Chris Roma
Pi Tank Project

Web Browser Controlled Aquarium

The goal of this project was to be able to control the lights, filters, feeder, air pump, heater, etc. on an aquarium via a web browser. This goal has been accomplished, with some extra features. This paper will describe the steps needed to accomplish this goal. A secondary goal of the project was to try to keep everything "in house" and as cheap as possible.

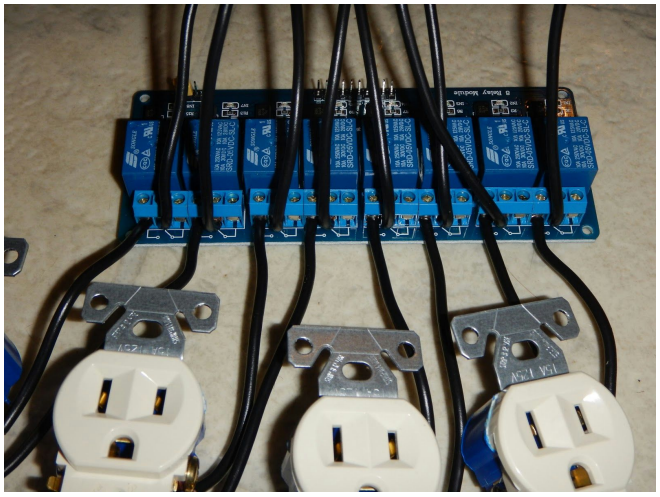
With the second goal in mind, the parts used for this project include: Internet service provider (Comcast), wireless router, Raspberry Pi model B+, 4GB SD card, WiFi dongle, 2 micro USB charging cables, spare Galaxy S4 cell phone (used as an IP camera), 8-channel 5V MAINS electric relay board, electrical box for 4 power outlets, 4 power outlets, extension cord, old PC AC cord, wire, tape, screws, hot glue.

The first step was to wire the 4 outlets so they will work when they receive power from a 120 volt source. I followed this guide to help with this because working with MAINS electric can be very dangerous or fatal.

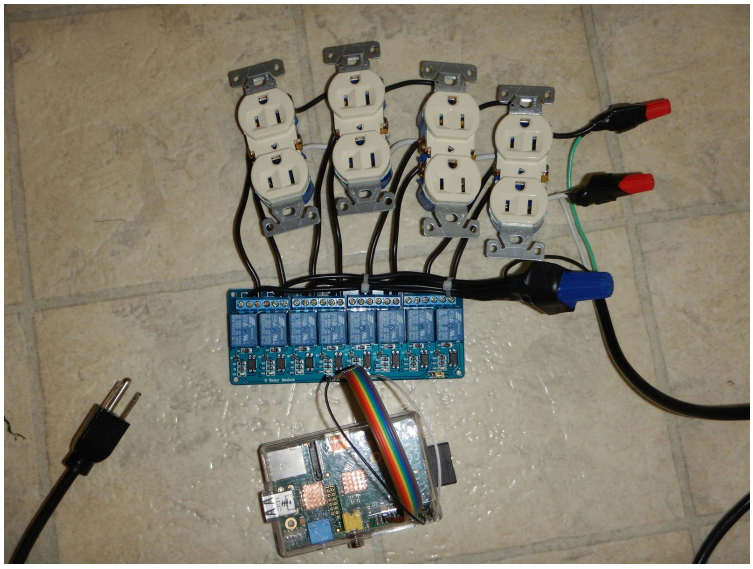
http://www.instructables.com/id/Web-Controlled-8-Channel-Powerstrip/

After following the guide, the result was 8 power outlets that all receive power individually.

