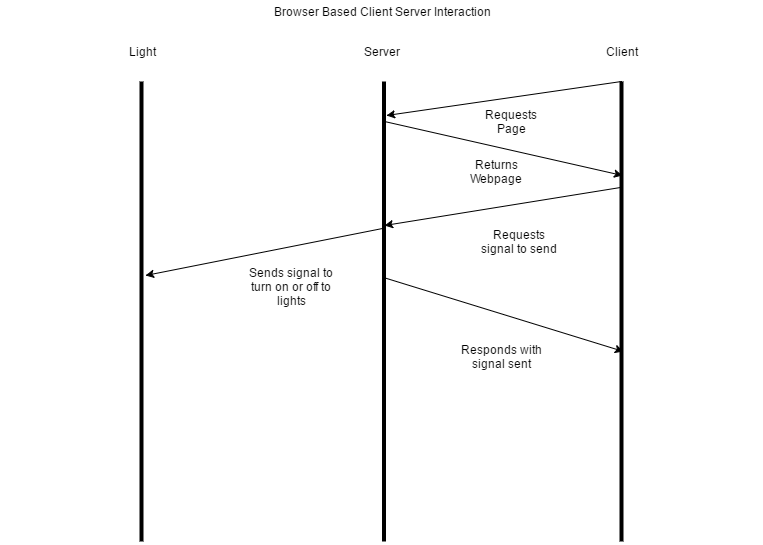
Goal

For this project I wanted to be able to control whether my lights were on or off from anywhere. To achieve this goal I originally wanted to use an Arduino since I already had one, hosting a small webserver and to control lights through a relay directly connected to the lights and the Arduino. After doing some research I decided against using this method for a few reasons. The first reason is that getting an Arduino connected to a network turned out to be either a pretty expensive if I bought an Arduino network shield, or it would be pretty difficult if I just bought a random Wi-Fi chip and tried to wire it up. The second reason is that finding a relay that did what I wanted and actually did it safely also seemed difficult as most of the ones that fit my criteria were incredibly expensive. Lastly I abandoned this idea because the relays would need to be connected directly to the lights which would require the lights to be in close proximity to the Arduino. I settled on using a Raspberry Pi using an RF transmitter to control RF outlets after seeing a few people be successful doing it online.

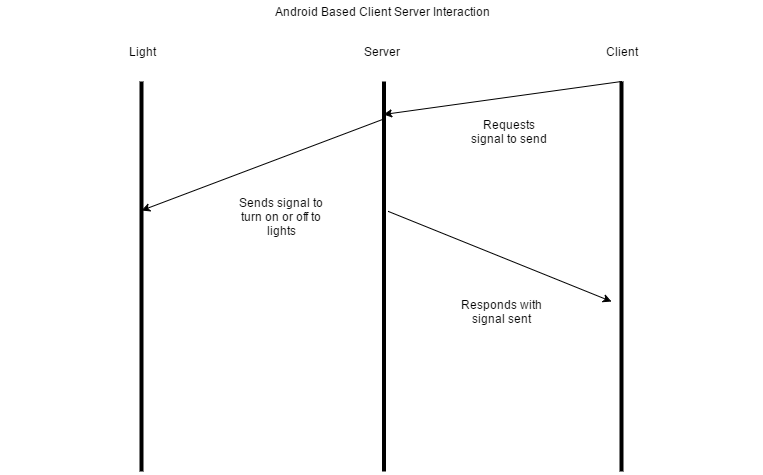
Design

There are 2 ways to interact with the Simple Security project. You can either connect using any browser on a device connected to the same network as the Raspberry Pi, or you can connect to it using an Android device running the Simple Security App.

The communication using a browser works as follows:

The browser first requests the webpage which the server then sends to the client. Next the client chooses what light to turn on or off by clicking a button. The browser then interprets what signal the user wants and sends a server a request with the signal in the payload. The server receives that request and interprets how to send the signal. When an outlet receives a signal it figures out if the command is for it by looking at what the signal is. If the signal is for the outlet it either turns on, off, or stays in the same state if it is already in that state. The outlet does not send out a signal back to the server so the server assumes that the signal was received and tells the client that the signal was sent.

Communication using an Android is very similar:



The only difference is that the Android device does not need to request the webpage from the server initially, as it already has all of the UI in the app. The rest of the communication is the same as the browser based client.

Files

There are 3 main files for the project, app.py, codesend, and app.js.

app.py - This is the file that hosts the webserver and handles all of the requests. It uses Flask as a framework to host the webserver and it handles 3 requests, the main request for the webpage, a request to send a signal to the lights, and a delayed request to send a signal to the lights after a period of time. When app.py receives a signal to send to the lights it calls the codesend program with the signal

codesend - This file handles all the communication to the lights using the RF transmitter. It uses arguments to receive what signal to send, what pulse length the signal should be and what pin the signal should be sent from. Once you know what the pulse length is and what pin the signal should be sent from, you don't vary from light to light unless you get a different batch of outlets, so the only argument that varies in the request usually is the signal.

app.js and controllers.js - These files control the frontend of the project. There is only 1 controller as there is only 1 page and the controller contains all of the signals that the user could wish to send and it has 2 functions for whether the user wants the signal delayed or not.

For the Android portion of the project there are 2 files that do a majority of the work while the rest are just configurations. These files are HTTPRequest.java and MainActivity.java.

MainActivity.java – This file sets up the basic responses for button presses and more importantly calls the HTTPRequest service when one of the light buttons is pressed and passes it what signal should be sent.

HTTPRequest.java – This file handles sending the request to the server. When the function is called it creates an intent service in the background that sends an HTTP request to the server with the signal of what light should turn on or off

File Tree

.  
├── Android  
│   └── src  
│      ├── main  
│      ├── AndroidManifest.xml  
│         ├── java  
│            └── com  
│            └── stevenscheffelaar  
│            └── simplesecurity  
│            ├── HTTPRequest.java  
│            └── MainActivity.java  
├── app.py  
├── codesend  
├── README.md  
├── static  
│   ├── css  
│   │   └── default.css  
│   ├── js  
│   │   ├── app.js  
│   │   └── controllers.js  
│   └── partials  
│   └── main.html  
└── templates  
 └── index.html