Lab 1 - WiFi-Controlled LED Stoplight

Online Links

This lab is available on my personal website at: http://AaronNelson95.com/IT441Lab1.php
The code is also available on my private GitHub repository at https://github.com/AaronNelson95/IT441

Objective

The purpose of this lab is to learn the basics of GPIO pins and simple circuits by creating a WiFi-controlled stoplight. This stoplight is connected to GPIO pins on the Raspberry Pi. The Pi broadcasts a webpage which allows a user to choose the color they would like the light to turn, and the server will adjust GPIO pins to reflect their choice. The page will also offer an "auto" option, which will rotate through the three light colors. This lab will help one learn:

- How basic state diagrams work in developing and understanding a project
- How to utilize GPIO pins on a Raspberry Pi to turn on and off lights
- How to broadcast a webpage from a Raspberry Pi that will run server-side scripts

Materials

To complete this lab, you will need:

- A Raspberry Pi that contains WiFi and GPIO pins
- A Raspberry Pi power supply
- A breadboard
- A traffic light LED display module
- 4 Female-Male Jumper Wires
- A computer or an HDMI cable/USB keyboard/Mouse for connecting to the Pi

The LED traffic light

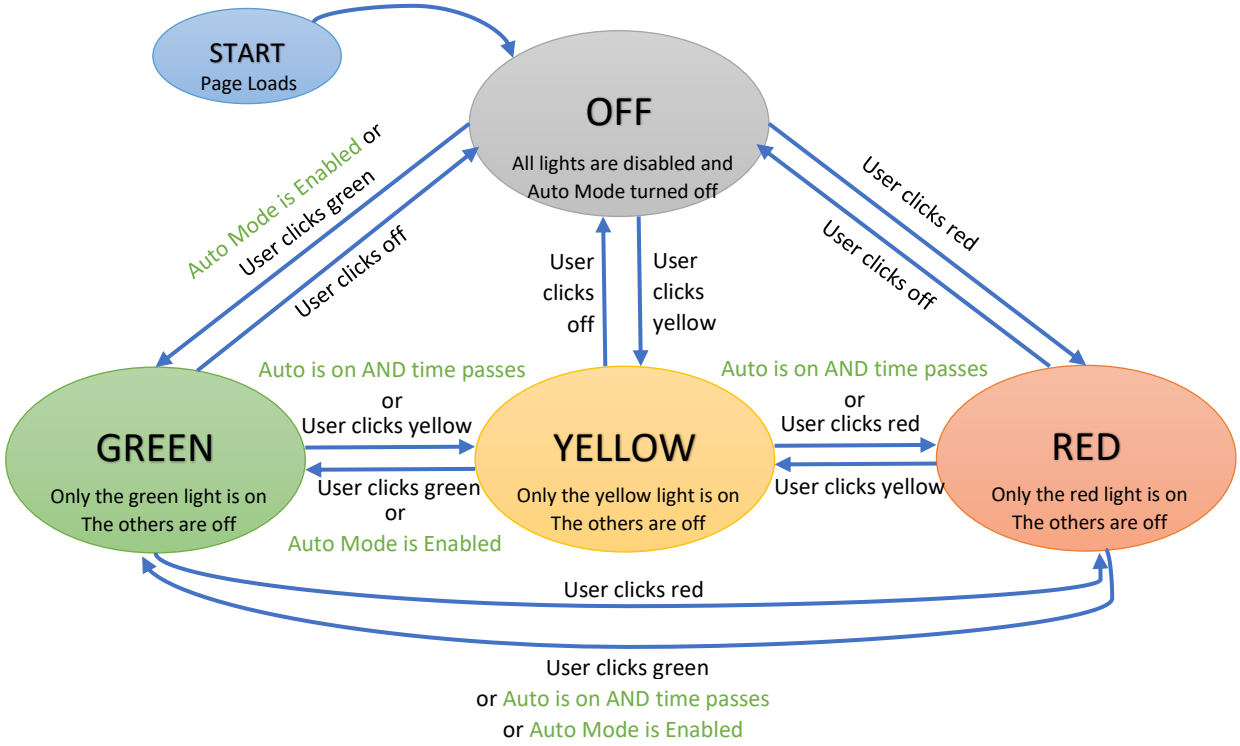
References

The following links may be helpful throughout this lab:

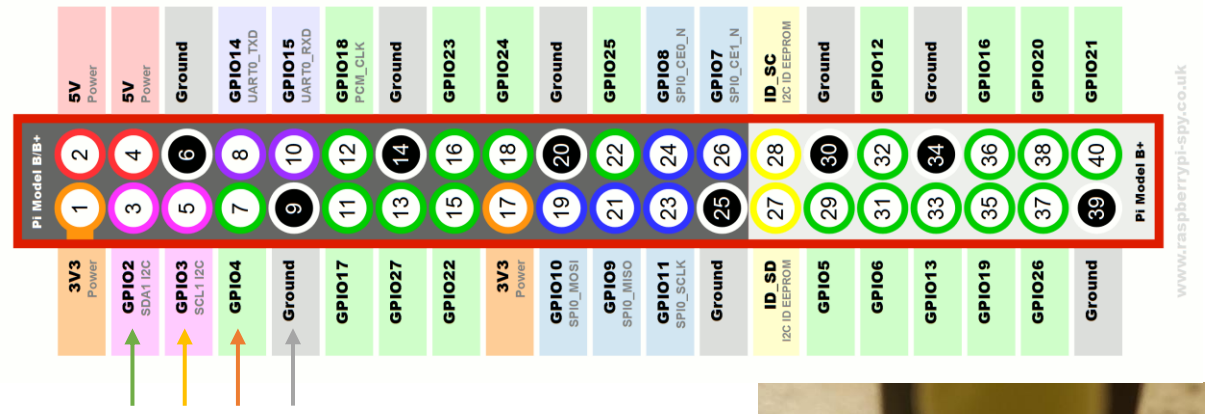
- https://www.w3schools.com/ - A great reference for coding HTML, CSS, JavaScript, and Ajax. This is a great help for designing your webpage.
- https://www.raspberrypi.org/downloads/raspbian/ - Installation instructions for initially setting up the Raspberry Pi with the Raspbian OS.
- https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md - Putting Apache Web Server on a Raspberry Pi. Used for broadcasting a website.
- https://www.instructables.com/id/Simple-and-intuitive-web-interface-for-your-Raspbe/ - A helpful guide on implementing a simple web interface on a Raspberry Pi. It teaches how to wire various lights, communicate with their related pins, and write a php script to run system terminal commands.
- https://www.youtube.com/watch?v=EAMLwbShFFQ – A useful video that teaches how to write PHP scripts to affect GPIO pins, and how to run Ajax commands to run those scripts on the server.

