Anthony Chavez

EEE 174-CpE 185 Summer 2020

Monday, Wednesday

Lab 3

Raspberry Pi

Dennis Dahlquist

Introduction:

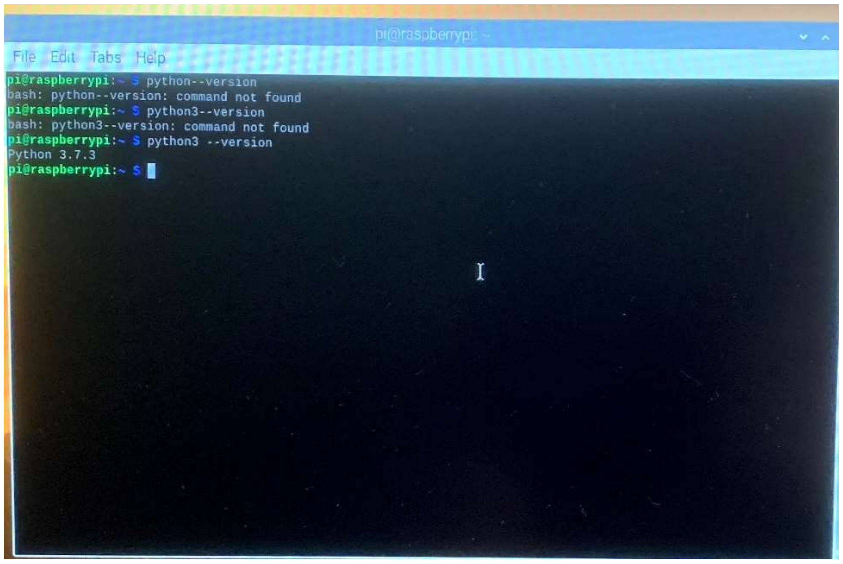
The purpose of this lab is to gain experience with using the Raspberry Pi 4 microcontroller. The Raspberry Pi is an affordable device with many applications in several different fields of interest. This device can host web servers, act as a VPN for your home network, and much more. For this lab, we will explore the capabilities of the Raspberry Pi 4 microcontroller in six sections. First, we will install the operating system on the microSD card. Second, we will write two simple python programs. Third, we will write a simple C program. Fourth, we will build a simple LED circuit with a push button in C and connecting to the microcontroller's GPIO pins. Fifth, we will a simple Web Server. Finally, we will create a simple project of our own that interacts with the GPIO pins or using some network programming.

Part 1: Raspberry Pi Setup

For this section of the lab, we will install the Operating System. Since we purchased the Raspberry Pi 4 Starter Kit, the micro SD card already has the OS preloaded. Once the OS finished installing, we checked to see if the Raspberry Pi was properly connected to the internet.

Part 2: Python On Raspberry Pi

For this section of the lab, we will write two simple python programs. First, we checked if the latest version of Python was installed by typing the command in Figure 2.1. For the first program, we wrote a python program to write "Hello World" to a file when the program is run. We created a folder on the desktop named "Lab3_Part2" and created a new file using the nano text editor in the terminal. We then wrote the code in Figure 2.2, compiled it, and ran it (see Figure 2.3). As for the second python program, we wrote a program to write data to a Comma Separated Values (CSV) file. We created a new file in the nano text editor in the terminal and wrote the code in Figure 2.4. We then compiled the code and ran the program to get the output shown in Figure 2.5.

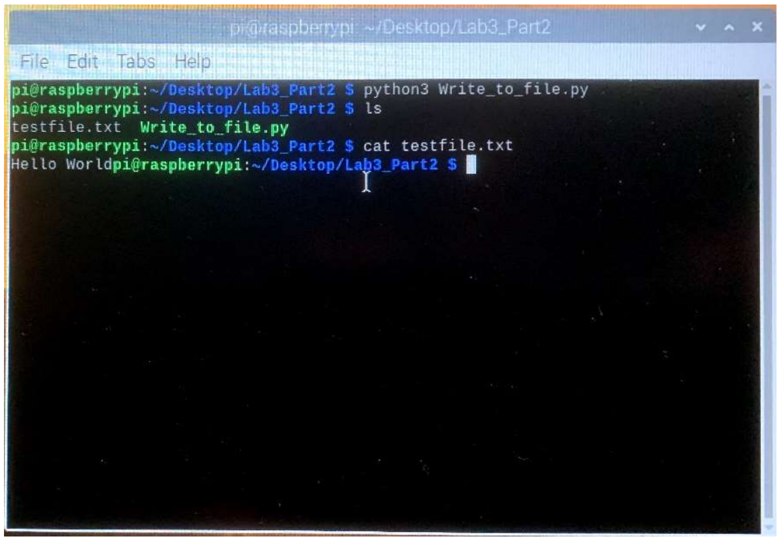


```
pi@raspberrypi:~ $ python--version
bash: python--version: command not found
pi@raspberrypi:~ $ python3--version
bash: python3--version: command not found
pi@raspberrypi:~ $ python3 --version
Python 3.7.3
pi@raspberrypi:~ $
```

Figure 2.1: Python Version command

```
#!/usr/bin/python

file = open(“testfile.txt”,”w”)

file.write(“Hello World”)

file.close()
```

