rootBeer Me

Smart Dormroom Refrigerated Can Dispenser

CS121A Final Project
10/23/20
The Dream Team
Jared DiScipio, Yuliang Peng, Henry Staunton, Jason Stillerman

Table of Contents

Introduction

rootBeer Me is a dorm room beverage dispensing system that will be internet enabled and allows for voice control.

There will be cans of soda stored in a mini fridge, and upon receiving a web request, it will open the fridge and roll a can down a chute. It will also be able to keep track of how many cans it has in inventory, and have a web Ui where you can view inventory and also dispense.

Voice control will be done through google home. API will be written in Node, and the UI will be written in React.

Definitions

MVP - Minimum Viable Product - This is the project we will deliver in its simplest form. We will not necessarily have enough time to add features outside the MVP, so we will lay out the baseline as what we will definitely accomplish.

Node - Node.js - This is server-side javascript code. It has the same syntax as standard Javascript but it runs on the server instead of the browser.

Express - Express.js - This is a framework written in Node that allows you to write web servers simply and succinctly.

Typescript - This is a Javascript superset that allows for optional typing of variables. It will require slight overhead in the form of pre-processing, but will speed up development significantly.

React - This is a front-end framework that we will use to rapidly prototype/develop the UI.

JSX - This is a javascript “syntax extension” which adds extra features to make React development more pleasant. Will require pre-processing via parcel.

Parcel - Parcel is a Javascript preprocessor that will allow us to convert Typescript into node code AND allows us to convert JSX into native browser-ready javascript.

ES5 - This is the standard syntax of javascript that is supported by all browsers as well as Node. Javascript is not actually a real language, every browser and node version interprets JS slightly differently, but ES5 is the standard that all parties must support.

NPM - Node Package Manager - This is what we will use to pull in external libraries such as Parcel, React, and Express. We will also use it to run scripts such as kicking off the preprocessing.

AJAX - Asynchronous Javascript and XML - Ajax is a pattern used by modern web developers to make asynchronous requests to the server from a browser, all without reloading the page.

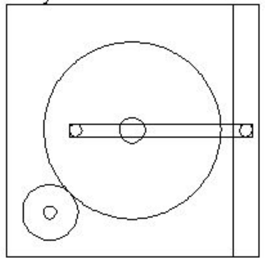
Axios - A Javascript library for making AJAX requests.

GPIO - General Purpose In/Out - These are the pins on the Pi that we will use to communicate with the hardware.

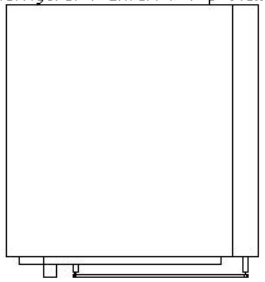
AWS - Amazon Web Services - A collection of cloud computing tools, the most interesting being their server rental. We can host a server powerful enough for our needs, 24/7 for free.

