Controlling LED Lights Over the Web

By: Asmaa Elkeurti, Kiel Martyn, Ramya Puliadi

Intro

Using a simple HTML form, users are able to control the brightness levels of three LED lights attached to a circuit board powered by a Raspberry Pi computer. Any system which shares a network with the Pi is able to access this web page and select which light to target and the brightness value for that light.

Our project uses two programming languages: Python and HTML. We chose Flask, a micro web framework written in Python, for the networking component and the GPIO library, which allows us to interact with the LED lights.

On the hardware side, we used a Raspberry Pi which was hooked up to a breadboard on which the LED lights are connected.

We used Git as our VCS. Our project repository can be found here [1].

Development Process

The first step we took was setting up the Flask framework and using that to serve up our simple html page [2]. We then set up our breadboard circuit and connected that to our Raspberry Pi. The next step was to get comfortable with the Python GPIO library by experimenting with the lights directly from a script on the Raspberry Pi. After we had the lights working, we started implementing a user friendly interface on our website that would control the lights. We used WireShark to see what was actually encoded in the http packets for the requests [3]. The web interface was created using an HTML form that had two drop down menus, one to select which pin to control and another to choose the brightness level at which to set the previously selected pin. Then we looked at how use curl commands the control our application from the command line by sending client POST requests [4].

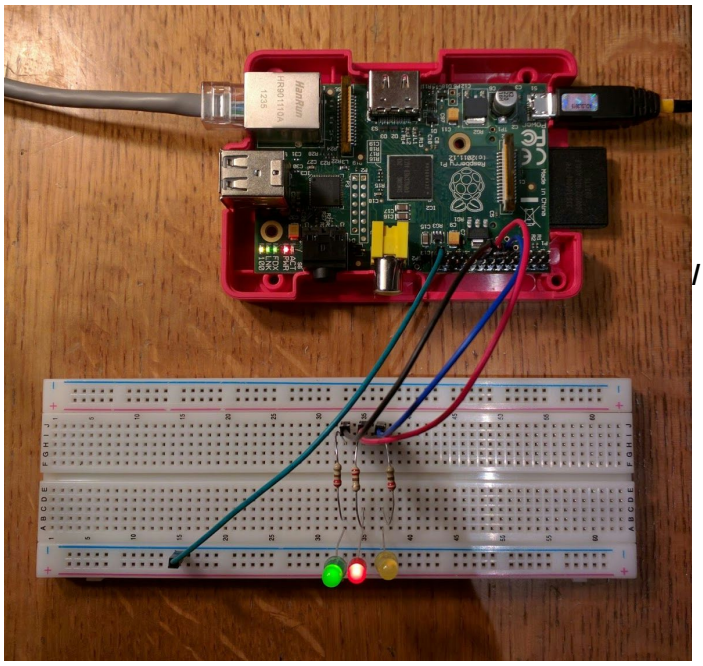

Figure 1: Final set up for Raspberry Pi and breadboard.

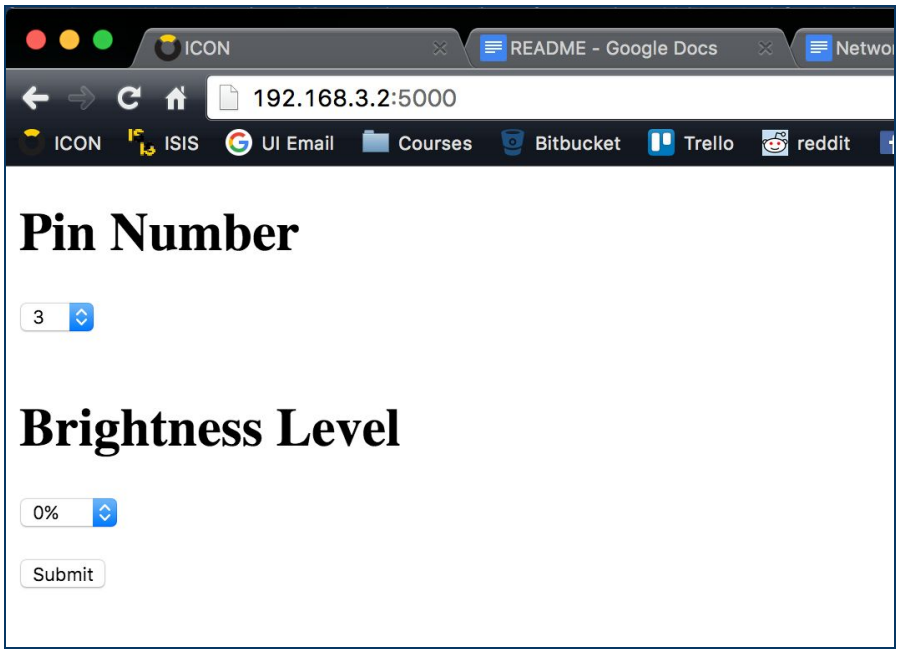

Figure 2: Homepage for web application to control LEDs.

Skills Developed

One of the first parts of the project that we had to familiarize ourselves with was understanding how to build the hardware for our project. None of us had had any experience with connecting Raspberry Pi with a breadboard. We learned how to set up our breadboard with the necessary LEDs and resistors. With the LEDs, we originally had a problem because we had not realized that one of the pins on the LED was shorter and that side had to be connected to ground.

For our web application, we chose to write our program in python because we thought that would be the most straightforward language to program our app in and that was the language that all of us were the most comfortable with. When looking at which micro-framework to use, we found that Flask was documented well and the tutorial was easy to follow. Additionally, it was the framework that Bryan recommended. The Flask framework was very simple to use; we had one route that could process both our POST requests and GET requests. Within that function we used if statements to differentiate between what code needed to be executed for GET versus POST requests. We also learnt how to use POST requests with HTML when we created the form to submit the pin numbers and brightness values to control our Flask application.

Problems Encountered

One of the largest issues we faced had to do with the GPIO library PWM implementation. Duty cycles can only be implemented using a timer, and because this is done over software, once settings had been changed via PWM, the script would not process any other request until the original timer had finished counting down. We had three possible ways of overcoming this problem: we could have redirected the user to a page that didn't accept

requests until the timer had finished counting down, we could have used multithreading within our python script, or we could have tried changing the brightness values via hardware to bypass having to use a software PWM. We chose the second option and implemented another program thread which stops a running dc timer if another request is received.

We also ran across minor issues related to coding in html and the way our circuitry was wired, but each was pretty easily resolved using simple trial and error processes and using StackOverflow [5].

Ideas for Project Extension

There are many ways that we could take this project further than what we've so far created. One way to do this would be to revamp our website's interface to make it more interactive and more aesthetically appealing.

Another possible improvement would be to do more with the lights on our board. One idea we had was to add more LED lights to our board and program light shows. These programs would appear as presets to the user on our web interface.

A third creative idea we had was to somehow implement audio into our brightness. A tone would sound when each light that came on. The tone's loudness and pitch would be determined by the light's brightness. This would allow a user to program music!

GitHub Repository

[1] https://github.com/asmaaelk/RPI

References

[2] Flask.pocoo.org/docs

[3] wiki.wireshark.org/

[4] curl.haxx.se/docs

[5] Stackoverflow.com