Louis Jankowski

Cps210

Due 4/3/2022

Final phase 2

My final result has been a local Flask server that uses python to operate the functionality while the display is ran on html with some CSS functionality. This program operates a LED by a web browser interface. This is all hosted on a razzberry pi.

The first step to this project is getting the supplies. The required Items are as followed:

* Computer
* [Raspberry pi](https://www.amazon.com/Raspberry-Model-2019-Quad-Bluetooth/dp/B07TC2BK1X/ref=dp_fod_1?pd_rd_i=B07TC2BK1X&psc=1) with power cable
* [Micro SD](https://www.amazon.com/SanDisk-256GB-MicroSDXC-Memory-Adapter/dp/B08GY8NHF2/ref=sr_1_3?crid=3TWBQ2BNOO6JO&keywords=microsd&qid=1649277397&s=electronics&sprefix=microsd+%2Celectronics%2C85&sr=1-3) with adapter
* A [breadboard](https://www.amazon.com/Breadborad-Solderless-Breadboards-Distribution-Connecting/dp/B082VYXDF1/ref=sr_1_1_sspa?crid=2CJ1EJ8TOBFFB&keywords=breadboard&qid=1649277435&s=electronics&sprefix=breadboard%2Celectronics%2C77&sr=1-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzUFM3WlRITTlKOFA5JmVuY3J5cHRlZElkPUEwODk1NzQxM0U0QlMwRkM2T01DMSZlbmNyeXB0ZWRBZElkPUEwOTM3NjM1MUxSWVZFMzU0QUdKWiZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=)
* [LED](https://www.amazon.com/eBoot-Pieces-Emitting-Diodes-Assorted/dp/B06XPV4CSH/ref=sr_1_14?keywords=LED&qid=1649277466&sr=8-14)
* [Mail to female cables](https://www.amazon.com/EDGELEC-Breadboard-Optional-Assorted-Multicolored/dp/B07GD2BWPY/ref=sr_1_9?crid=3HN1B1XWXOXTK&keywords=male+female+pin+wire&qid=1649277529&sprefix=mail+female+pin+wires%2Caps%2C89&sr=8-9)
* [A 1k ohm resister](https://www.amazon.com/Elegoo-Values-Resistor-Assortment-Compliant/dp/B072BL2VX1/ref=sr_1_4?crid=N3GC60396J0B&keywords=resisters&qid=1649277594&sprefix=resisters%2Caps%2C81&sr=8-4)
* Keyboard, mouse, and display

To start this project first the micro-SD needs to be loaded with the operating system Raspberry Pi OS. ([Video tutorial](https://youtu.be/y45hsd2AOpw)) To start, download the installer on a computer from [here](https://www.raspberrypi.com/software/). Then select the standard full 32-bit download of Raspberry PI OS image. Make sure the right SD is selected and install.

Once installed on the SD plug it into the storage slot on the Raspberry Pi and plug the Pi up to the keyboard, mouse, display, and power. Next setup the pi and connect it to the internet. Update the pi which will be prompted automatically.

Then open the console command and enter “pip install FLASK”. This will install [Flask](https://en.wikipedia.org/wiki/Flask_(web_framework)), This is a micro web framework written in python. This will be used to interact with the HTML files from the python file.

The programs were made in a folder called final. In that folder is another folder called templates. It is important that it is named templates because that is how flask locates the html templates.

