Arduino Traffic Light Server

Online Link:

This project is available on my online portfolio at: http://timothyjgow.com/

Objective:

The purpose of this project is to learn the basics of controlling a Wi-Fi capable Arduino board. This includes the following principles:

- Install and configure the Arduino IDE for the Wemos D1 Mini Microcontroller
- Become familiar with connecting LEDs to GPIO pins
- Learn to code simple instructions and compiling and uploading to the Arduino board
- Building a basic web server that can receive commands to turn on or off LEDs

Materials:

The materials needed for this project are the following:

- Personal Computer
- Arduino IDE
- Wemos D1 Mini
- MicroUSB cable
- 1x Breadboard
- 1x Red LED
- 1x Yellow LED
- 1x Green LED
- 4x Jumper Wires
- 3x 220 Ω Resistos

References:

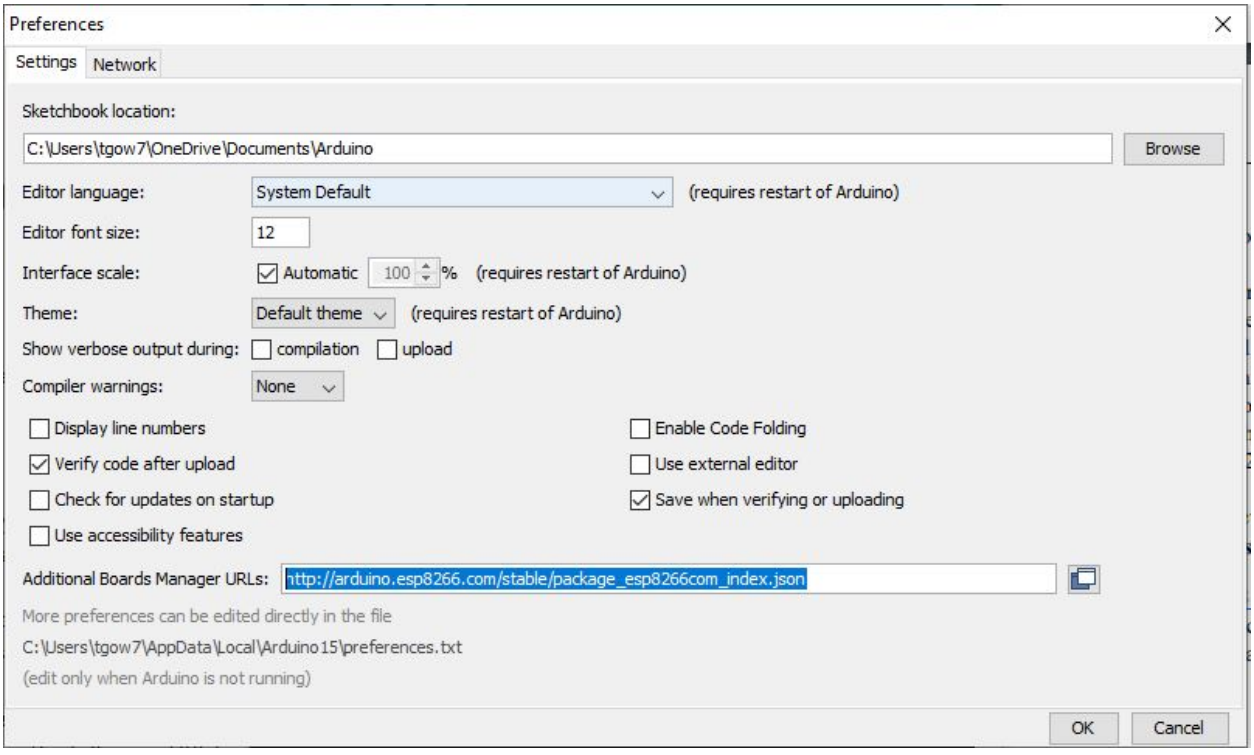
Below are the resources I used to build this project:

- http://www.esp8266learning.com/wemos-webserver-example.php This provided me with the information for building the web server and connecting to a Wi-Fi network.
- https://blog.jeronimus.net/2018/03/rfid-and-wemos-d1-mini-1.html This provided me with layout of the Pins on the Wemos D1 Mini.