Procedures

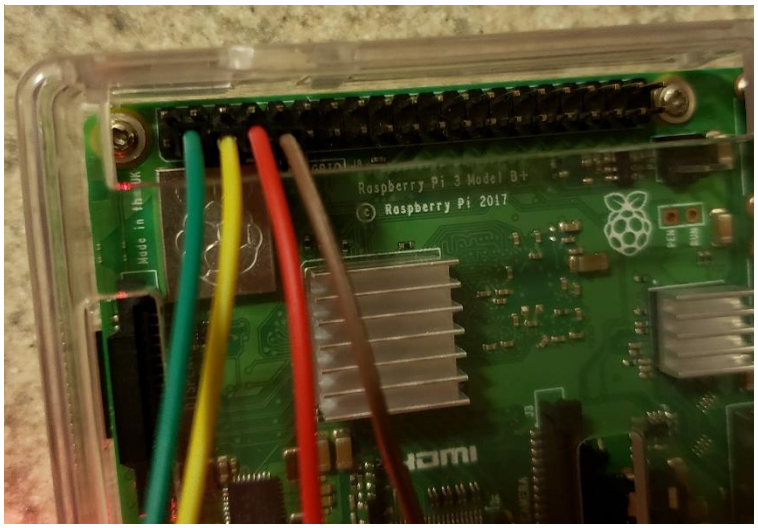
1. Setup the Raspberry Pi if this has not been done this before. This project assumes your Raspberry Pi is already setup with a web server running on it (skip this step if it is configured to host webpages).
    a. Install the Raspbian operating system (or one you are comfortable with). For Raspbian, follow the instructions at https://www.raspberrypi.org/downloads/raspbian/.
    b. Install the Apache2 Web Server. Instructions can be found at https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md.
2. Design a state diagram to understand the logical flow of the system. This can help one to understand how the program will function and respond to user controls.
    a. This page will include 4 light states: when no lights are on, when the green light is on, when the yellow light is on, and when the red light is on. There is also an "auto" mode that, when enabled, will rotate between the green, yellow, and red lights after time passes.

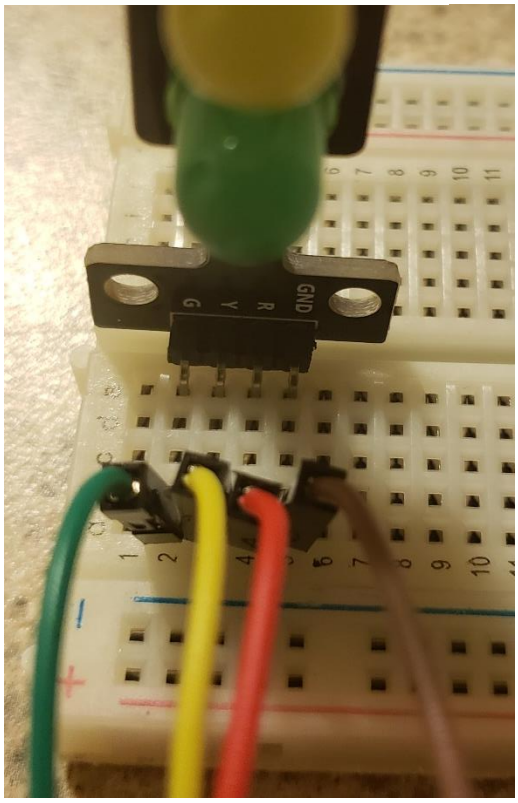


    b. When the page loads (on startup), the device goes to the OFF state. In this state, all of the GPIO pins for the three lights will be set to off. Additionally, if the user previously started auto mode, going to this mode will stop that process (and cancel the related timers).
    c. Simply, when a user selects one of the color options on the webpage, go to that respective state. Also, if they select the off option, move back to the off state.
    d. Regardless of the current state, if auto mode is enabled, move to the green state. Here, timers are set. If auto mode is still on (the off state has not been selected by the user) and so much time passes, move to the yellow state. Now, if auto is still on and the time set passes, move to the red state. Once again, if time passes and auto is still on, move back to the green state and repeat the cycle.
3. Connect the Raspberry Pi to the proper physical components on the breadboard. In this project, the green light will be connected to GPIO2 (physical pin 3), the yellow light is connected to GPIO3 (physical pin 5), the red light is connected to GPIO4 (physical pin 7), and ground is connected to a GRD pin (such as physical pin 9).

