# Lab 1 - Stoplight on Raspberry Pi

#### Online Link

The details and code to this lab can be found at: https://github.com/Rhemsley/Stoplight-Lab.git

#### Objective

The purpose of this lab is to learn how to use the Raspberry Pi GPIO Pins to create a web server controlled Stoplight. This teaches principles like the following:

* Create circuits using a resistor, LED and the GPIO pins of a Raspberry Pi
* Setup a Raspberry Pi with an Apache Web Server and PHP
* Using the Apache Web Server, control each of the circuits and create a Looping effect like a Stoplight.

#### Materials

I used the following materials to complete this lab:

Raspberry Pi

4 x Male to Female Jumper Cables

1 x Red LED

1 x Yellow LED

1 x Green LED

3 x 100 Ω 5% Resistor (BrBlBrGd)

1 x Breadboard

#### References

I used the following resources to complete this lab:

<http://wiringpi.com/the-gpio-utility/> All GPIO commands explained here

<https://www.w3schools.com/php/default.asp> PHP coding structure relearned from here

<https://pimylifeup.com/raspberry-pi-gpio/> Skim reading for how the Raspberry Pi works with the GPIO pins

<https://www.php.net/manual/en/function.shell-exec.php> Shell\_exec help

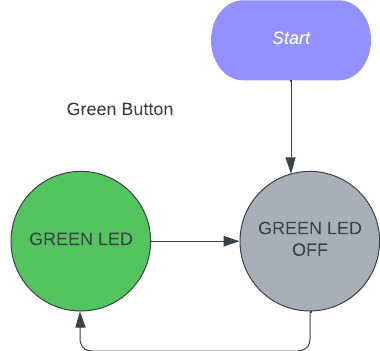
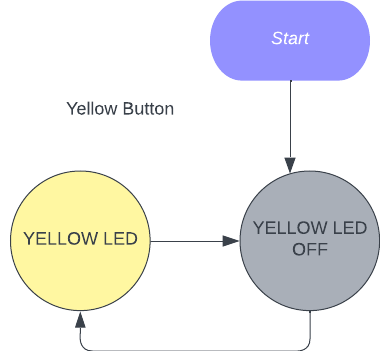
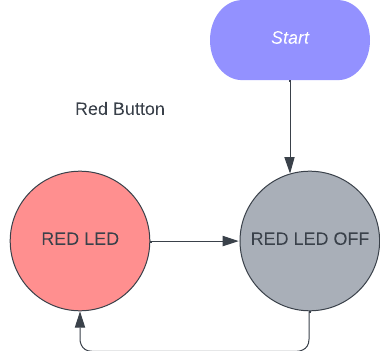
<https://www.faqcode4u.com/faq/550226/html-button-to-call-php-shell-exec-command> Using shell\_exec with a button

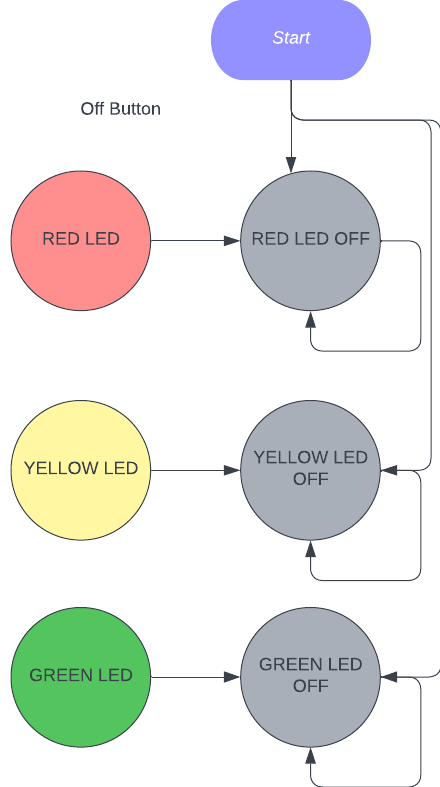
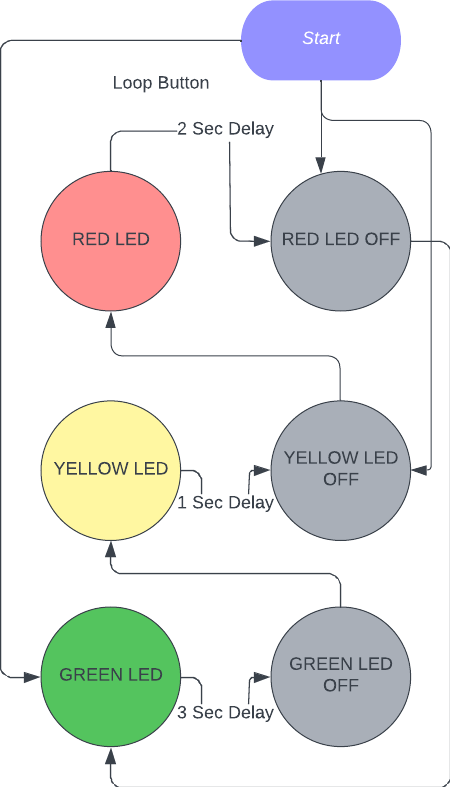
<https://www.makeuseof.com/tag/host-website-raspberry-pi/> Getting apache up and running

