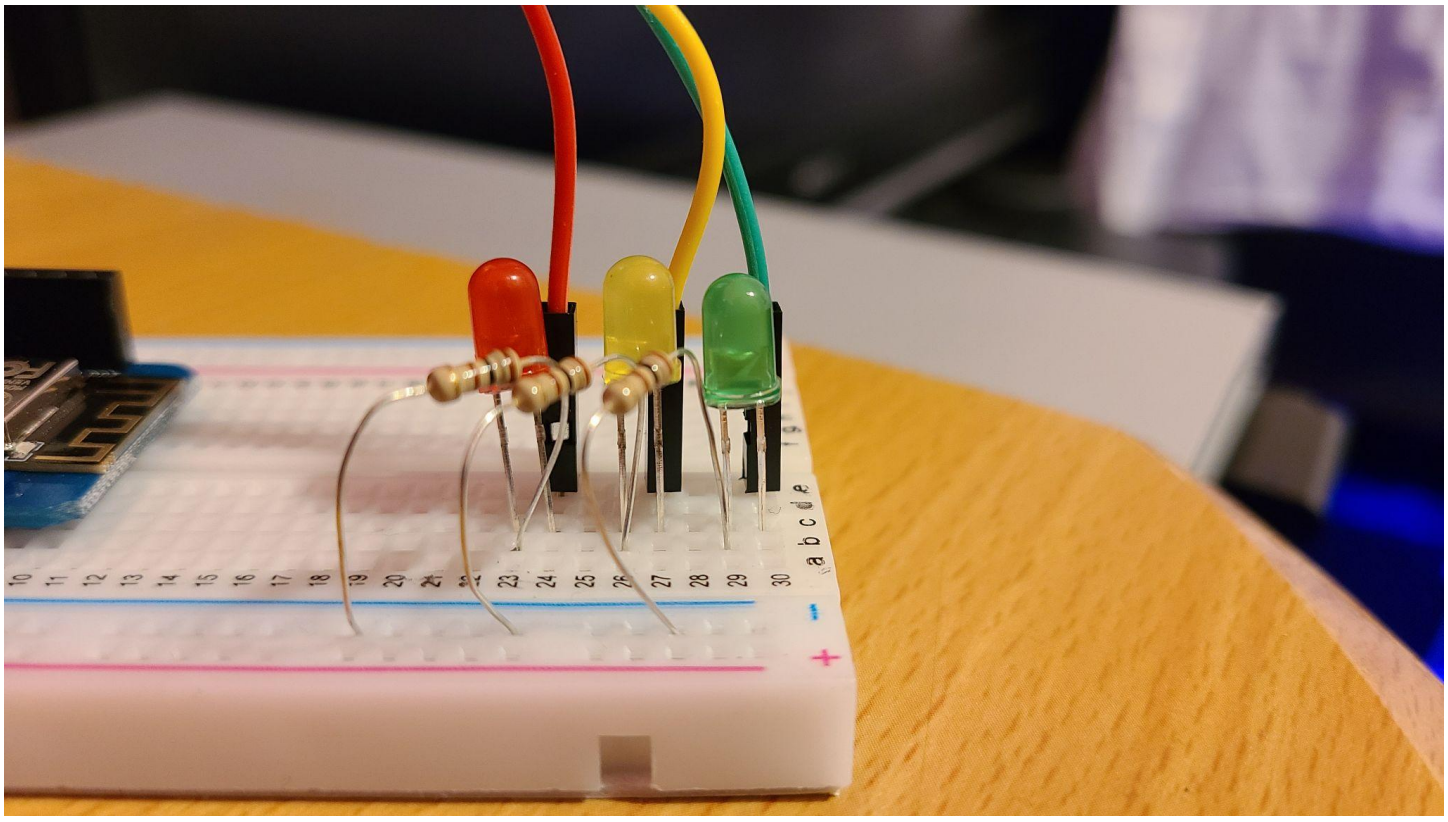
Appendix

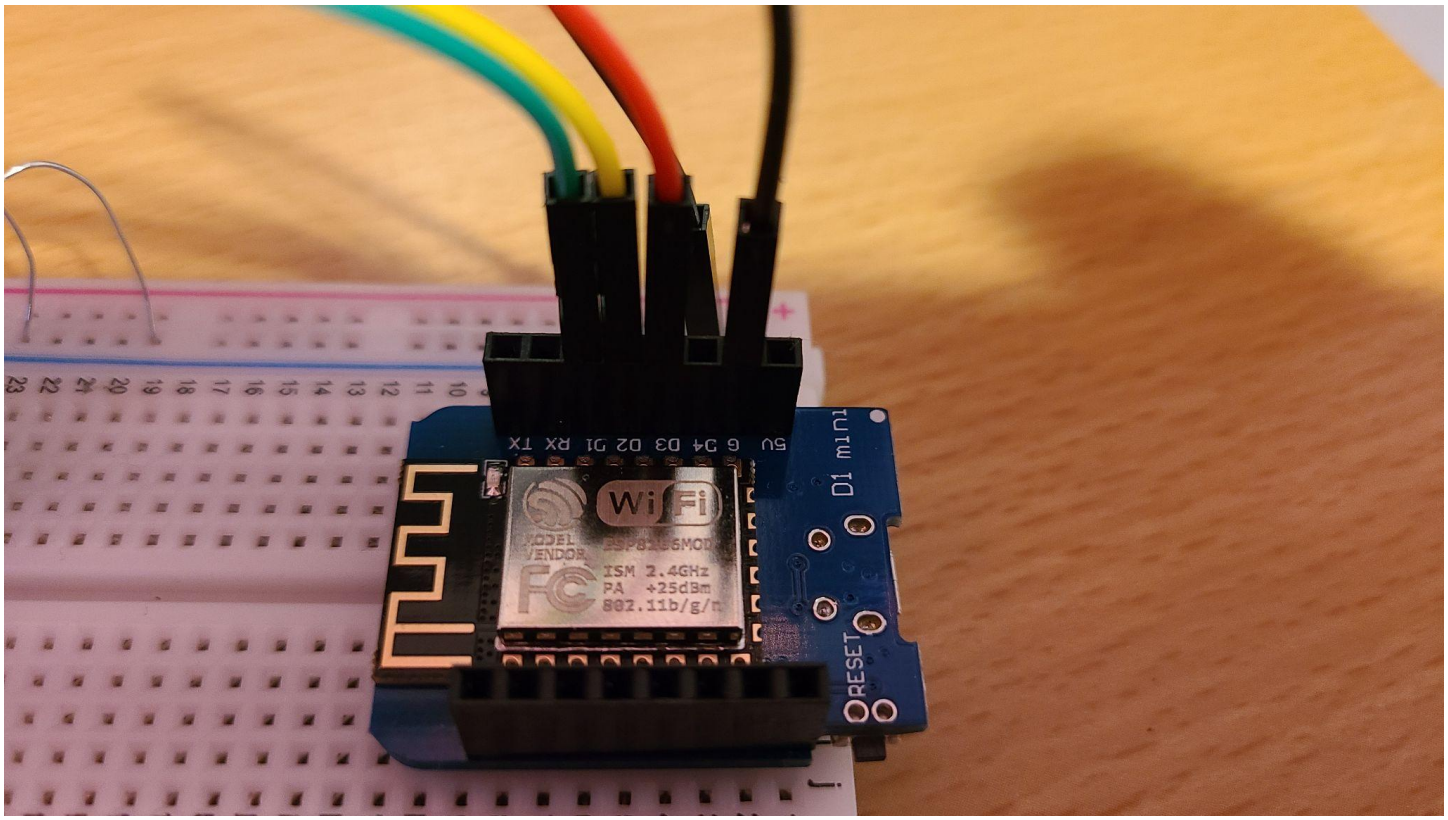
Appendix 1: State Diagram