Figure 2.2: ‘Hello World’ Python Code

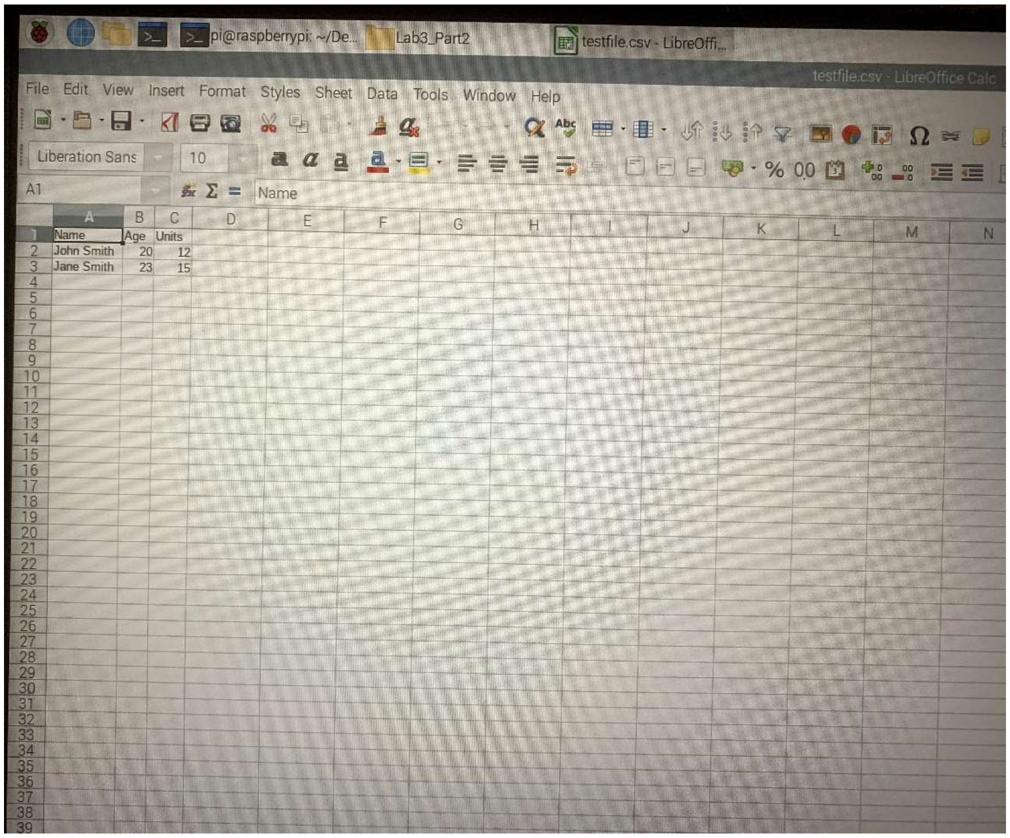


```
pi@raspberrypi:~/Desktop/Lab3_Part2 $ python3 Write_to_file.py
pi@raspberrypi:~/Desktop/Lab3_Part2 $ ls
testfile.txt  Write_to_file.py
pi@raspberrypi:~/Desktop/Lab3_Part2 $ cat testfile.txt
Hello Worldpi@raspberrypi:~/Desktop/Lab3_Part2 $
```

Figure 2.3: Output of ‘Hello World’ Python Code

```
#!/usr/bin/python

file = open("testfile.csv","w")

file.write("Name,Age,Units,\n")
file.write("John Smith,20,12,\n")
file.write("Jane Smith, 23,15,\n")

file.close()
```

Figure 2.4: Write to CSV file Python Code



	A	B	C
1	Name	Age	Units
2	John Smith	20	12
3	Jane Smith	23	15

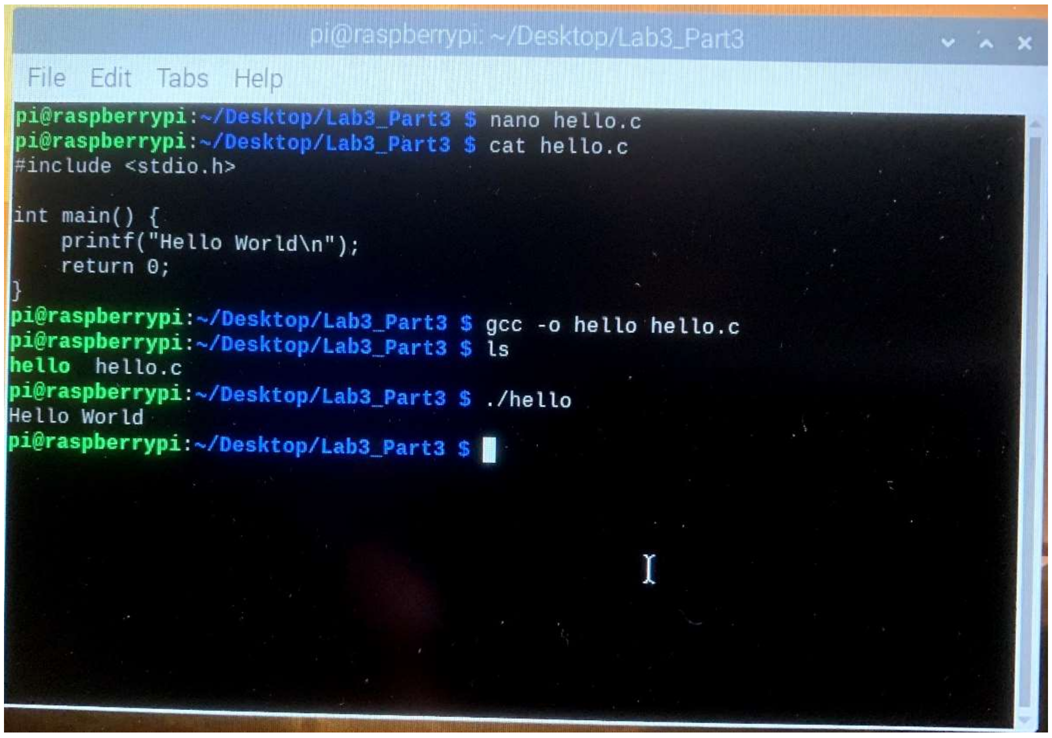
Figure 2.5: Result of Write to CSV file Python Program

Part 3: Writing a C/C++ Program for Raspberry Pi:

For this section of the lab, we will write a simple C program using the Raspberry Pi to compile and run our program. First, we created a new file using the nano text editor in the terminal and wrote the code shown in Figure 3.1. Then we compiled and ran the program to get the output shown in Figure 3.2.

```
#include <stdio.h>

int main(){
    printf("Hello World\n");
    return 0;
}
```