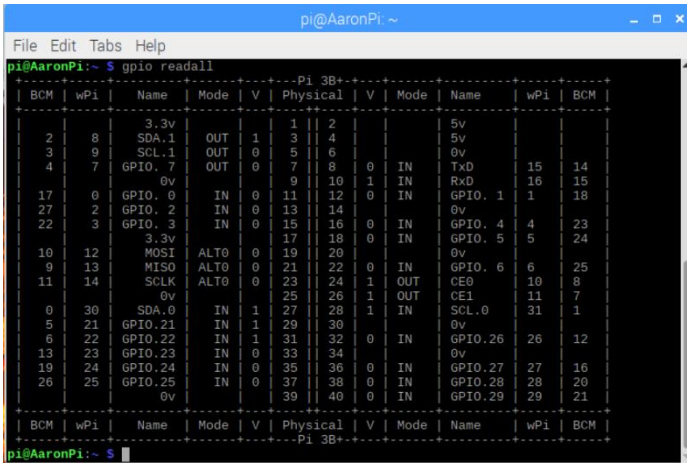


The green jumper wire goes from GPIO2 (pin 3) into the stoplight G
The yellow jumper wire goes from GPIO3 (pin 5) into the stoplight Y
The red jumper wire goes from GPIO4 (pin 7) into the stoplight R
The brown jumper wire goes from Ground (pin 9) into the stoplight GND





4. Test the GPIO Pins and learn how they work.
    a. On the Raspberry Pi, open a terminal and type the command "$ gpio readall". This shows you the current state of all the GPIO pins. BCM is the GPIO# number (in our case, we want BCM 2 to be the green light). Mode tells whether the device is OUTputting data (such as sending an on/off signal *to* a light) or INputting information (such as reading *from* a sensor).



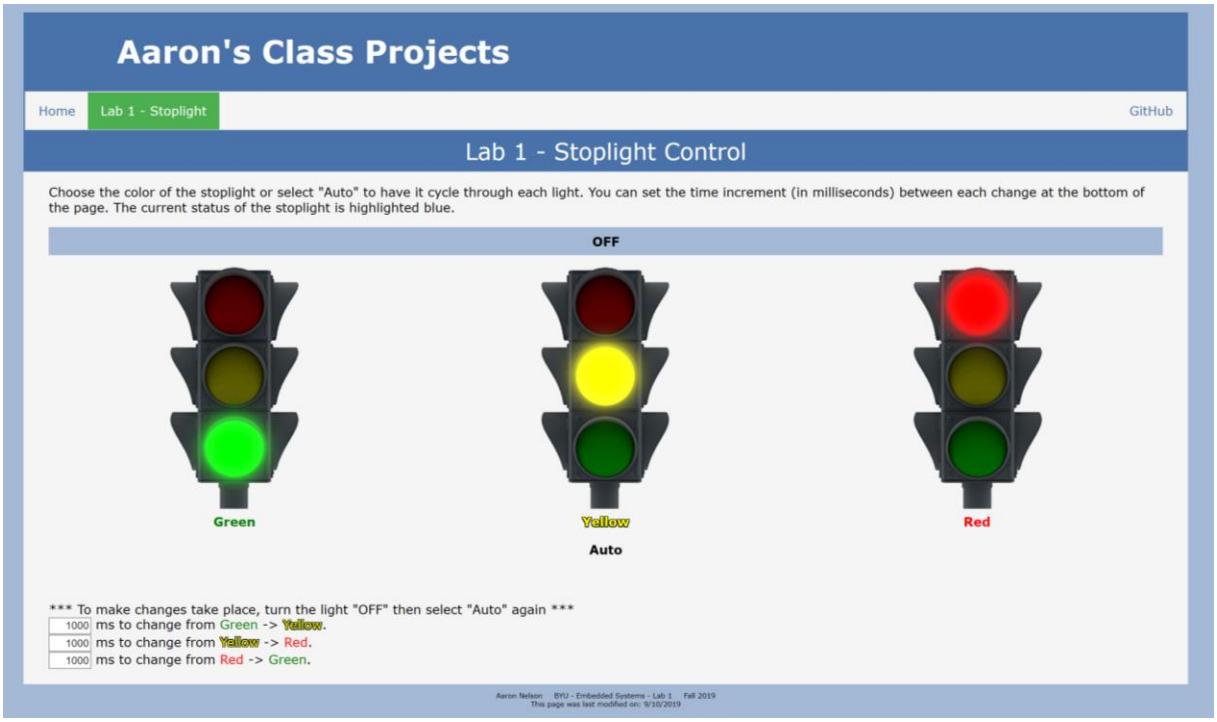
    b. You can change the mode with the following command: "$ gpio -g mode 2 out", where this will change BCM pin 2 to output mode. This will set the pin the green light is on to listen for a signal.

c. To actually turn the light on, you need to enable its voltage. Run the command: "$ gpio -g write 2 1" to set the BCM pin 2 to have the V value of 1, allowing a true value and letting power go through the pin. To turn it back off, simply change the final number to 0.

d. Explore around with changing the mode and values in the table. Try to turn on and off each physical stoplight light.

5. Develop the webpage the user will use to select a light.

a. Using HTML and CSS, design a webpage for a user to select the mode they would like. This can be nicely done using a table and images. Be sure to give each option a unique id.



b. Design a JavaScript page that contains object listeners. When a user clicks on an option on the webpage (such as Green), it will run a specific JavaScript function (such as runGreen()). These functions will utilize Ajax later to run PHP scripts.

c. Develop separate php scripts for the off, green, yellow, and red modes (save as unique files). PHP runs requests on the server, where your Pi is broadcasting the page. You can simply enter your terminal commands (from step 4) within the "System()" function:

```
<?php
    system('gpio -g mode 2 out);
    …
    system('gpio -g write 2 1');
?>
```

d. Now that you have scripts containing your terminal code, you are ready to write Ajax requests that point to the code. These are in the form of XMLHttpRequest() and will "run" your script selected ("Scripts/StoplightGreen.php" should be the location of the green script written in the step before).

```
document.getElementById("green").addEventListener("click", runGreen);

function runGreen(){
    $(document).ready(function(){
        var request = new XMLHttpRequest();
        request.open("GET","Scripts/StoplightGreen.php");
        request.send();
    });
}
```

   e. Now that each of the lights are set to respond to the webpage's button, and their functions push an Ajax request to run that script, develop functionality for the auto button. Remember from the state diagram, once this is pressed, it should run a function that will start with green (run the green light function), wait a set amount of time then change to yellow (run the yellow function), wait more time, change to red (run the red function), then after more time passes, repeat the process (call this auto cycle function again).