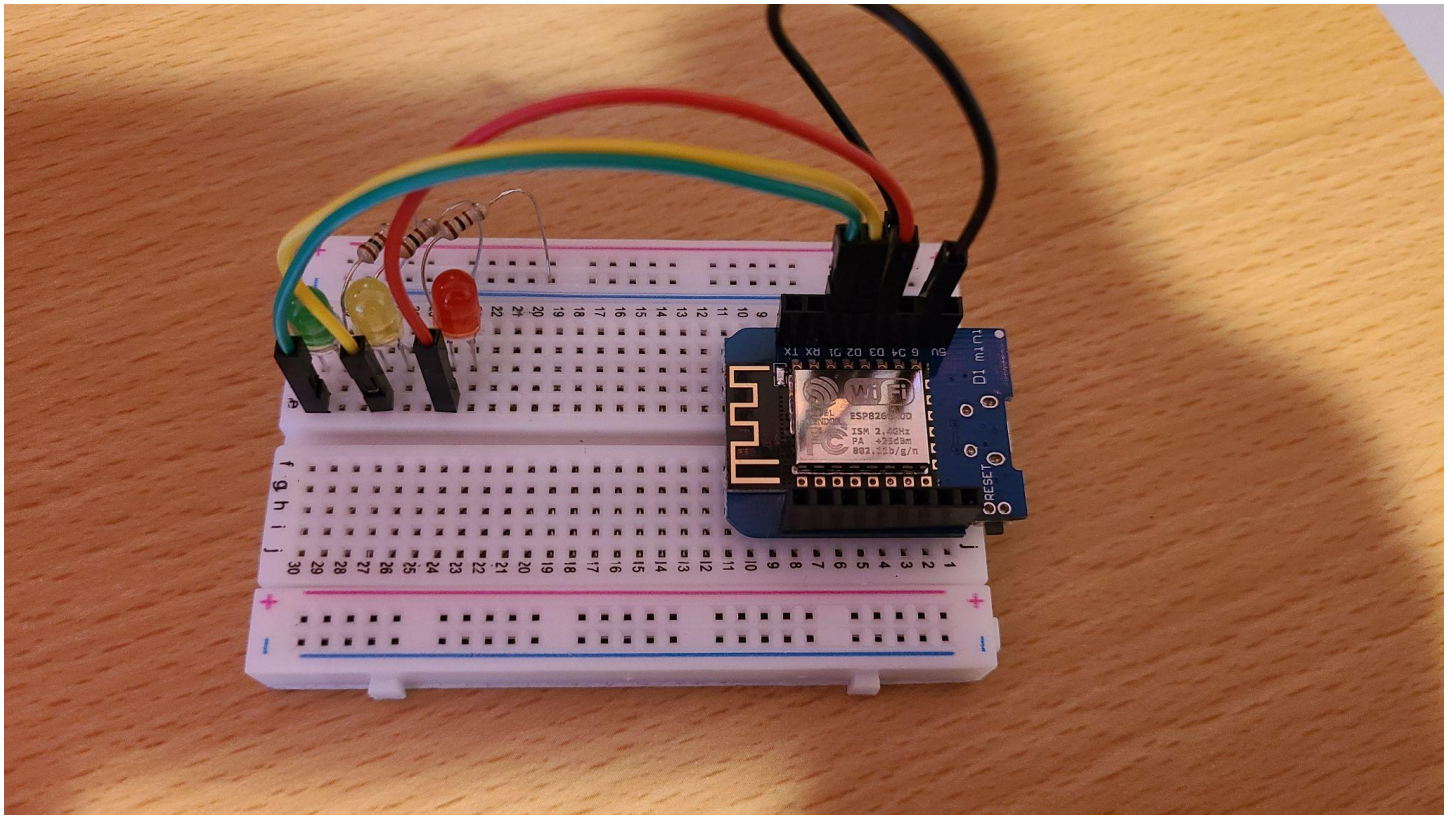


Appendix 2: Physical Components Layout



