Google-App-Script-Unblocker

Code is split between server side and Google App Script code. Server side is ideally run on a VPS, can be run on Repli.it but be warned it is against their TOS and is NOT RECOMMENDED NOR ENDORSED.

As a fork of alloy proxy if functions in much the same way: it intercepts every request sent from a user, and proxies them to the website that is being “unblocked” while appearing to a naive web filter as being allowed traffic and appearing to the server on the other end as a normal user. However it is unique in that it proxies requests twice, first to a google app script server, then to a custom server I am running, before reaching the end destination.

This allows all of a user's web requests to be sent to a google.com domain which looks normal for a user on a google doc in which this is embedded instead of some shady random server. In order to proxy requests through google app scripts fetch and xml open had to be rewritten from scratch following the spec. I highlight my rewrite of window.fetch in the file Google App Script Code/inject.js in lines 267-288

Roomba-Discord-Bot

This is a discord bot I made for the Project Grindstone Discord server. Its main feature is a quaint snake game I made to be controllable by discord messages. It's mainly included in this portfolio for the sentimental value it contained to me and the effort that was behind it.

I choose to highlight the lines 33-38 in index.js. These lines create a simple file server that serves a blank html document. I needed this bot to stay up 24/7 for the server and to do that I needed a quick way to test if it was online, and to send an alert if it wasn't. I had spent god knows how long attempting to use a discord api to check the online status of the bot, but they all ran into the same issue. Any code that was interacting with the Discord server would have too be on a bot, if said bot was down the code wouldn't be running, meaning I could only check if the bot was offline while the bot was online. That was not an effective strategy.

My solution was to add a few lines to create a webpage that could be queried to see if the bot was online. If the webpage could not be found and returned a 404 error then the bot was down and I'd be messaged by a separate service.

A-Trot-Through-Time

A JS web game I made with friends, all code is mine however. The game runs on a standard node.js file server and is rendered on the html canvas. The entire game engine and logic was written by me. This game was mostly a challenge in system building and design. No individual part of it was difficult to make, however the combination of all the parts, and getting the different systems to run at once in real time was a challenge.

Analytical-Balance

I choose to highlight lines 29-82 in the ScaleCode/ScaleCode.ino file in this repository.

The code itself is incredibly rough and not my best work, but the idea behind using a binary search in a physical machine context I feel was one of my single brightest moments. For context I was attempting to make an analytical balance, which canceled out the weight of a mass by running voltage through a magnet. From the examples I could find online they mostly appeared to use either analog voltage feedbacks or PID loops to adjust the voltage to find the exact cutoff. I lacked the circuit parts to use analog feedback and due to the limitations of my Arduino and specific peculiarities of the magnets force output the PID switched between being either underdamped or settling on the wrong value (due to a lingering magnetism on the metal not overdamping, meaning a higher integral term couldn't fix it).

The code segment therefore represents my solution to my dilemma. By using a binary search and checking discrete voltage values I can avoid aforementioned headaches and get an exact output voltage precise to 2^16 bits.

Itsy-Bitsy-Robot

Arduino path following code for a robot I made. You can specify paths in the path.cpp file. I was working with a 20-dollar kit with encoders so terrible that any odometry relying on them would be doomed to inaccuracy. It took weeks of work, combining data from accelerometers, ultrasonic distance sensors, and a Kalman filter (running on the built mpu6050s built in DMP - Digital Motion Processor) to yield anything resembling accurate tracking.