Figure 3.1: 'Hello World' C Program



```
pi@raspberrypi:~/Desktop/Lab3_Part3 $ nano hello.c
pi@raspberrypi:~/Desktop/Lab3_Part3 $ cat hello.c
#include <stdio.h>

int main() {
    printf("Hello World\n");
    return 0;
}
pi@raspberrypi:~/Desktop/Lab3_Part3 $ gcc -o hello hello.c
pi@raspberrypi:~/Desktop/Lab3_Part3 $ ls
hello  hello.c
pi@raspberrypi:~/Desktop/Lab3_Part3 $ ./hello
Hello World
pi@raspberrypi:~/Desktop/Lab3_Part3 $ 
```

Figure 3.2: 'Hello World' C Program Output

Part 4: Use GPIO for Raspberry Pi

For this part of the lab, we will use a simple LED and push button circuit. We wired the breadboard as shown in Figure 4.1. Then we wrote the provided code shown in Figure 4.2 into a text file and called it ‘gpiotest.c’. We compiled the code and ran the program. When the push button is open, the LED will not turn on and when the push button is closed, the LED will turn on.

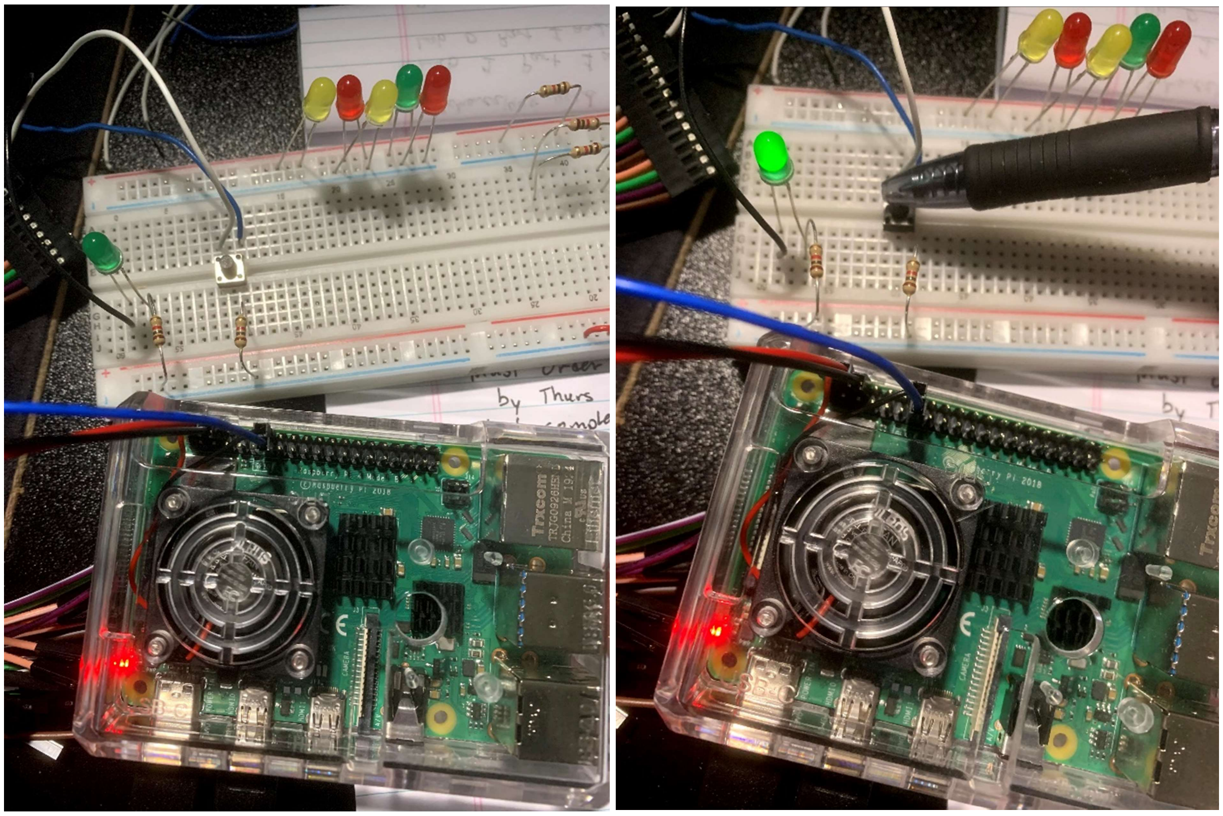


Figure 4.1: LED and Push Button Circuit GPIO

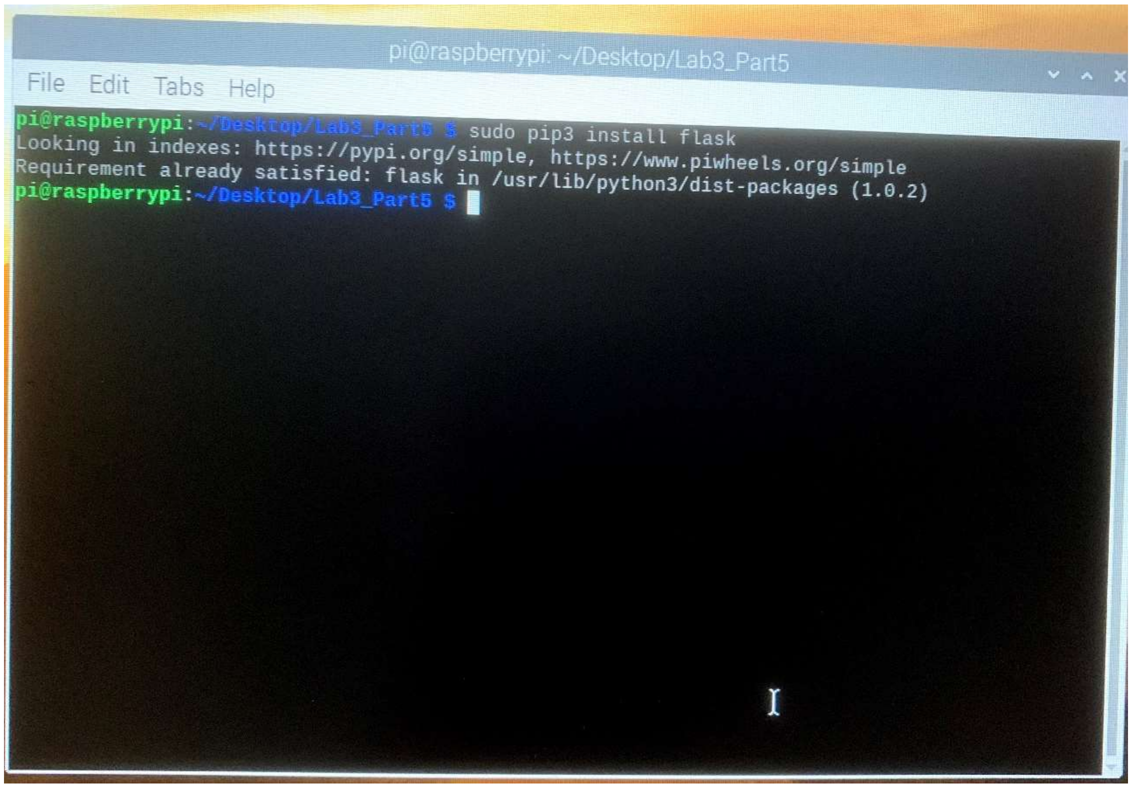
```
#include <stdio.h>   //Used for printf() statements
#include <wiringPi.h> // Include WiringPi library!

// Pin number declaraction. We're using the Broadcom chip pin numbers.
const int ledPin = 17; // Regular LED - Broadcom pin 23, P1 pin 16
const int buttonPin = 27; // Active-low button - Broadcom pin 17, P1 pin 11

int main (void)
{
   // Setup stuff:
   wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

   pinMode(ledPin, OUTPUT);   // Set regular LED as output
   pinMode(buttonPin, INPUT);  // Set button as INPUT