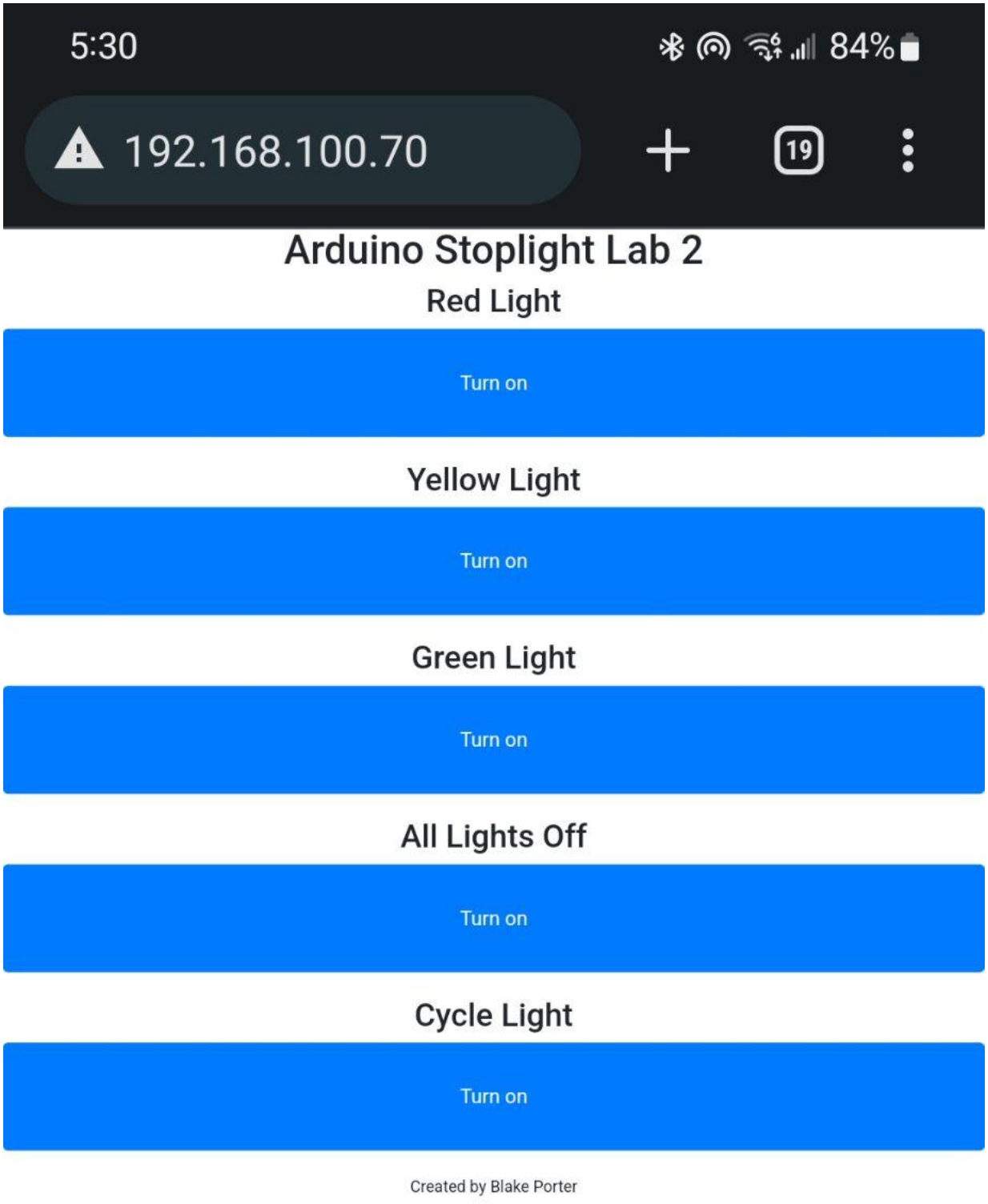



ISM 2.4GHz
PA +25dBm
802.11b/g/n
RESET

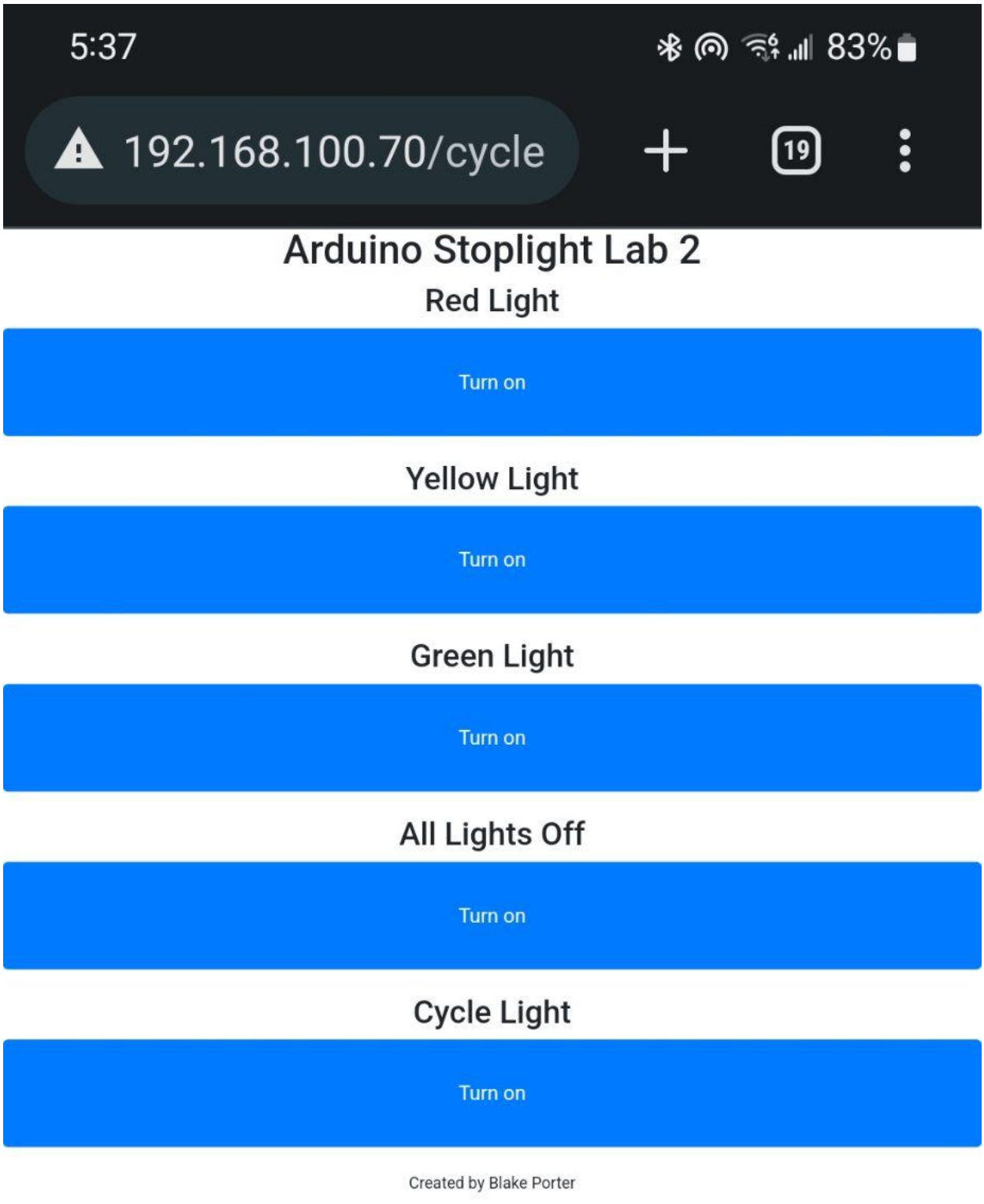


Appendix 3: System Interface - Web Page

Main page before any endpoint is called.



Example state showing the /cycle endpoint. This is similar to how all the direct light controls are triggered.