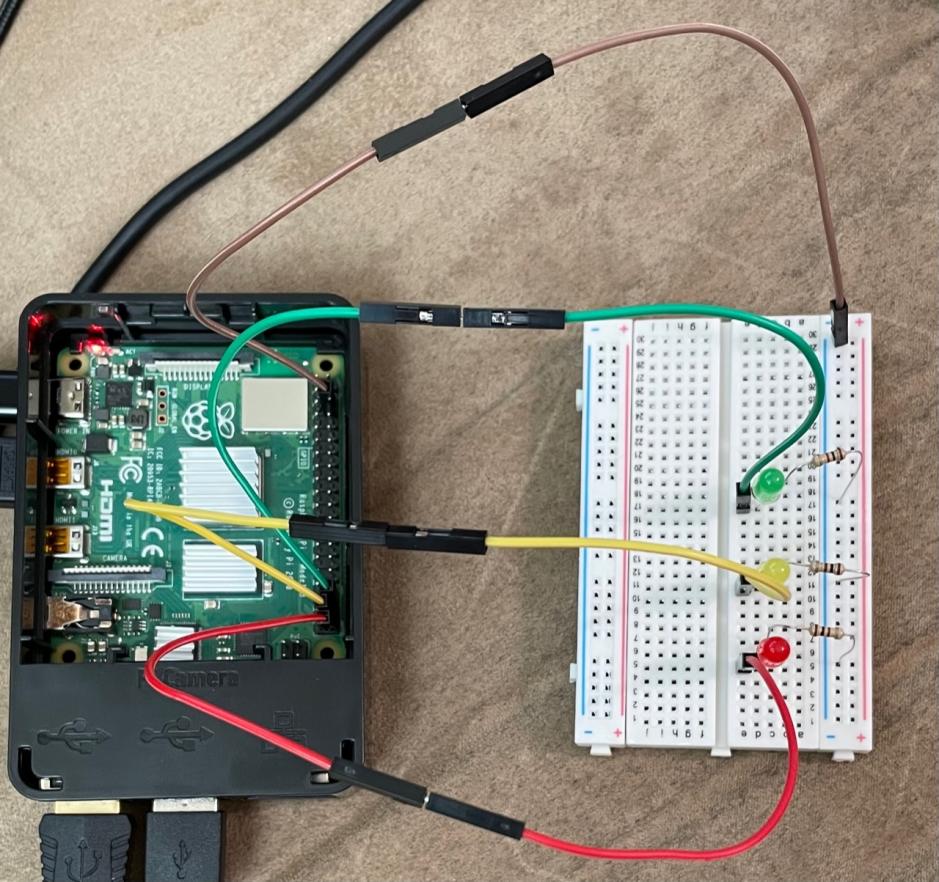
<https://lucid.app/documents#/dashboard> Building Flow Diagrams

#### Baseline Information

Below are the Flow charts for the five different buttons 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. Then the Loop button loops the LEDs from Green to Yellow to Red with identified delays between each one.