   printf("C GPIO program running! Press CTRL+C to quit.\n");

   while(1)
   {
       if(digitalRead(buttonPin))
         digitalWrite(ledPin, HIGH);
       else
         digitalWrite(ledPin, LOW);
   }

   return 0;
}
```

Figure 4.2: Code Provided

Part 5: Web Server

One of the most common applications of the Raspberry Pi is using network protocols to display data or control aspects of your project through a Web Server. In this portion of the lab, we will use python with Flask framework to create a simple web server that is hosted on your Raspberry Pi.

First, we check to see if Flask Framework was already installed by typing the command shown in Figure 5.1. Second, we wrote the Python Program shown in Figure 5.2 to host our Web Server. We then receive a screen shown in Figure 5.3 when we run the program. In order to observe the web page, we opened a web browser and typed in the following address into the address bar: http://0.0.0.0:5000/ (see Figure 5.4). To make the Web Server more personal, I modified the background and the text of the Web Server pages by using HTML, CSS, and changing some parts of the initial Python program (see Figures 5.5, 5.6, 5.7). See Figure 5.8, 5.9, and 5.10 for code used to create the Web Server.

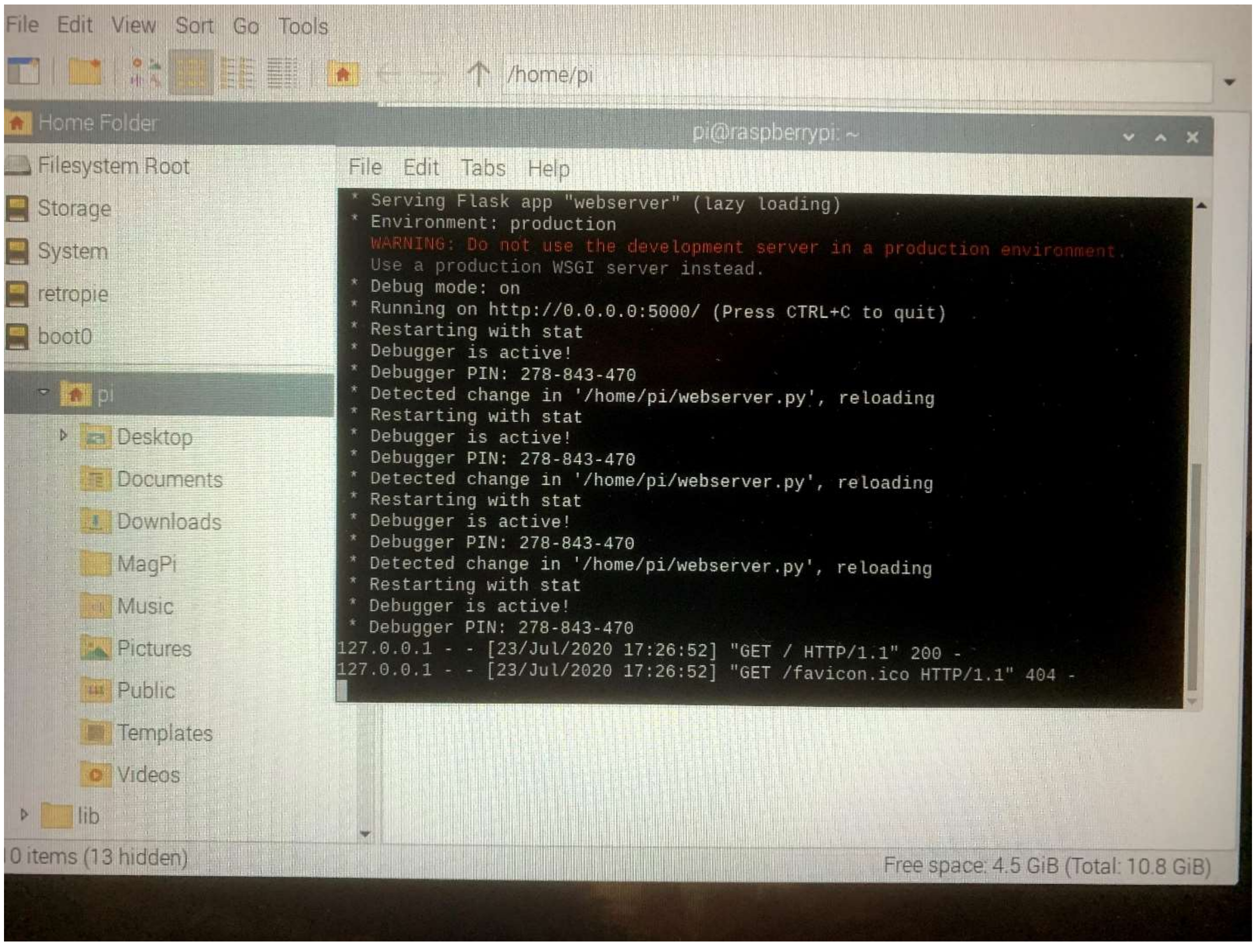


```
pi@raspberrypi:~/Desktop/Lab3_Part5 $ sudo pip3 install flask
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: flask in /usr/lib/python3/dist-packages (1.0.2)
pi@raspberrypi:~/Desktop/Lab3_Part5 $ 
```