6. Now that the pages are complete, if you haven't already, save all data to Apache 2 server folder on the Raspberry Pi to host it. "/var/www/html" should contain all of your pages and scripts.
7. Find the IP Address of your Raspberry Pi with the command "$ ifconfig" (it may be under "wlan0" and the value after "inet"). It will likely start with 192.168, with two more values after that. On another computer or phone connected to the same WiFi, type that value into an internet browser followed by a slash and your webpage's saved name, such as: http://192.168.1.93/StoplightCommand.php (where 192.168.1.93 is my Pi's IP Address, and StoplightCommand.php is my HTML webpage). Click the buttons on your site and test that it works. You may need to go through additional testing and troubleshooting if the lights don't work as expected. Often, the F12 developer tools may help you troubleshoot if there is an error in your code syntax.

Thought Questions

1. What language did you choose for implementing this project? Why?

   I used HTML to write my Raspberry Pi's webpage for the project, and I incorporated CSS and JavaScript with it. However, these are all *client*-side languages so they will work on a user's computer but cannot run scripts on my server (which is the Pi connected to the stoplight). The GPIO pins turn on and off with simple terminal commands. I wanted to implement a language that can simply run these direct commands (such as Python). However, with the existing website architecture I had, I had to use a workaround. Ultimately, my light-changing "scripts" are written in PHP. These scripts simply use the "system()" function to run terminal commands. For a client user to run these scripts, the JavaScript functions which run when they click a light option on the webpage triggers a JavaScript Ajax XMLHttpRequest to run them. I chose these languages because I had experience with them in my IT210 class.

   The overall project makes use of HTML (that contains a .php page extension simply in case I wish to expand the site with an adjustable navigation bar), CSS, JavaScript, and Terminal Commands (that are contained in PHP "scripts" because PHP runs on the server-side, and these are activated through JavaScript's Ajax capabilities).

2. What is the purpose of the resistor in this simple circuit? What would happen if you omitted it?

   Our traffic light modules contained resistors built in for each light. A separate resistor was not necessary for this project. However, in the case that simple, separate LED lights are used, a resistor should be used in each circuit. Typically, a resistor is used to limit the current that passes through the LED (by converting it to heat) and this prevents the LED from burning out (http://www.resistorguide.com/resistor-for-led/). Depending on the power passing through the LED, if the resistor was omitted, it will appear brighter comparatively, but it will eventually burn out and die faster. If there is too much power passing through, the LED may just completely burn out instantly. Although a resistor may dim the light a little, it will greatly help with the LED's usable life.

3. What are practical applications of this device? What enhancements or modifications would you make?

   This device currently does not have very much of a practical application. However, it does teach a good foundation for future projects and ideas. This project helped us learn how to make and use WiFi controls that will run scripts to alter the state of GPIO pins. Concerning the modules utilized, the 3 LED stoplight doesn't have much use (and this probably couldn't be implemented as a useful life-size stoplight), but other devices and

outputs can be connected instead. For example, multiple GPIO pins can be utilized to design a simple number counter like a digit on a digital clock. Multiple brighter lights can also be used to light up a room. For example, if you have trouble waking up in the morning when it is dark, you can set your Pi to act as an alarm that will brightly display lights next to your bed, right when your clock starts buzzing. Web-based communication has great implications as well. You won't have to be directly connected to your Pi to be able to run scripts from it! Rather, you can use any device anywhere in your home to perform various functions. This can let you use GPIO functions with a headless Pi.

For my personal learning and enjoyment, I did work on some more advanced features with this project. I focused a lot on the website and design to make it nice and presentable. I utilized JavaScript so when a light was lit on the Pi, the button on the website is also highlighted to reflect it as well. With this feature, even if you are in a different room, you can know exactly which light is turned on at the time. I also allowed an option for the user to type in how many milliseconds they would like for the light to change between the green->yellow, yellow-> red, and red->green transitions in auto mode. They can make the light change as slow as they would like, or they can have it blink and flicker really fast by setting these values to only 50 ms, for a fun effect. I already implemented all of the enhancements I would make with this device, and now I look forward to other devices I can make that would build from this project's foundation.

4. Estimate the time you spent on this lab and report.

This is my first project in a while that involved programming, so I spent a large amount of my time with remembering how to design a webpage. I spent about 30 minutes with trying to plan the project, design my state diagrams, and think about how I would like to implement them. One day, I spent around 4 hours designing the hosted webpage and a couple more hours the next day completing the client-side functionality. I still did not have the jumper wires needed for the project, but I tried to implement the scripts by examining the "$ gpio readall" command on the Pi. This was the hardest part for me, because I was not aware that everything I had done up to this point was strictly client-side programming only, and that the GPIO pins needed to be ran on the server. Here, I tried to look at different languages, such as Python, and how/if I could implement them. Eventually, I learned that the client can run an Ajax request to run a PHP script on the server. I spent around 6 hours on this, working hard at the challenge until 5am. The next day, I finally got the jumper wires I needed. I connected my project up within minutes, and my webpage instantly worked to light up the physical lights like I expected! I spent an extra hour or two with commenting my code and making small adjustments for the pages after that.

All in all, I spent around 15 hours on the webpage and script functionality, and about 7 hours writing this lab report.