Appendix 4: Ardunio Code

(available at https://github.com/DemonPiranha/ITC-441-labs/blob/a9becd69870dc45b8526d4c29288d064d46c5ef4/Lab2/Lab2.ino)

```
#include <ESP8266WiFi.h>

//Sets some global variables
const char* ssid = "Blake-Hotspot";
const char* password = "blakepublic";
WiFiServer server(80);
unsigned long startMillis;
bool cycle = false;

//Setup for pins and clears all of them
void setup() {

  //Setup serial and all LED outputs then clears the LEDs
  Serial.begin(9600);
  pinMode(D1, OUTPUT);
  pinMode(D2, OUTPUT);
  pinMode(D3, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  allLightsOff();

  //Using the Wifi library begin connecting to the set network and provide status updates.
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  redLight();
  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);
  delay(100);
  yellowLight();
  delay(100);
```

```
  Serial.println();
  Serial.println("(( WiFi connected ))");
  Serial.println(WiFi.localIP());

  server.begin();
  Serial.println("((( Server Started )))");
  greenLight();
  delay(500);
  allLightsOff();
}

void loop() {

  // Sets up a client object for connections to the webpage. Constantly checks if a request
is made
  WiFiClient client = server.available();
  if (client) {
    digitalWrite(LED_BUILTIN, LOW);
    delay(10);
    digitalWrite(LED_BUILTIN, HIGH);
    while (!client.available()) {
      delay(10);
    }
  }
  client.setTimeout(100);

  //This section reads for inbound requests made by the web client. If any one of the
endpoints is hit the corresponding code will trigger.
  String req = client.readStringUntil('\r');
  if (req.indexOf("/off") != -1) {
    cycle = false;
    allLightsOff();
  }
  if (req.indexOf("/green") != -1) {
    cycle = false;
    greenLight();
  }
  if (req.indexOf("/yellow") != -1) {
    cycle = false;
    yellowLight();
```

```
  }
  if (req.indexOf("/red") != -1) {
    cycle = false;
    redLight();
  }
  if (req.indexOf("/cycle") != -1) {
    cycle = true;
    startMillis = millis();
  }

  //Checks if the program is in cycle mode and plays the appropriate light pattern.
  if (cycle == true){
    if ((millis() - startMillis) < 5000 ){
      greenLight();
    }
    if ((millis() - startMillis) > 5000 && (millis() - startMillis) < 8000 ){
      yellowLight();
    }
    if ((millis() - startMillis) > 8000 && (millis() - startMillis) < 13000 ){
      redLight();
    }
    if ((millis() - startMillis) > 13000){
      startMillis = millis();
    }
  }

  //These client.print commands print out the html for the webpage and ultimately generates
the page. This allows for the page to be served and interacted with in real time.
  client.println("<!doctype html>");
  client.println("<html>");
  client.println("<head>");
  client.println("<link rel='stylesheet'
href='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css'
integrity='sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh'
crossorigin='anonymous'>");
  client.println("<title>Aurdunio Stoplight</title>");
  client.println("</head>");

  client.println("<center>");
  client.println("<body>");
```

```
  client.println("<header>");
  client.println("<h1>Arduino Stoplight Lab 2</h1>");
  client.println("</header>");

  client.print("<h2>Red Light</h2>");
  client.print("<a href='http://");
  client.print(WiFi.localIP());
  client.print("/red' class='btn btn-block btn-lg btn-primary' role='button'><br>Turn
on<br><br></a><br>");

  client.print("<h2>Yellow Light</h2>");
  client.print("<a href='http://");
  client.print(WiFi.localIP());
  client.print("/yellow' class='btn btn-block btn-lg btn-primary' role='button'><br>Turn
on<br><br></a><br>");

  client.print("<h2>Green Light</h2>");
  client.print("<a href='http://");
  client.print(WiFi.localIP());
  client.print("/green' class='btn btn-block btn-lg btn-primary' role='button'><br>Turn
on<br><br></a><br>");

  client.print("<h2>All Lights Off</h2>");
  client.print("<a href='http://");
  client.print(WiFi.localIP());
  client.print("/off' class='btn btn-block btn-lg btn-primary' role='button'><br>Turn
on<br><br></a><br>");

  client.print("<h2>Cycle Light</h2>");
  client.print("<a href='http://");
  client.print(WiFi.localIP());
  client.print("/cycle' class='btn btn-block btn-lg btn-primary' role='button'><br>Turn
on<br><br></a><br>");


  client.println("<span>Created by Blake Porter</span>");
  client.println("</body>");
  client.println("</center>");
  client.println("</html>");
}
```

```
//These functions directly control the lights so that the loop can call these at any time to change the light that is on.
void allLightsOff() {
  digitalWrite(D1, LOW);
  digitalWrite(D2, LOW);
  digitalWrite(D3, LOW);
}

void greenLight() {
  digitalWrite(D1, HIGH);
  digitalWrite(D2, LOW);
  digitalWrite(D3, LOW);
}

void yellowLight() {
  digitalWrite(D1, LOW);
  digitalWrite(D2, HIGH);
  digitalWrite(D3, LOW);
}

void redLight() {
  digitalWrite(D1, LOW);
  digitalWrite(D2, LOW);
  digitalWrite(D3, HIGH);
}
```