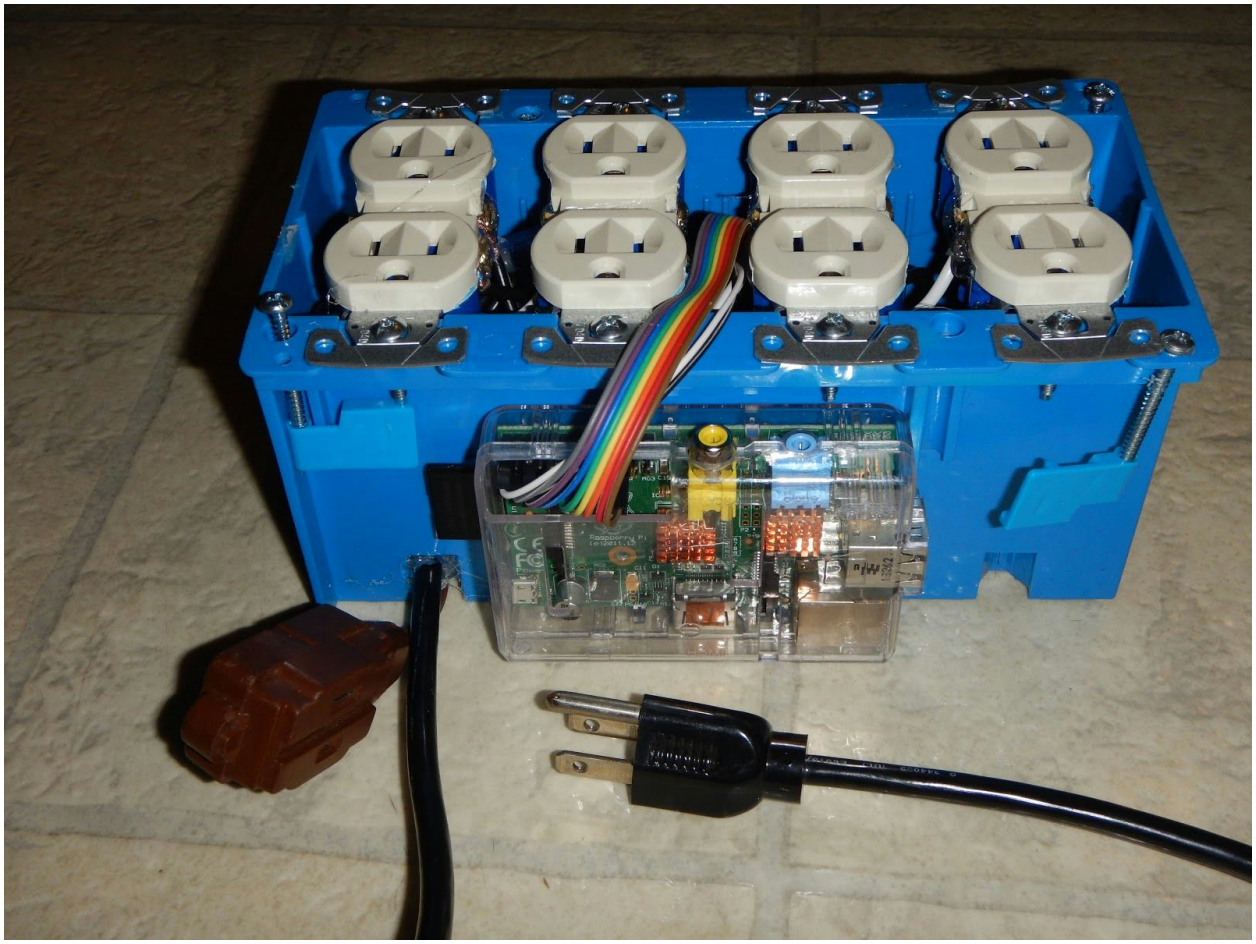
The next step was to wire the 8 power outlets to the 8 channel relay board. The 8 channel relay board is very important.



The relay board is the component that takes a signal from the Raspberry Pi's GPIO pins and uses that signal to decide if an outlet receives power or not. The 8 channel relay board has input wires from the GPIO on the Pi, and output wires that connect to the outlets. The old PC AC cord was then wired in to allow the outlets to be powered by the wall.



Next, the extension cord is cut and the female end is wired into circuit so that the Raspberry Pi can be powered by the project box and not need its own power supply. After that was complete all the parts were ready to be put into the project box for final assembly.