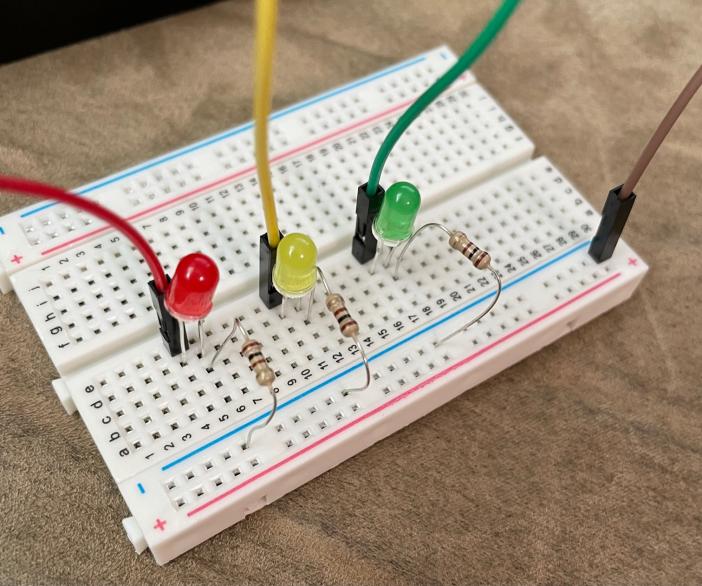
  



To the right we have the overview of the entire build showing the breadboard with the LEDS and Resistors connected to the Raspberry Pi GPIO pins with the 4 wires. The Brown wire is used as the ground wire for each of the circuits going to a ground GPIO pin. Then the other 3 wires make up each of the separate LED circuits with their associated resistor.

Lastly we have two pictures of close up views showing the positioning of everything on the breadboard and where each of the wires are plugged into the GPIO pins.