- https://www.arduino.cc/en/main/software This is were I downloaded the Arduion IDE.

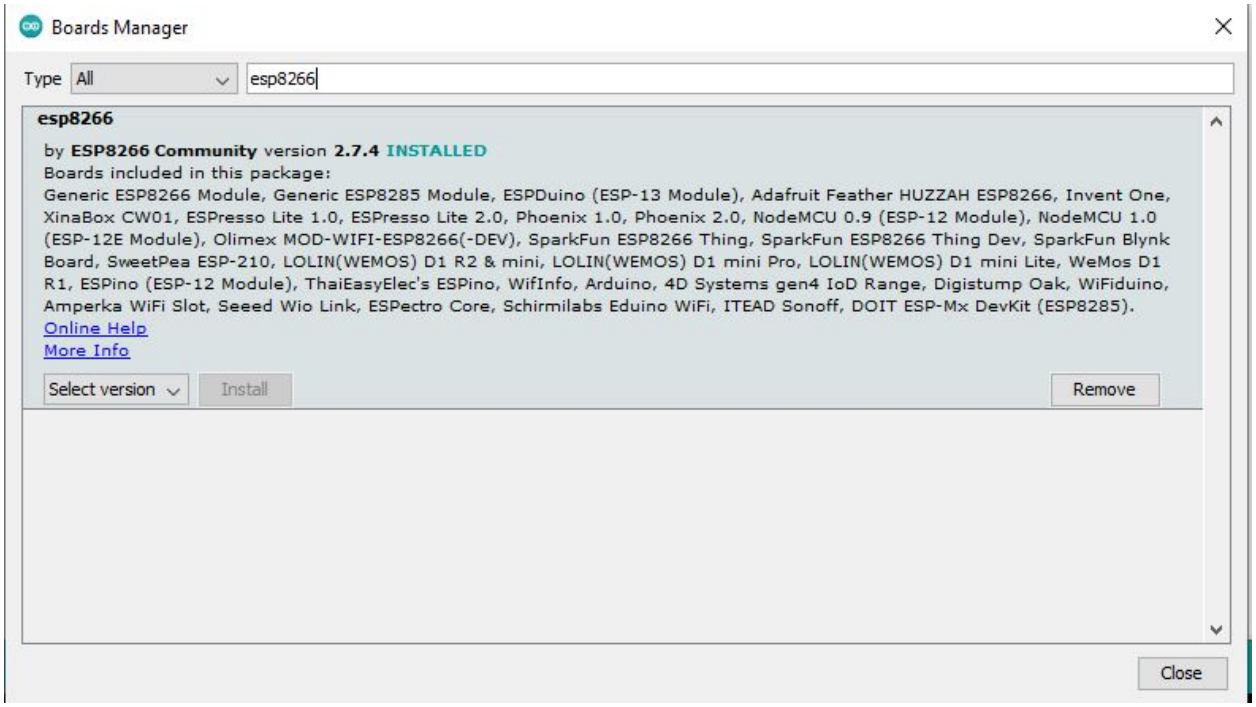
Procedures:

1. Download and install the Arduino IDE from https://www.arduino.cc/en/main/software
2. Install the CH340 USB to Serial controller (if not automatically installed) on the computer with the Arduino IDE.
   a. The Arduino IDE needs to be able to upload and program the D1 mini. The USB to serial chip can be installed in Windows Device Manager.
3. Add the Wemos D1 Library to the Arduino IDE
   a. In the Arduino IDE, Select **File** and then **Preferences.**
   b. In the Additional Boards Manager URLs field, paste in the following URL for the Wemos D1 Mini: http://arduino.esp8266.com/stable/package_esp8266com_index.json