Certification of Work

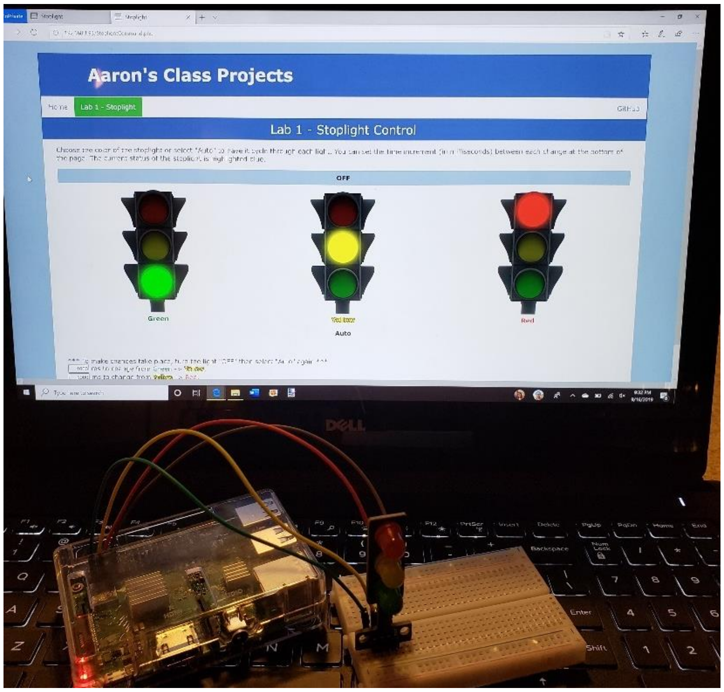
I certify that this lab represents my own work. I used various web resources to remind me certain code syntax, but I did not copy anything that was not basic knowledge unless I specified it inside of my code comments. I did base my website design from a project I made 3 years ago in a Web Design class, but even that was of my own work. I did use three images on my page that were obtained from the internet, and that web resource is linked to in my code comments.