Figure 5.1: Install Flask Framework

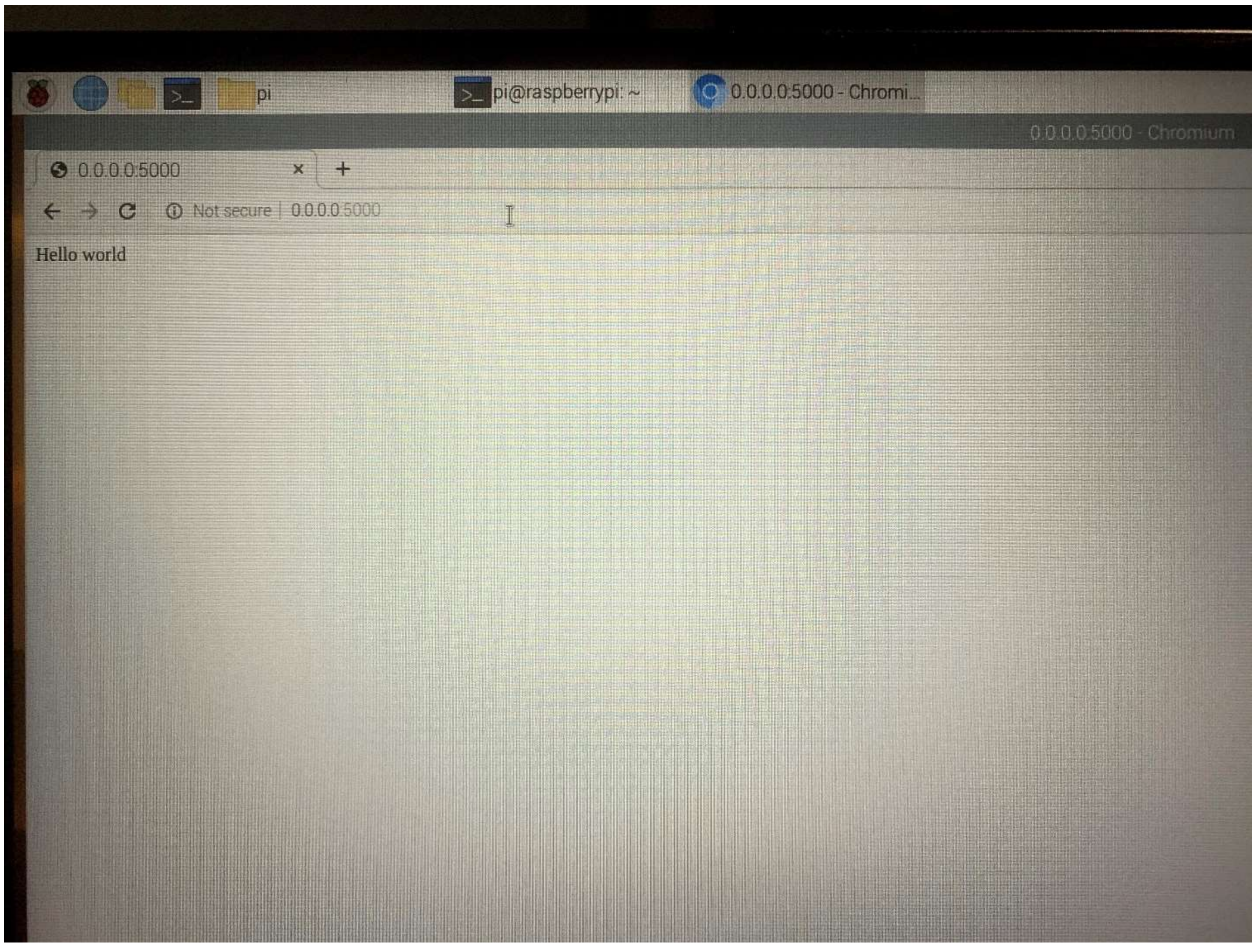
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
        return 'Hello world'