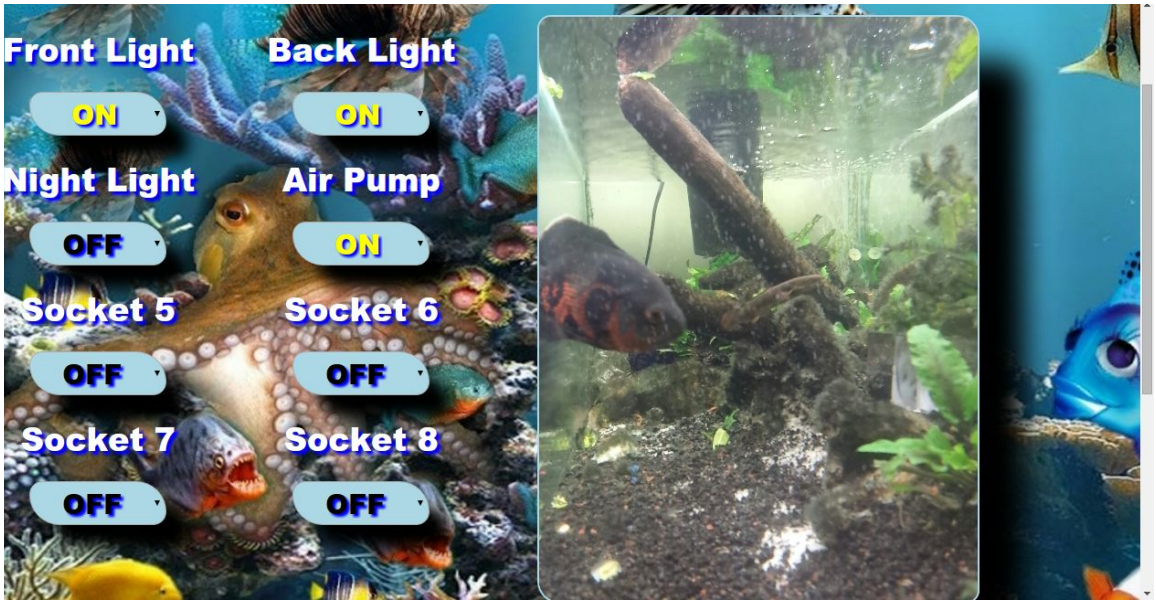


At this point we are done with the hardware and need to move on to the software end (This is when I stopped following the guide in the URL above).

The next step was to set up a web server on the raspberry pi. The Raspberry Pi is set to have a static IP address and uses Apache to serve the webpage. At first, the website was run locally and consisted of only simple on/off buttons for each power outlet. This was for testing purposes and proof of concept. The buttons on the website use PHP to call Python files that control the GPIO pins.

To test that the system was working, a light was plugged into each of the 8 outlets. From a computer connected to the same router as the Pi, a web browser was opened and each outlet tested for on/off functionality. When all the tests were successful, all that was left was to add functionality to the site such as security, efficiency, and live video.

The next phase was to add something to the page to prove what was clicked on in the web page actually triggered the action in the aquarium. The page has 2 solutions for this problem. First, there is a live video feed of the aquarium, so you can see what happens when you click a button. Second, the color of the on/off button on the web site changes based on weather the outlet is on (yellow) or off (black).



The color changes based on status of the outlet was a little tricky to get working. When the page loads, it uses PHP to call Python scripts that check the status of each outlet and returns the results to the page. The page then uses the results to decide what color the button should be. After page load, whenever a button is clicked, the page uses AJAX to re-check the status of the outlets and uses the results to update the colors without needing to reload the page.

The video is served from an old cell phone camera. The camera is attached to the side of the aquarium. The phone is set to have a static IP address and runs third party software called “IP camera” to serve video to a web browser via it’s IP address.

Once all of these components were successfully working together, it was time to make it available via the internet instead of just locally. This wasn't a big problem because my Comcast external IP address never changes. The only thing needed was to setup port forwarding in the router. With the router setup to forward the web server and video, the site was accessible from any internet connection via my IP address. This works, but is ugly. The next step was getting a domain name. http://RomasFish.xyz was acquired for a price of $1 per year, and setup to point to my IP address.

Once, the aquarium was able to be controled via http://RomasFish.xyz the project was almost complete.

Since the aquarium could be accessed from anywhere, it needed some security. This is when a password authentication page was added to site. The site allows you to view the video of the tank without a password, however none of the settings can be changed.

When the aquarum was able to be controlled via http://RomasFish.xyz securly using a password, and the results of the button clicks could be observed via the site, I considered the project complete.