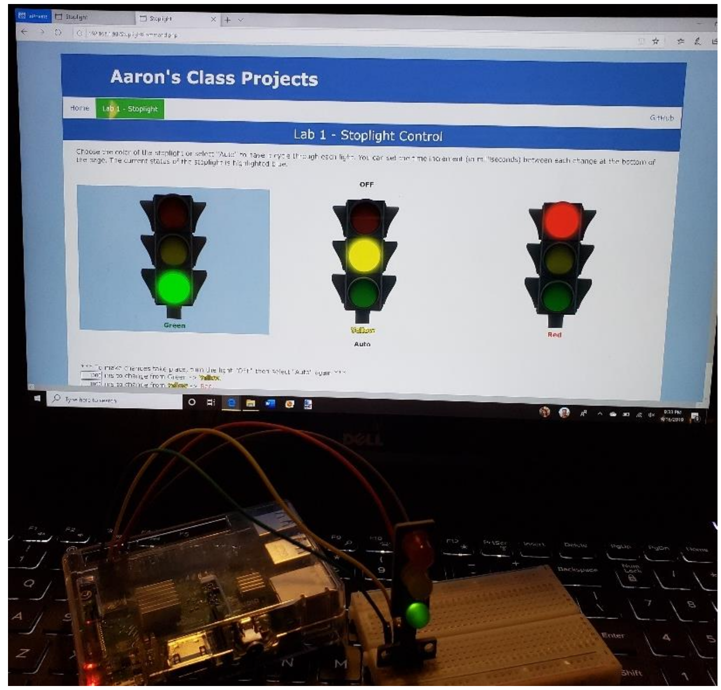
-- Aaron Nelson

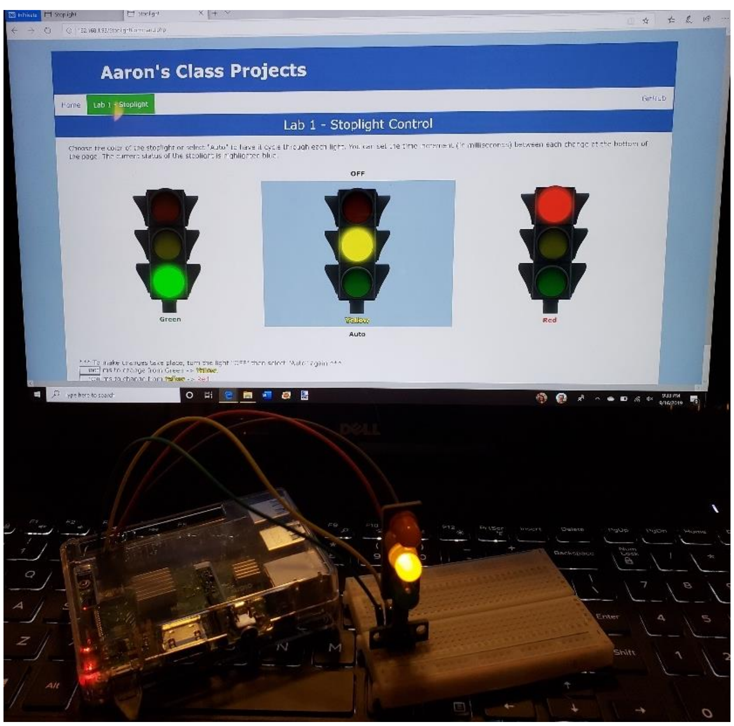
Appendix

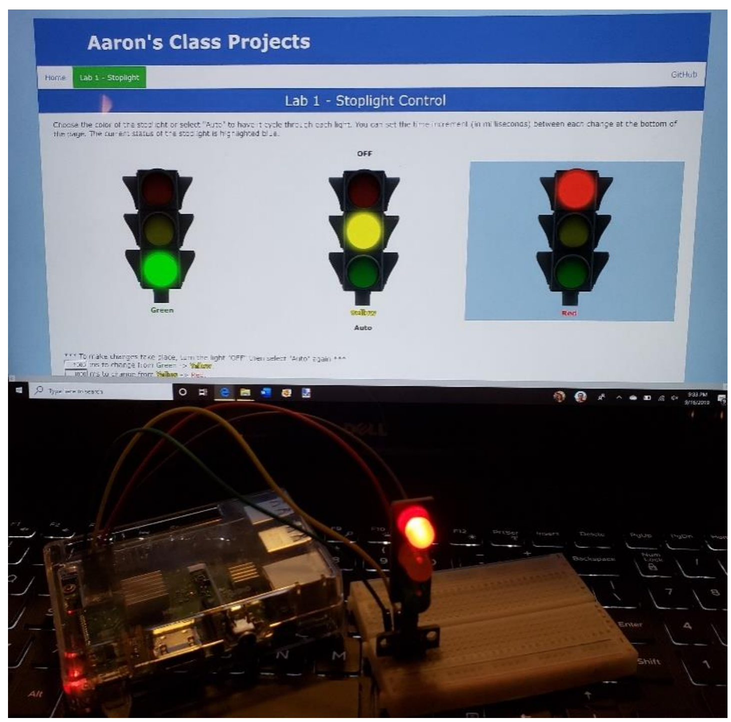
Images of Final Product

As can be seen in the images below, when a light is selected on the webpage, the respective light will turn on. Auto mode will also rotate around between the green, yellow, and red lights.

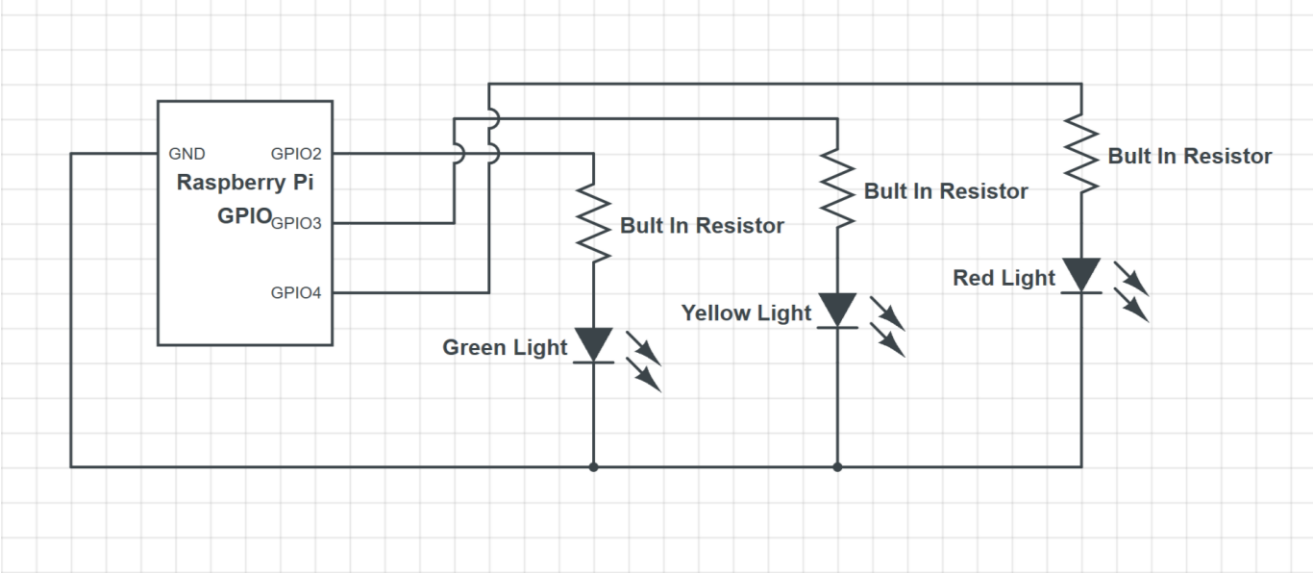








Circuit Diagram



Code (available at https://github.com/AaronNelson95/IT441)

Webpage Code

stoplightCommand.php

```
<!doctype html>
<html>
    <head>
        <!--Aaron Nelson
        Last Updated: 9-10-19
        A page written for lab 1 of IT441 that simply allows you to select a stoplight color
        (or have it automatically rotate through them) and it reflects this change with GPIO
         pins. -->
        <title>Stoplight</title>
        <link rel="stylesheet" href="Styles/Styles.css" type="text/css">
    </head>
    <body onload="offClicked()"> <!-- When the page first loads, initialize all of the pins by
                                    setting their mode to output and turning them all off. -->
        <div id="wrapper">
            <header>
                <h1>Aaron's Class Projects</h1>
            </header>
                <nav><ul>
                <li><a href="index.php" id="homelink">Home</a></li>
                <li><a href="StoplightCommand.php" id="lab1link" class="active">Lab 1 -
                                                                        Stoplight</a></li>
                <li style="float:right"><a href="https://github.com/AaronNelson95/IT441"
                                        target="_blank" id="githublink">GitHub</a></li>
            </ul></nav>
            <div id="pageTitle">Lab 1 - Stoplight Control</div>


            <div id="main">
                <p>Choose the color of the stoplight or select "Auto" to have it cycle through
                   each light. You can set the time increment (in milliseconds) between each
                   change at the bottom of the page. The current status of the stoplight is
                   highlighted blue.  </p>

                <!-- This table has the 3 stoplight colors, as well as an associated image,
                   sorted along with an off option and an auto option. JavaScript responds to
                   the page clicks to highlight the table cell currently running, and to run a
                   script that alters the GPIO pins. -->
                <table class=SpotlightColors>
                    <tr>
                        <td id="off" class="running" colspan=3><h4 class="off">OFF</h4></td>
                    </tr>
                    <tr>
                        <!-- Stoplight images from https://www.freemanhealth.com/blog/the-
                              behavior-traffic-light-colors-of-the-light -->
                        <td id="green"><img src="Images/GreenLight.png"><br><h4
                                                                    class="green">Green</h4></td>
                        <td id="yellow" ><img src="Images/YellowLight.png"><br><h4
                                                                    class="yellow">Yellow</h4></td>
                        <td id="red"><img src="Images/RedLight.png"><br><h4
                                                                    class="red">Red</h4></td>
                    </tr>
                    <tr>
                        <td id="auto" colspan=3><h4 class="auto">Auto</h4></td>
                    </tr>
                </table>
```

```
            <!-- This allows the user to input the number of ms they wish to wait until
                    the light changes to the next color. -->
            *** To make changes take place, turn the light "OFF" then select "Auto" again
              ***<br>
            <input type="number" name="GY" id="GY" class="colorPick" value=1000 /> ms to
                    change from <span class="green">Green</span> -> <span
                    class="yellow">Yellow</span>.
            <br>
            <input type="number" name="YR" id="YR" class="colorPick" value=1000 /> ms to
                    change from <span class="yellow">Yellow</span> -> <span
                    class="red">Red</span>.
            <br>
            <input type="number" name="RG" id="RG" class="colorPick" value=1000 /> ms to
                     change from <span class="red">Red</span> -> <span
                     class="green">Green</span>.
            <br><br>


        </div>

        <footer>Aaron Nelson   BYU - Embedded Systems - Lab 1   Fall 2019
        <br>This page was last modified on: 9/10/2019
        </footer>
    </div>

    <script src="Scripts/StoplightScripts.js"></script>
    <script src="http://code.jquery.com/jquery-1.11.0.min.js"></script> <!-- Allows the
        ajax scripts to work (which pushes a request for the GPIO color scrips to run) -->
</body>
</html>
```

