Jeremy Goldberg

5/7/2021

Final Project Reflection

For this final project I created a WebSocket server on an ESP8266 WiFi development board that can control a 300 LED strip of NeoPixels. Currently, the user can choose from one of eight different color options, and toggle the power of the strip. When once client connects to the server and makes a change to the strip, that change is then propagated to the rest of the clients connected so every device has current information. In the future, functionality will be extended to allow the user to fully customize the LED strip.

Before beginning this project, I had to setup the hardware. I used an LED NeoPixel strip with 300 pixels, an ESP8266 board, and a 20-amp power regulator to give the system power. I connected the control wire of the strip to GPIO pin 4 on the board. In a fully optimized build, there should be resistors between the direct power source and the LED strip to avoid providing too much voltage to the strip.

My first problem I had to solve was determining how to actually write and send code to the board. First, I tried to use Arduino IDE. This allowed me to load example codes and send them to the board when it was connected to the computer, but it is a very lightweight IDE so more complicated projects, such as adding a file structure, were difficult. Then I started looking at Visual Studio Code. I have used that IDE extensively and I found an extension called PlatformIO that could accomplish the tasks of the Arduino IDE and more within VS code.

My next step was determining how to start and run a server on the device. I found some starter code on the internet that could change the power state of an onboard LED. I used that

code as a base and began to dissect it to learning about each component. In this code, the HTML code was stored as a string literal that was read in by the WebSocket code. When a client connected to the server, the server would output the client ID and the IP address of the client. When a client changes the power state of the LED, the server would read the request, change the necessary variable, and then notify all the clients of the change. This would allow for all clients to have up to date information.

I knew my project wanted to do something similar. However, there were some key differences I knew I wanted to make. First, and most importantly, I wanted to control an external LED strip, not on onboard LED. Secondly, I wanted to user to be able to change a variety of factors of the strip including power, color, and brightness. Finally, I wanted to structure my files in a way that would separate the HTML and CSS files from the main driver code.

Making the server control an external led strip was relatively easy. I had to import a library that had functions for interacting with a NeoPixel strip and ensure the control pin was connected properly. From there, I wrote some code that would set all the pixels of the LED strip to a specific color and then display the color. I also wrote code to set the brightness of the strip to 0 or 25 when the user toggled the power of the strip to simulate the strip turning off and on.

My next step was to figure out how to send signals from the clients to the server requesting changes to the state of the LED strip, and have the server propagate those changes to the clients. Previously, the code was structured in a way that only allowed on variable to be passed around, but I knew I needed to keep track of both the brightness and color of the led strip. I modified the code to send different signals depending on the signal received by the clients. Similarly, the clients would receive new signals from the server that they could interpret as

changes to the state of the LED strip. If for some reason the server or client sent some uninterpretable code, the server would ask the client to refresh.

This project taught me a lot about how data gets sent between clients and the server. One part is figuring out how the server should interpret data and one part is how the client should interpret data. Even though that data is representative of the same change in this case, the way each side figures out how to handle it is very different. I can see how with more signals being sent and more actions being represented, this problem could just become worse without proper care to the network.