if __name__ == '__main__':
        app.run(debug=True, host='0.0.0.0', port='5000')
```

Figure 5.2: Provided Python Program



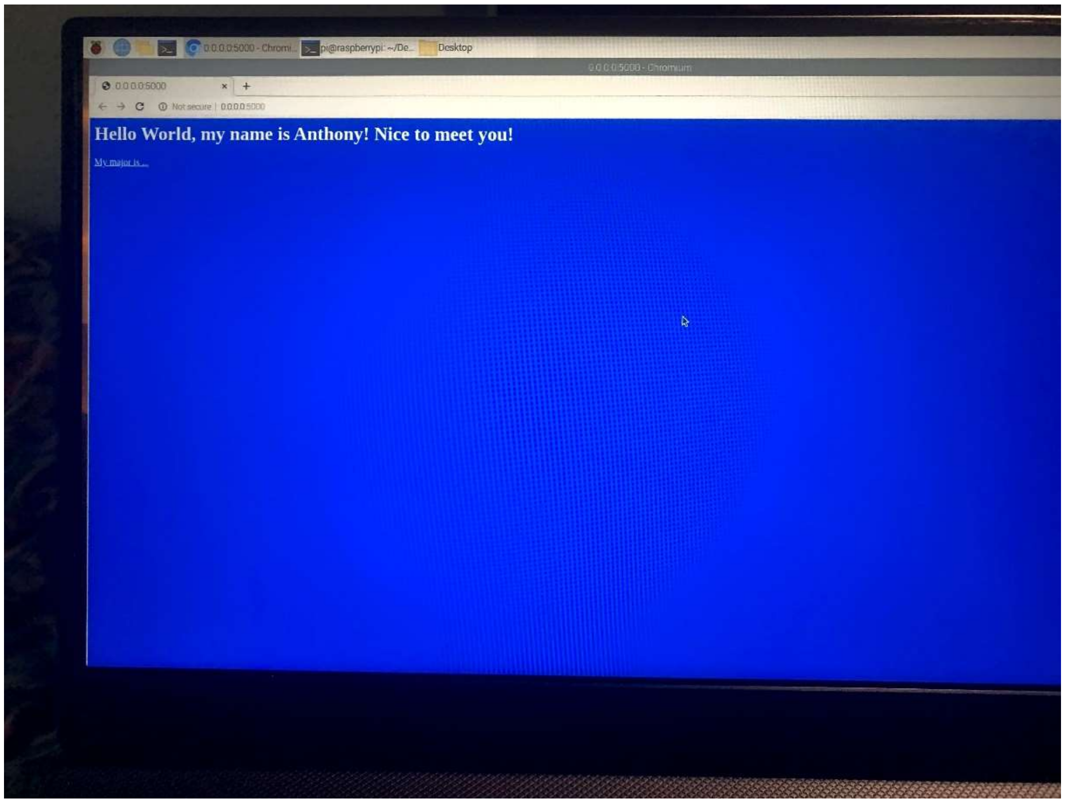
```
 * Serving Flask app "webserver" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 278-843-470
 * Detected change in '/home/pi/webserver.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 278-843-470
 * Detected change in '/home/pi/webserver.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 278-843-470
 * Detected change in '/home/pi/webserver.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 278-843-470
127.0.0.1 - - [23/Jul/2020 17:26:52] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Jul/2020 17:26:52] "GET /favicon.ico HTTP/1.1" 404 -
```

Figure 5.3: Running the Web Server Program



Hello world

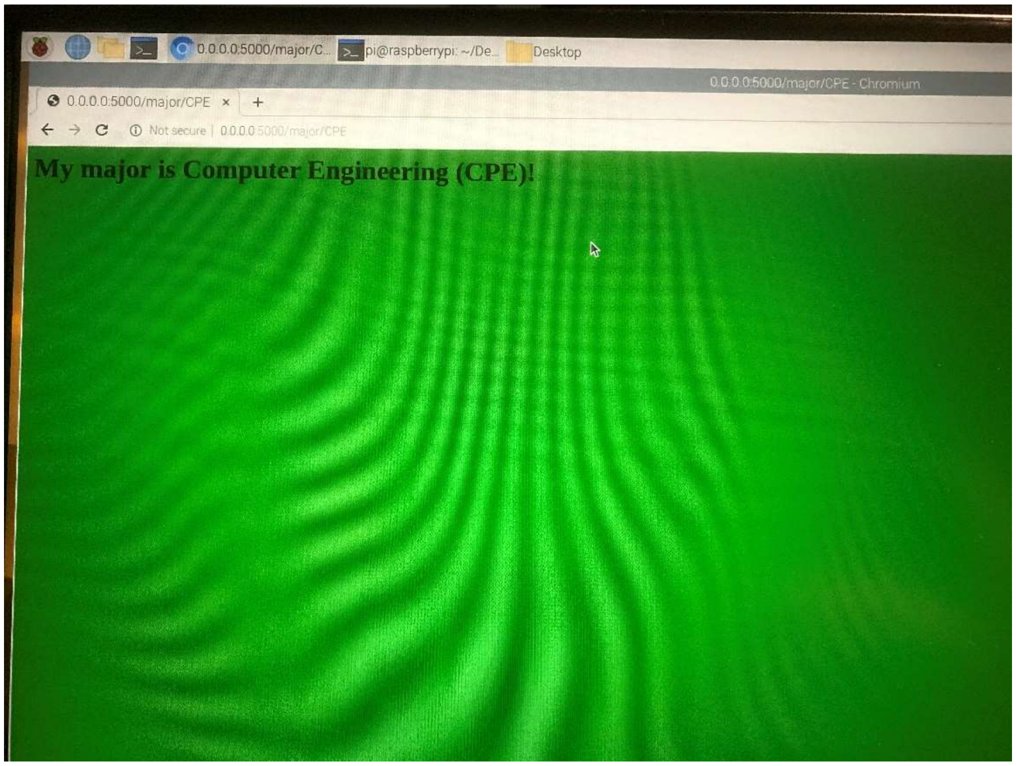