CSS Stylesheets (located in 'Styles' folder)

Styles.css

```
/* CSS Styles for Embedded Systems Projects
    Aaron Nelson
    Last Updated: 9-10-19
    Note- I originally built the basis of this style design in my COSC 2360 Web Page Dynamics
      & Scripting class at Western Wyoming Community College during the Spring 2017 Semester.
*/


/* UNIVERSAL STYLES */

html {overflow: -moz-scrollbars-vertical; overflow: scroll;} /* To always show the scroll bar
                        so the widths of the pages can remain consistent across the site */

/* Overall Page Structure */
body {
    background-color: #a4b9d5;
    color:#000000;
    font-family: Verdana, Arial, serif;
}
#wrapper {
    background-color: #F5F5F5;
    width: 90%;
    margin-top: 25px;
    margin-left: auto;
    margin-right: auto;
    min-width: 600px;
    max-width: 1480px;
}



/* Header Appearance (Site Title) */
header {
    text-align: left;
    background-color: #4972aa;
    height: 100px;
}
header h1 {
    padding-left: 8%;
    padding-top: 0.6em;
    color: white;
    font-size: 2.5em;
}



/* Navigation Buttons */
nav {
    border-left: 2px solid #4972aa;
    border-right: 2px solid #4972aa;
}
nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    border: 1px solid #e7e7e7;
```

```
    background-color: #f5f5f5;
}
nav li {
    float: left;
}
nav li a {
    display: block;
    color: #666;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
li a:hover:not(.active) {
    background-color: #ddd;
}
li a.active {
    color: white;
    background-color: #4CAF50;
}



#pageTitle {
    font-size: 1.8em;
    font-style: bold;
    background-color: #4972aa;
    color: white;
    padding: 0.3em;
    text-align: center;
}
#main {
    padding-left: 2em;
    padding-right: 2em;
    display: block;
}



/* Link Colors */
a:link {color: #4972aa; }
a:visited {color: blue; }
a:hover {color: #4CAF50; }
h1 a {text-decoration: none; color: white;}
h1 a:link {color: white; }
h1 a:visited {color: white; }
h1 a:hover {color: white; }



/* Footer Format */
footer {
    text-align:center;
    background-color: #a4b9d5;
    font-size: .55em;
    padding: 1em;
}
```

```
/* SPECIFIC WEBPAGE STYELS */



/* LAB 1 */
/* Spotlight Table */
.SpotlightColors {
    width: 100%;
    border: 0;
    border-collapse: collapse;
    margin-bottom: 3em;
}
.SpotlightColors td, th {
    border: 0;
    padding: 5px;
    text-align: center;
}
h4 {
    margin-top: 3px;
    margin-bottom: 3px;
}
/* Spotlight Table Text */
.green {
    color: green;
}
.yellow {
    color: yellow;
    /* Yellow text is not readable so it needs a black outline (using a text-shadow as an
            outline idea is from https://wpshout.com/quick-guides/create-text-outline-css/ )*/
    text-shadow:
        -1px -1px 0 #000,
        1px -1px 0 #000,
        -1px 1px 0 #000,
        1px 1px 0 #000;
}
.red {
    color: red;
}
.auto, .off {
    text-align: center;
}
#green:hover, #yellow:hover, #red:hover, #auto:hover, #off:hover {
    /* Change the color of what is being selected */
    background-color: #ddd;
}
.running {
    background-color: #a4b9d5;
}
.autoOn {
    background-color: #a4b9d5;
}
.colorPick {
    width: 50px;
    text-align: right;
}
```

JavaScript Files (located in 'Scripts' folder)