#### Procedures

1. Setup and access your Raspberry Pie with Raspbian. If not sure how to do that, please see other online tutorials.
2. On your Raspberry Pie Install Apache2 and PHP using the terminal
   1. To install Apache2 type in “sudo apt install apache2” and accept the prompts. If you ever need to restart the web server you can type in “sudo service apache2 restart”. I would recommend doing this after php is installed just to make sure it is running smoothly and as intended.
   2. To install PHP type in “sudo apt install php -y”.
3. Make sure you can use the Wiring Pi library on your Pi. Newer Pi’s like the 4th gen do not have it by default.
   1. In the terminal type “gpio readall”. If you see your GPIO layout printed then you are set. If it says it is an unknown command then you need to install or update Wiringpi which you can do with the following. First type, “cd /tmp” then “wget [https://project-downloads.drogon.net/wiringpi-latest.deb](https://project-downloads.drogon.net/wiringpi-latest.deb" \t "https://usc-word-edit.officeapps.live.com/we/_blank)” and then lastly, “sudo dpkg -i wiringpi-latest.deb” Once updated, you should be able to run the “gpio readall” command and see your GPIO layout. (doing “sudo apt-get install wiringpi” may work though I never tried it.)
4. Set up the breadboard, LEDs and circuit with your GPIO pins.
   1. First build the board and circuit as seen in the picture above. You will have 3 full circuits which use different GPIO wins and the same ground pin to complete the circuit.
   2. You will have the Red circuit going to the bottom right GPIO pin, pin 21 using the BCM numbering and which numbering system we will use for this lab moving forward.
   3. The Yellow circuit will be in pin 20.
   4. And the Green circuit will be in pin 16. As seen, these are the bottom 3 pins.
   5. Your Ground wire will go to the pin third from the top right, right above the green wire, or in any slot that is indicated with 0V in the readall command or as a ground pin.
   6. In doing this, make sure each of the LEDs have the long lead going towards the GPIO pin and the short lead going towards the ground and in this case through the resistor. Also take note that the resistor and second lead of the LED are in a different line than the first lead and the wire going to the board.
   7. Once the board is built, each pin can be tested by first running the command “gpio -g mode PIN\_NUMBER out” and then “gpio -g write PIN\_NUMBER 1” where the -g argument indicates that we are using the BCM numbering and PIN\_NUMBER is the pin number of the circuit you are testing, for example the red circuit with pin number 21 could be tested. Once tested, make sure to run “gpio -g write PIN\_NUMBER 0” to turn the circuit back off.
5. Now to setup the web server and associated files.
   1. First navigate to your web servers site files with “cd /var/www/html”. In here lets first remove the current index page with “sudo rm index.html”.
   2. Now we can replace the the previous index page with the new index.php page and the other needed files from my git repository with “sudo git clone https://github.com/Rhemsley/Stoplight-Lab.git”. If this doesn’t work or you want to make your own files then you can manually recreate the files in this folder yourself with something like “sudo nano index.php” etc.
   3. Once the files are in there, lets check their permissions and make sure those are all correct. First, lets back out of the html folder with “cd ..” then lets check the html folders permissions with “ls -alt”. Your html folder should have your Pi’s username followed by the group of www-data. Now run “sudo chown -R USERNAME:www-data html” with your username put in there to make sure the ownership is correct for the folder and all the files in side of it.
   4. Now navigate into the html folder with “cd html” and then again run “ls -alt.” This time lets make sure the files have the right permissions. To the left of each of the files you should have -rw-r--r-- meaning read and write for owner and then read for the rest. To set all the files to this permission do “sudo chmod 644 \*.php”.
   5. Then we should be done and can now access the site and test out your work.
6. Accessing the site.
   1. On the Raspberry Pi you can access the site by pulling up an internet browser and going to “localhost” or your raspberry pi’s ip address.
   2. On another device on your network you can access the webserver site by going to the Raspberry Pi’s ip address. The Raspberry Pi’s ip address can be found by putting in the command “ifconfig” and then finding the inet IP address in the wlan0 section.
   3. From here you should be able to click the button of the site and see the magic happen. Each of the color buttons can be toggled on or off and then the Loop button will start a loop and the Off button will turn off all lights whenever pushed.
   4. If you are given any errors then it is possible that your file permissions are incorrect and see step 5 parts c-e. Otherwise restart the server or look up the type of error you are having.

#### Observations

I absolutely enjoyed this first real dive into internet of things devices and exploring the beginning possibilities of the Raspberry Pi and its GPIO Pins. I know there will be plently more of this sort of work, exploration, and fun to come even after this class.

I was surprised at how quickly I ran into hours of struggles upon trying to get my Python code to run on an Apache server. Even with exploring several sites and videos online I was still hitting a heavy wall and I think that is because Apache tries pretty hard to put a solid wall between what the user of the web page can do to the server. I was very relieved to learn about shell\_exec in PHP even though it still mean rewriting my entire python code in another language.

#### Thought Questions

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

I first built code to run the stoplight loop in Python using RPi.GPIO. This was very straightforward and I was able to build it in a matter of minutes. I then decided I would implement my web server with Apache as I knew it was extremely common and knew the basics about it. Upon setting up the Apache server, I explored for hours on how to get it to run my Python code to absolutely no avail. I was able to get it to run Python in the browser but it wouldn’t run my python code on the server. After further exploring I found out that PHP could run shell\_exec commands which executed commands on the server as if they were being ran in the terminal. So, I decided to install PHP on the server and rewrite my python code using WiringPi GPIO commands ran using shell\_exec commands. This took quite a while but ended up working.

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

This limits the voltage and current going through the LED to make it run at its expected brightness level and not burn out. If there was no resistor it would be very bright and then burn out quickly.

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

I could see this used in a home or work environment where a simple Pi is running some sort of indicator LED of something and then someone at home or on the team can activate the LED when desired. The looping aspect could be used as some sort of notifier or timer of something needing accomplished at the lit LEDS indication. If I were to upgrade this maybe I would 3d Print a little enclosure for the entire system that looks a bit nicer and hides all the wires.

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

I would estimate I spent like 8 hours coding and configuring and getting it to work and then 5 hours building the documentation and report.

#### 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