Using Thonny Python IDE I created the following python programed called app.py this is also a name that flask looks for. This is the code:

**app.py**

from flask import Flask, render\_template

import RPi.GPIO as GPIO

app = Flask(\_\_name\_\_)

pin = 11

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(pin , GPIO.OUT)

GPIO.output(pin, GPIO.LOW)

r\_val = map

@app.route('/')

def home():

    return render\_template('on.html')

@app.route('/on')

def on():

    GPIO.setmode(GPIO.BOARD)

    GPIO.setup(pin , GPIO.OUT)

    GPIO.output(pin, GPIO.HIGH)

    return render\_template('on.html')

@app.route('/off')

def off():

    GPIO.setmode(GPIO.BOARD)

    GPIO.setup(pin , GPIO.OUT)

    GPIO.output(pin, GPIO.LOW)

    return render\_template('off.html')

#p\_R = GPIO.PWM(pin, 2000)

GPIO.cleanup()

Graphical user interface, text, application

Description automatically generatedThere are 3 routes that control the actions done by each page the first route sets the default page and sends it to the /on rough. The /on route turns on the light and sends the user to on.html

Graphical user interface, text, application

Description automatically generatedWhen the checkbox is clicked it signals app.py to return route /off which turns off the light and sends the user to off.html.

When the box is clicked again it returns /on again in a loop.

**On.html:**

<html>

<body>

<H1> On/Off </H1>

<form action="/on">

<lable class="switch">

<input type="checkbox" onclick="submit();">

<span class="slider round"></span>

</lable>

</form>

</body>

</html>

**Off.html:**

<html>

<body>

<H1> On/Off </H1>

<form action="/on">

<lable class="switch">

<input type="checkbox" onclick="submit();">

<span class="slider round"></span>

</lable>

</form>

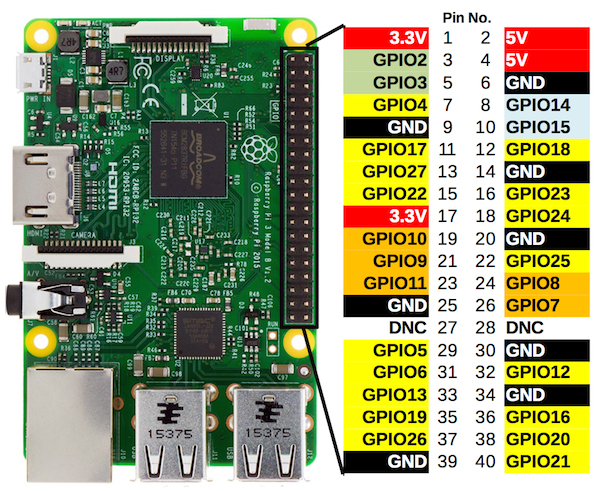
</body>

</html>

Both html files were written in the preinstalled text editor.

To run this program in the command prompt, change the directory to the project file then enter the commands “set FLASK\_APP=app.py” this gives flask the proper route. Then enter “flask run” this will start the program and return the link to open the sight. This is all the coding needed.

**Wiring:**

Note: never attempt to wire anything while the raspberry pi is on that is how I burnt out my board and I had to borrow a friend’s to finish this project.

First, I connected ground pin 9 to the breadboard with jumper wire. I then connected GPIO17 pin 11 using another jumper wire.

A picture containing text, indoor, floor, wooden

Description automatically generatedThen I plugged the ground end of the light to the ground line and the positive end on the main section of the breadboard and connected the GPIO signal to the light with a 1k ohm resister.

A picture containing electronics

Description automatically generated

Wrap Up:

This project was very difficult to implement, and I went through several ideas on how to implement my idea. As I developed it, I had to make several changes from phase one. Many sources have helped me implement the final design.

I was unable to fit the relay in the final product, but they are controlled in much the same way the only change that would be needed is a 5-volt power line and no resister. This would allow what I have already implemented to be more useful in providing light. When I get my new Raspberry pi I will probably add the relay.

**Resources:**

* <https://youtu.be/0meTbQQaosU>
* <https://youtu.be/vwcAngFFj9g>
* <https://youtu.be/Tsy8SwY_jic>
* <https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pis-gpio-pins>
* <https://flask.palletsprojects.com/en/2.1.x/quickstart/>
* <https://www.raspberrypi.org/help/>
* <https://www.deviceplus.com/raspberry-pi/an-introduction-to-raspberry-pi-gpio-pins/>