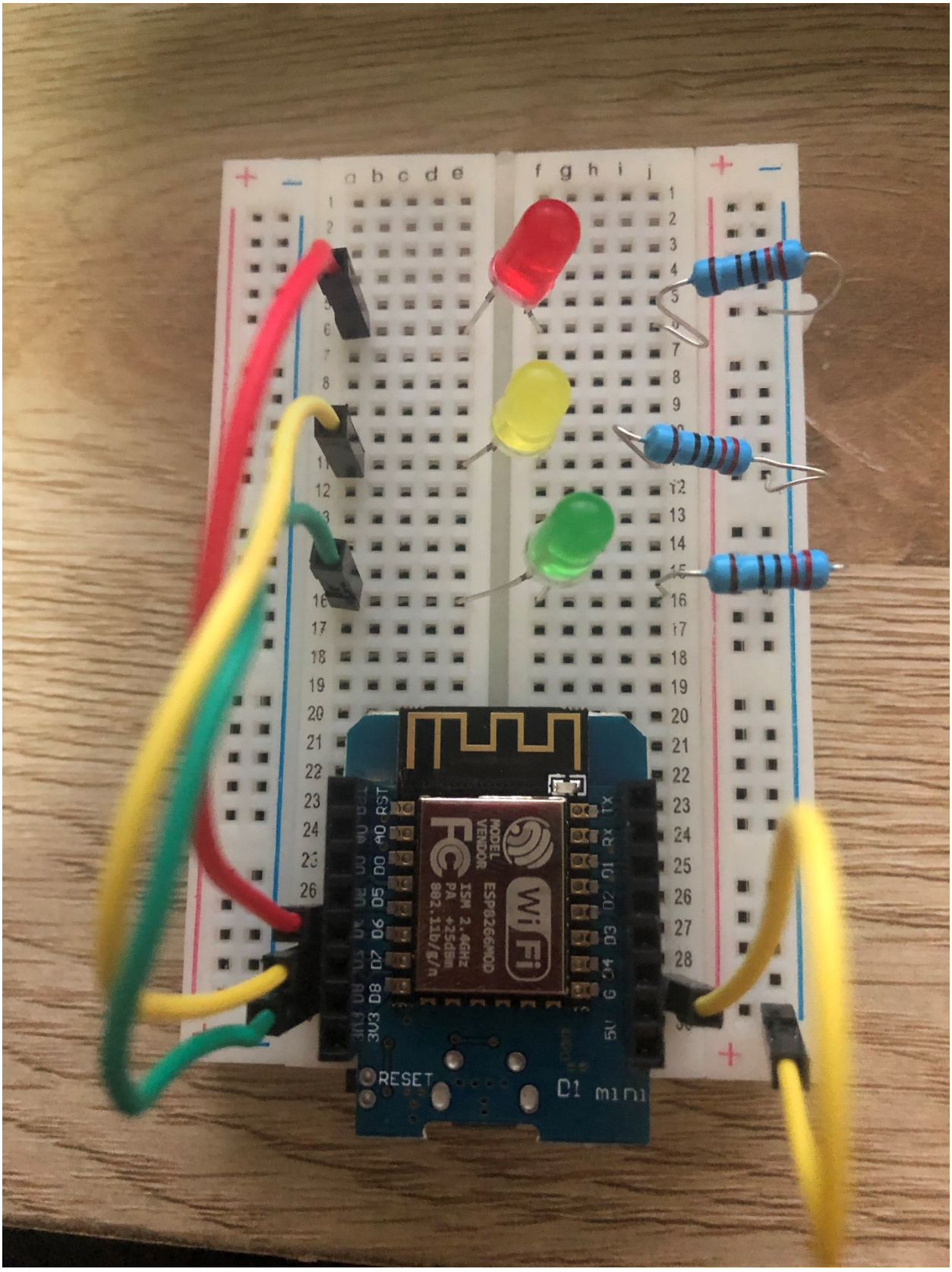


4. Install the ESP8266 Library:
   a. **Go to Sketch** / **Include Library** / **Manage Libraries,** in the search box enter **ESP8266 Platform**
   b. Install the Library called **ESP8266 Microgear**
5. Next step is to install the board so that it shows up in the tools menu alongside the other Arduino boards.

a. Go to **Tools** / **Board** / **Boards Manager**, then search for **ESP8266**. Install the first option.