Figure 5.4: Initial Web Server in Browser



Hello World, my name is Anthony! Nice to meet you!

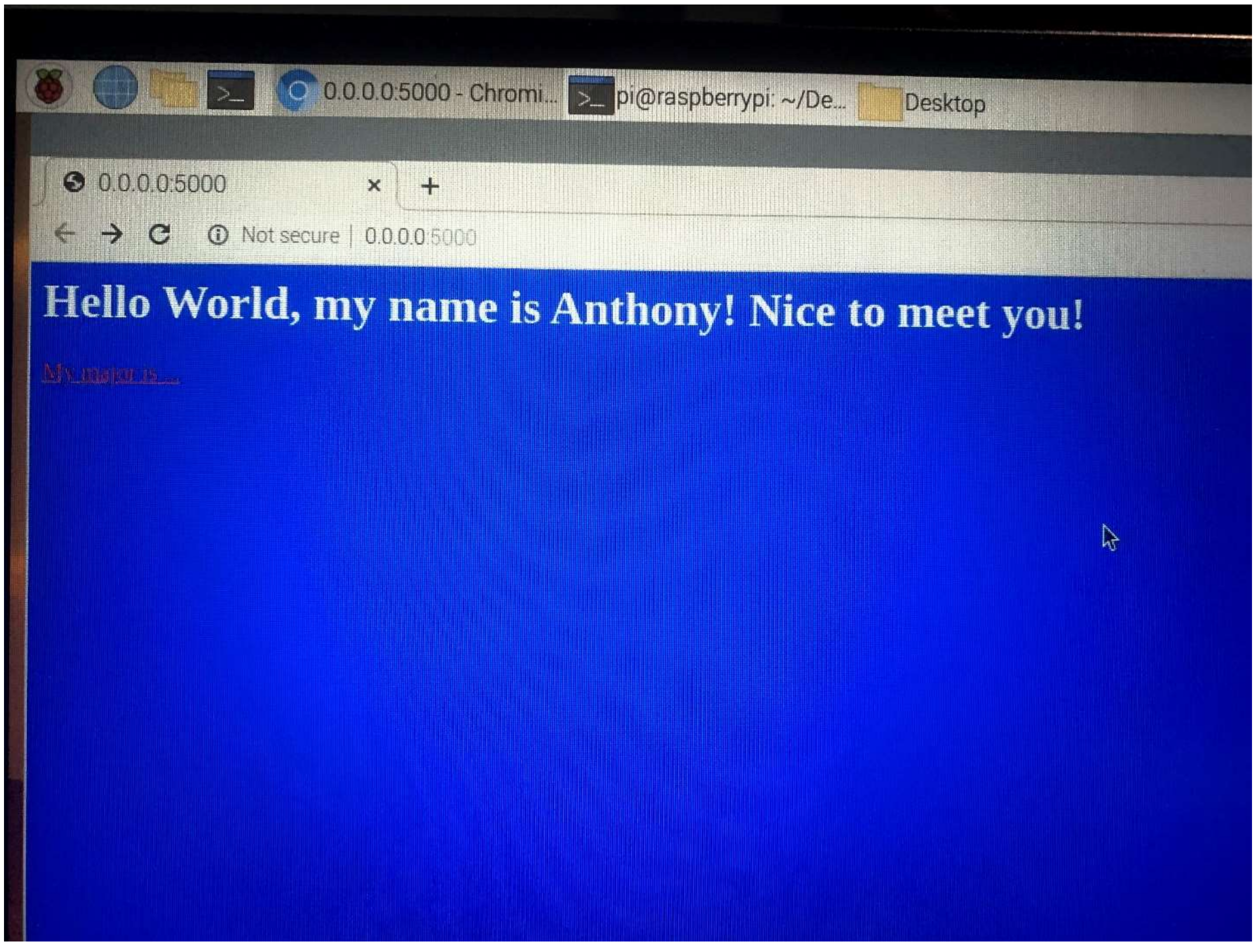
My major is...

Figure 5.5: Modified Web Server Index Page, Hyperlink not clicked, and color not changed



My major is Computer Engineering (CPE)!

Figure 5.6: Modified Web Server Second Page, Hyperlink clicked



Hello World, my name is Anthony! Nice to meet you!

My major is...

Figure 5.7: Modified Web Server Index Page, Hyperlink clicked, and color changed

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
   return render_template('index.html')

@app.route('/major/<name>')
def major(name):
          return render_template('page.html', name='Computer Engineering (CPE)')

if __name__ == '__main__':
   app.run(debug=True, host='0.0.0.0', port='5000')
```