StoplightScripts.js

```
/* Lab 1 Spotlight Control Scripts */
var autoIsOn = 0;    // When this is a 1, the user clicked the auto button so the lights should
                        be rotated through
var GY = 1000;       // Preset values for the auto button's change between the light colors (1
                        second)
var YR = 2000;
var RG = 3000;

document.getElementById("off").addEventListener("click", offClicked);
function offClicked(){
    $(document).ready(function(){
        // When the user clicks the off button, run the off script.
        var request = new XMLHttpRequest();
        request.open("GET","Scripts/StoplightOff.php");
        request.send();
    });

    // Find which element is currently running and stop it. Then give the off button the
          running class.
    /* Help with removing a class from multiple elements is here:
      https://clubmate.fi/remove-a-class-name-from-multiple-elements-with-pure-javascript/ */
    let x = document.getElementsByClassName("running");
    while (x[0]){
        x[0].classList.remove("running");
    }
    document.getElementById("off").className="running";
    stopAuto();
}


document.getElementById("green").addEventListener("click", greenClicked);
function greenClicked(){
    runGreen();
    stopAuto();      // If auto is running, make it stop
}


document.getElementById("yellow").addEventListener("click", yellowClicked);
function yellowClicked(){
    runYellow();
    stopAuto();      // If auto is running, make it stop
}


document.getElementById("red").addEventListener("click", redClicked);
function redClicked(){
    runRed();
    stopAuto();      // If auto is running, make it stop
}


document.getElementById("auto").addEventListener("click", autoClicked);
function autoClicked(){
    if(autoIsOn == 1){alert("Please turn the light off to change the time increments.");
return;};  // For some reason if auto is clicked multiple times while it is already running,
             the timing gets extremely messed up. This stops the button from working again.
```

```
    getTimerValues();   // Obtains the preset values or the ones the user entered for the time
                                                        that passes between the lights
    let x = document.getElementsByClassName("running");
    while (x[0]){
        x[0].classList.remove("running");
    }
    document.getElementById("auto").className="autoOn";
    autoIsOn = 1;
    runCycle();         // Rotates through the light colors using the timer values that were
                                obtained
}

function getTimerValues() {
    GY = parseInt(document.getElementById("GY").value);
    // The next two values adds the previous ones because the setTimeout function will run
            from the start (so if they were all 1000ms, then they will all be triggered to
            turn on at the same time, we want them to rotate)
    YR = parseInt(document.getElementById("YR").value) + GY;
    RG = parseInt(document.getElementById("RG").value) + YR;
}

function runCycle(){
    // This begins with the green light and then after the set amount of time passes, it will
            show the next color. At the end, it runs this function again.
    runGreen();
    timey = setTimeout(function(){
        runYellow();
    }, GY);
    timer = setTimeout(function(){
        runRed();
    }, YR);
    timeg = setTimeout(function(){
        runCycle();
    }, RG);
}

function stopAuto(){
    // This will stop auto from running. It turns off the variable that says it is on, and it
            clears the timers that are running (otherwise they will still run one final time)
    document.getElementById("auto").classList.remove("autoOn");
    autoIsOn = 0;
    clearTimeout(timey);
    clearTimeout(timer);
    clearTimeout(timeg);
}



/* These functions work just like the off function does. When that light color is selected, it
sends an ajax request to run the associated script (JavaScript works on the client side and we
need the php to run on the server side (where the pi is connected), these requests makes that
possible). After the request sends, the light will turn on by changing settings of the GPIO
pins. It removes the previous cell on the page that was highlighted, and it instead highlights
the current color. */
function runGreen(){
    $(document).ready(function(){
        var request = new XMLHttpRequest();
        request.open("GET","Scripts/StoplightGreen.php");
        request.send();
```

```
    });

    let x = document.getElementsByClassName("running");
    while (x[0]){
        x[0].classList.remove("running");
    }
    document.getElementById("green").className="running";
}

function runYellow(){
    $(document).ready(function(){
        var request = new XMLHttpRequest();
        request.open("GET","Scripts/StoplightYellow.php");
        request.send();
    });

    let x = document.getElementsByClassName("running");
    while (x[0]){
        x[0].classList.remove("running");
    }
    document.getElementById("yellow").className="running";
}

function runRed(){
    $(document).ready(function(){
        var request = new XMLHttpRequest();
        request.open("GET","Scripts/StoplightRed.php");
        request.send();
    });

    let x = document.getElementsByClassName("running");
    while (x[0]){
        x[0].classList.remove("running");
    }
    document.getElementById("red").className="running";
}
```

PHP Scripts (located in 'Scripts' folder)

StoplightOff.php

```php
<?php
    // This script turns off the GPIO pins associated with green, yellow, and red. It keeps
       the mode as "out" because when set "in", some lights may get random voltage coming
       through, weakly lighting the light.
    system('gpio -g write 2 0');
    system('gpio -g write 3 0');
    system('gpio -g write 4 0');
    system('gpio -g mode 2 out');
    system('gpio -g mode 3 out');
    system('gpio -g mode 4 out');
?>
```

StoplightGreen.php

```php
<?php
    // This script turns on the green light (if set to pin 2) and turns off the other ones.
    system('gpio -g mode 2 out');
    system('gpio -g mode 3 out');
    system('gpio -g mode 4 out');
    system('gpio -g write 2 1');
    system('gpio -g write 3 0');
    system('gpio -g write 4 0');
?>
```

StoplightYellow.php

```php
<?php
    // This script turns on the yellow light (if set to pin 3) and turns off the other ones.
    system('gpio -g mode 2 out');
    system('gpio -g mode 3 out');
    system('gpio -g mode 4 out');
    system('gpio -g write 2 0');
    system('gpio -g write 3 1');
    system('gpio -g write 4 0');
?>
```

StoplightRed.php

```php
<?php
    // This script turns on the red light (if set to pin 4) and turns off the other ones.
    system('gpio -g mode 2 out');
    system('gpio -g mode 3 out');
    system('gpio -g mode 4 out');
    system('gpio -g write 2 0');
    system('gpio -g write 3 0');
    system('gpio -g write 4 1');
?>
```

Other Code

I also had a webpage for the index page of the site, a php navigation bar, and a script to automatically update the date printed on the page based on when the page was last modified. However, these are unrelated and unnecessary for this project, and I included the navigation bar directly in the above code. It is still separated on my GitHub page, where all of these pages are also listed. I also had an "Images" folder, that contained the three stoplight images seen on my page.