6. Connect the Wemos board into your pc with the MicroUSB cable.
   a. Select which board to use. This could be the Generic ESP8266 module. **Tools** / **Boards** / **ESP8266 Boards** / **Generic ESP8266 Module**
   b. Next select the correct COM port to communicate to the board. Mine for example was COM3. **Tools** / **Port:**
7. Now it is time to wire the LEDs to the Breadboard so we can program them **Make sure** the USB is disconnected from the Board.
   a. Connect the Wemos board into the breadboard so that the female microUSB port is sticking out at the end edge of the breadboard.
   b. Connect a jumper to the G pin on the 5v side. Connect the other end on the negative far column.
   c. Connect the resistors one after another on the negative column with the other connection in the connecting to the middle.
   d. Connect the Anode ends of each LED to each of their corresponding resistors.
   e. Use jumper wires to connect the Cathode end of each LEDs to pins D8, D7, and D6. Which in my case is Green = D8, Yellow = D7, and Red = D6.


RESET
D1 mini

8. Finally, we can start coding the board. Now you can plug in the Wemos board back into your computer if it was disconnected.
   a. We will start by including the wifi requirements at the top along with some important global variables.

```
#include <ESP8266WiFi.h>

const char* ssid = "your wifi name";
const char* password = "password";

int Green = 15;
int Yellow = 13;
int Red = 12;
int cycle = 0;

WiFiServer server(80);
```

Here you include the wifi library, define your network (replace with your own), define your password (replace with your own), define the pin numbers for; green, yellow, and red, define the cycle flag which we will use later, and define the server to start on port 80.

   b. Next we will add the setup code for the program.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(10);
  pinMode(Green, OUTPUT);
  pinMode(Yellow, OUTPUT);
  pinMode(Red, OUTPUT);

  // Connect to Wifi Network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.print(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
```

```
    Serial.print(".");
  }

  Serial.println("\n");
  Serial.println("connected to ");
  Serial.println(ssid);


  //Start server
  server.begin();
  Serial.println("Server has started");

  //Print IP
  Serial.println("Server can be reached at: ");
  Serial.println("http://");
  Serial.println(WiFi.localIP());
  Serial.println("/");

}
//end setup
```

In essence we are connecting to the Wifi network and starting the server.

c. The last step to the code is the main loop. Which will wait for new clients to connect wait for requests. So if the ask for the main address it will return those html lines towards the bottom of the loop. If for example Yellow=ON is specified in the query string then the yellow LED will turn on.

```
void loop() {

// check if cycle has been called.  If it has then run the cycle
//this will run every time the main loop loops, unless cycle off is called.
   if (cycle == 1) {
    digitalWrite(Green, HIGH);
    delay(500);
    digitalWrite(Green, LOW);
    delay(500);
    digitalWrite(Yellow, HIGH);
    delay(500);
    digitalWrite(Yellow, LOW);
    delay(500);
    digitalWrite(Red, HIGH);
    delay(500);
    digitalWrite(Red, LOW);
```

```
  delay(500);
}


// Check to see if client connection
WiFiClient client = server.available();
if (!client) {
  return;
}


// wait for client input
Serial.println("new client connected");
while(!client.available()){
  delay(1);
}


// Read the first line of the request
String request = client.readStringUntil('\r');
Serial.println(request);
client.flush();

// Match the request



if (request.indexOf("/Green=ON") != -1) {
  digitalWrite(Green, HIGH);

}
if (request.indexOf("/Green=OFF") != -1){
  digitalWrite(Green, LOW);