Figure 5.8: websever.py Program modified

```
body {
   background: blue;
   color: white;
}

a:link {
          color: white;
}

a:visited {
          color: red;
}
```

```
body {
   background: green;
   color: black;
}
```

Figure 5.9: Top: style.css for Index Page, Bottom: style2.css for Second Page

```
<html>
<head>
<link rel="stylesheet" href='/static/style.css' />
</head>
<body>
<h1>Hello World, my name is Anthony! Nice to meet you!</h1>
<a href="/major/CPE">My major is ...</a>
</body>
</html>
```

```
<html>
<head>
<link rel="stylesheet" href='/static/style2.css' />
</head>
<body>
<h1>My major is {{ name }}!</h1>
</body>
</html>
```

Figure 5.10: Top: index.html for Index Page, Bottom: page.html for Second Page

Part 6: Project

For this portion of the lab we will make a simple project that interacts with the GPIO pins or uses some network programming. We decided to make a simple C program that will light up three LEDs in two unique patterns. First, we wired the breadboard as shown in Figure 6.1. Second, we modified the code used in Part 4 and came up with the code shown in Figure 6.2.

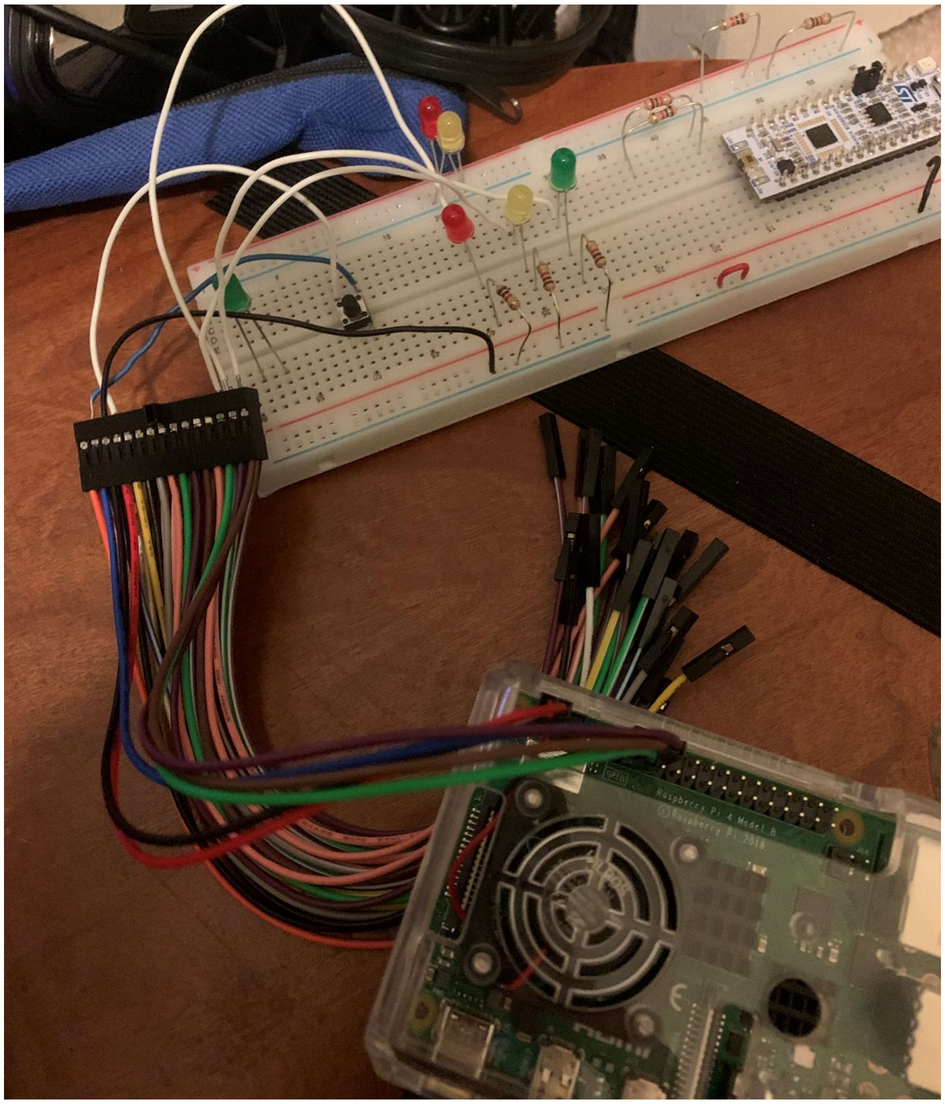


Figure 6.1: Breadboard Set Up

```
#include <stdio.h>   //Used for printf() statements
#include <wiringPi.h> // Include WiringPi library!

// Pin number declaraction. We're using the Broadcom chip pin numbers.
const int ledPin1 = 22;          // Regular LED - Broadcom pin 22, Pi pin 15
const int ledPin2 = 23;          // Regular LED - Broadcom pin 23, Pi pin 16
const int ledPin3 = 24;          // Regular LED - Broadcom pin 24, Pi pin 18

int main (void)
{
   // Setup stuff:
   wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

   pinMode(ledPin1, OUTPUT);   // Set regular LED as output, Green
   pinMode(ledPin2, OUTPUT);   // Set regular LED as output, Yellow
   pinMode(ledPin3, OUTPUT);   // Set regular LED as output, Red

   int answr = 0;

   printf("Welcome to LedLights!\n");

   printf("Please pick an option:\n");
   printf("Option 1\n");
   printf("Option 2\n");
   scanf("\n%d",&answr);

   if (answr == 1) {
           digitalWrite(ledPin1, HIGH);
           delay (500);
           digitalWrite(ledPin1, LOW);
           delay (500);
           digitalWrite(ledPin1, HIGH);
           delay (500);
           digitalWrite(ledPin1, LOW);
           delay (500);

           digitalWrite(ledPin2, HIGH);
           delay (500);
           digitalWrite(ledPin2, LOW);
           delay (500);
           digitalWrite(ledPin2, HIGH);
           delay (500);
           digitalWrite(ledPin2, LOW);
           delay (500);

           digitalWrite(ledPin3, HIGH);
           delay (500);
           digitalWrite(ledPin3, LOW);
           delay (500);
           digitalWrite(ledPin3, HIGH);
           delay (500);
digitalWrite(ledPin3, LOW);
            delay (500);
   }
   else if (answr == 2) {
               for(int i = 0; i < 20; ++i) {
                  if(i < 10 && i % 2 != 0) {
                            digitalWrite(ledPin2, HIGH);
                            delay (150);
                            digitalWrite(ledPin2, LOW);
                  }
                  else if(i % 2 == 0) {
                            digitalWrite(ledPin1, HIGH);
                            delay(150);
                            digitalWrite(ledPin1, LOW);
                  }
                  else {
                            digitalWrite(ledPin3, HIGH);
                            delay(150);
                            digitalWrite(ledPin3, LOW);
                  }
               }
   }


   digitalWrite(ledPin1, LOW);
   digitalWrite(ledPin2, LOW);
   digitalWrite(ledPin3, LOW);

   printf("Bye Bye.\n");

   return 0;
}
```

Conclusion:

At the end of this lab, I think I understand the basics of using the Raspberry Pi. I've heard of all the fun projects you can create using this little piece of hardware and look forward to incorporating it into our Final Project. Maybe in the future I'll try doing a networking project to become more familiar with the full capabilities of this device.