}
if (request.indexOf("/Red=ON") != -1) {
  digitalWrite(Red, HIGH);
  value = HIGH;
}
if (request.indexOf("/Red=OFF") != -1) {
  digitalWrite(Red, LOW);

}
if (request.indexOf("/Yellow=ON") != -1) {
  digitalWrite(Yellow, HIGH);
```

```
  }
  if (request.indexOf("/Yellow=OFF") != -1) {
    digitalWrite(Yellow, LOW);

  }

  if (request.indexOf("/Cycle=ON") != -1) {
    cycle = 1;
  }



  if (request.indexOf("/Cycle=OFF") != -1) {
    cycle = 0;
  }




  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(""); //  do not forget this one
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");


  client.println("<br><br>");
  client.println("Click <a href=\"/Green=ON\">here</a> turn on Green<br>");
  client.println("Click <a href=\"/Green=OFF\">here</a> turn off Green<br>");
  client.println("Click <a href=\"/Yellow=ON\">here</a> turn on Yellow<br>");
  client.println("Click <a href=\"/Yellow=OFF\">here</a> turn off Yellow<br>");
  client.println("Click <a href=\"/Red=ON\">here</a> turn on Red<br>");
  client.println("Click <a href=\"/Red=OFF\">here</a> turn off Red<br>");
  client.println("Click <a href=\"/Cycle=ON\">here</a> turn on Cycle<br>");
  client.println("Click <a href=\"/Cycle=OFF\">here</a> turn off Cycle<br>");
  client.println("</html>");

  delay(1);
  Serial.println("Client disconnected");
  Serial.println("");

}
```

9. The last step is to add upload the code to the board.
    a. In the Arduino IDE click on the arrow pointing to the right. This will compile and upload the code to the Arduino. Once everything has been loaded you can navigate to the IP Address given in the output terminal Go to **Tools** / **Serial Monitor** to open the output terminal. Use the URL address given to get to main html page. Which will have links to turn on the lights and do a cycle.

Thought Questions:

1. What are some key difference between developing this project on a Raspberry Pi, and developing on Arduino?

   Answer: The Arduino is more bare bones. Although I find the Arduino easier to program there are several things that would have been nice to have. The Raspberry Pi can function as a modern day computer. Setting up a powerful web server is pretty easier. It also has the ability to run python script which makes programming super intuitive. The one thing I find lacking on the Arduino was the lack of Threading.

2. What are the strengths and trades-offs of each of these platforms?

   Answer: Arduino doesn't have as much overhead. It is easy to program and is not as complicated as the Raspberry Pi. The Raspberry Pi offers much more power and can do a lot more than the Arduino could do by itself.

3. How familiar were you with the Arduino platform prior to this project?

   Answer: This was my first time using the Arduino platform.

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

   Answer: I had the challenge of figuring out how to use a task scheduler because the Arduino does not offer true multi-threading. However, I find a way around this and did not have to use a task scheduler. Obviously a better design pattern would be to implement a timer.

5. Please estimate the total time you spent on this project.

   Answer: ~5 hours

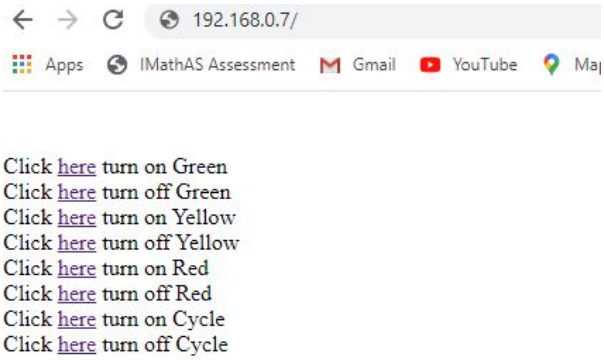
Certification of Work:

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.

--Timothy Gow

Appendix:

1. Screenshot of the web page.



2. Code can be found on my github page at: https://github.com/Choyero/Arduino-Traffic-Light.git