Blueprints

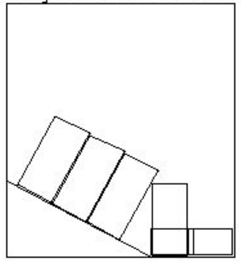
Refrigerator Exterior Side View



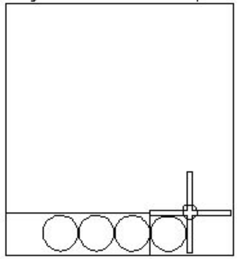
Refrigerator Exterior Top View



Refrigerator Interior Side View

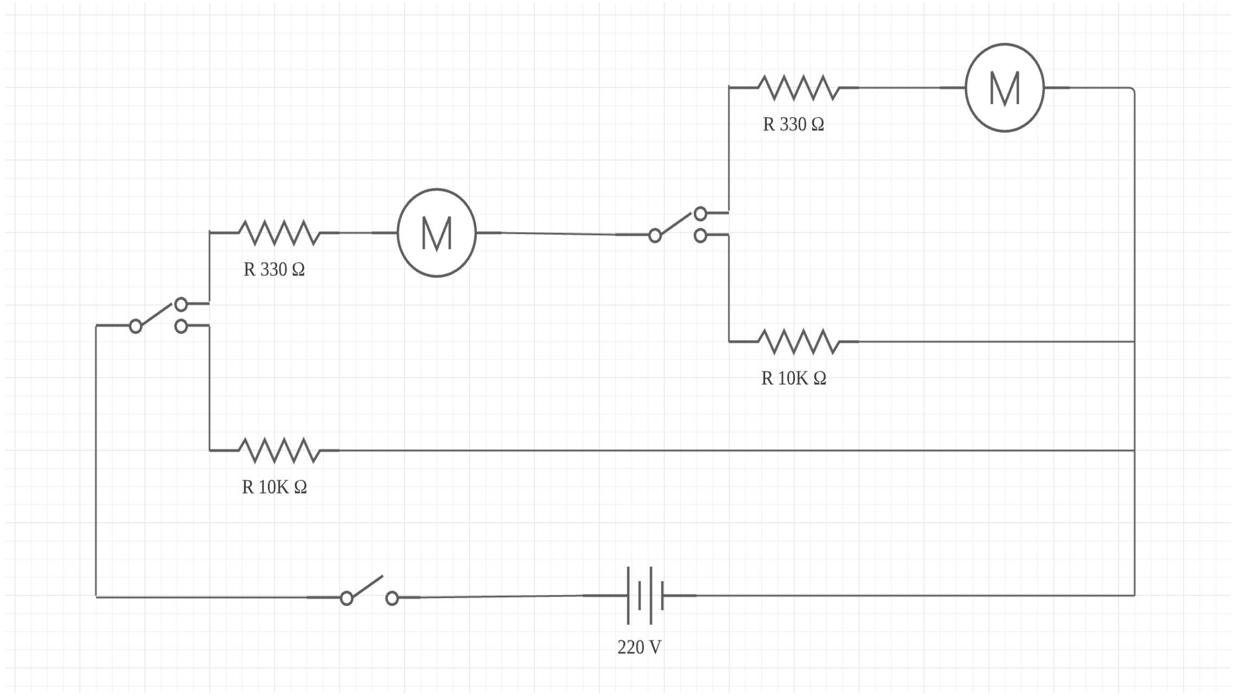


Refrigerator Interior Top View



Circuit Diagram

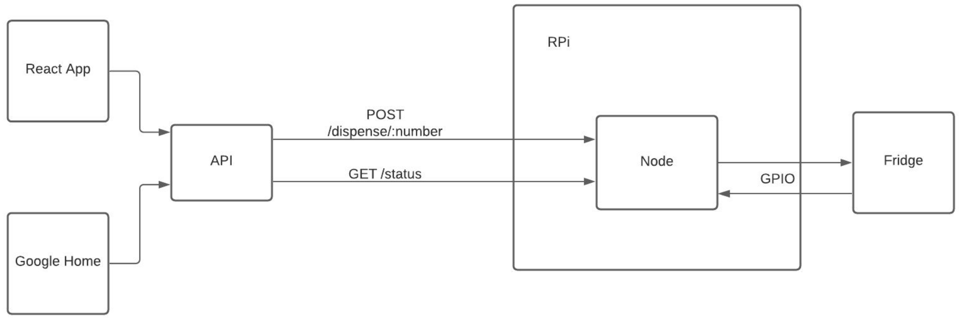
The M is a motor that the left side of the motor is for opening the door. The right side of the motor is whirl for can(tin) moving. We need to use resistance 330 Ω for the line which has a motor and 10K Ω for other lines (prevent the short circuit).



Software Phase 1

The software of this system will be rolled out in two stages. Stage one is the minimum viable product, and is diagrammed below by the figure titled "LAN Setup". Only after completing the whole project will we consider trying to develop stage 2, which is significantly more complicated.

LAN Setup



Details

Phase 1 Overview

There will be a Node server running Express on the Pi. This will field incoming API calls, and handle their responses as well. On the other end of the API, we have two clients - the google home and the react app.

Google home will be configured to make web requests to the API after hearing certain commands. “I’m thirsty” will make a POST request to the node server on the route “/dispense/1”. “What’s the stock looking like?” will make a GET request to “/status” and then parse the results to craft a response something like “You have 3 sodas left.”

The react app will be a very simple UI that shows how much stock you have left, as well as have buttons to dispense one or more sodas. This will be making GET and POST requests asynchronously to interact with the server.

The codebase for the Node server will be written in Typescript, and get transpiled into ES5 via a build step. The build step will be triggered manually via Parcel and NPM. It will field web requests with Express and will interact with the hardware via the Pi’s GPIO pins.

For the React client, the codebase will be written in the JSX flavor of typescript and will also be preprocessed with Parcel. Due to the nature of the app, we will not need any of React’s complicated state management libraries. The whole UI could probably be implemented in a single component. We will use Axios for ajax requests.

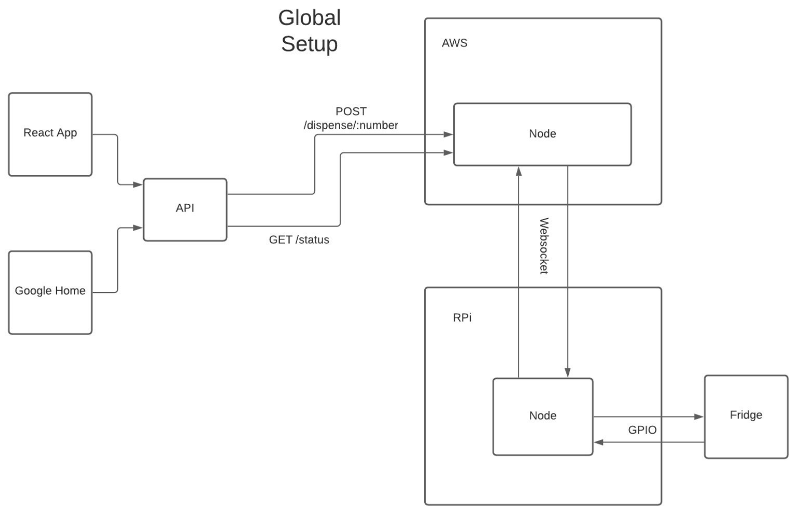
Software Phase 2

If time permits -

This stage builds off of stage 1, but it allows the fridge to me interacted with from the global internet instead of just the dorm wifi. Because of limitations of the UVM network, we cannot expose the API to the external internet. This is why we would be forced to implement a system like the above one.

We have moved the API portion of the Node server off of the Pi and onto a globally accessible AWS server. The hardware communication will be done over websockets from AWS to the Pi. This would be implemented with something like socket.io.

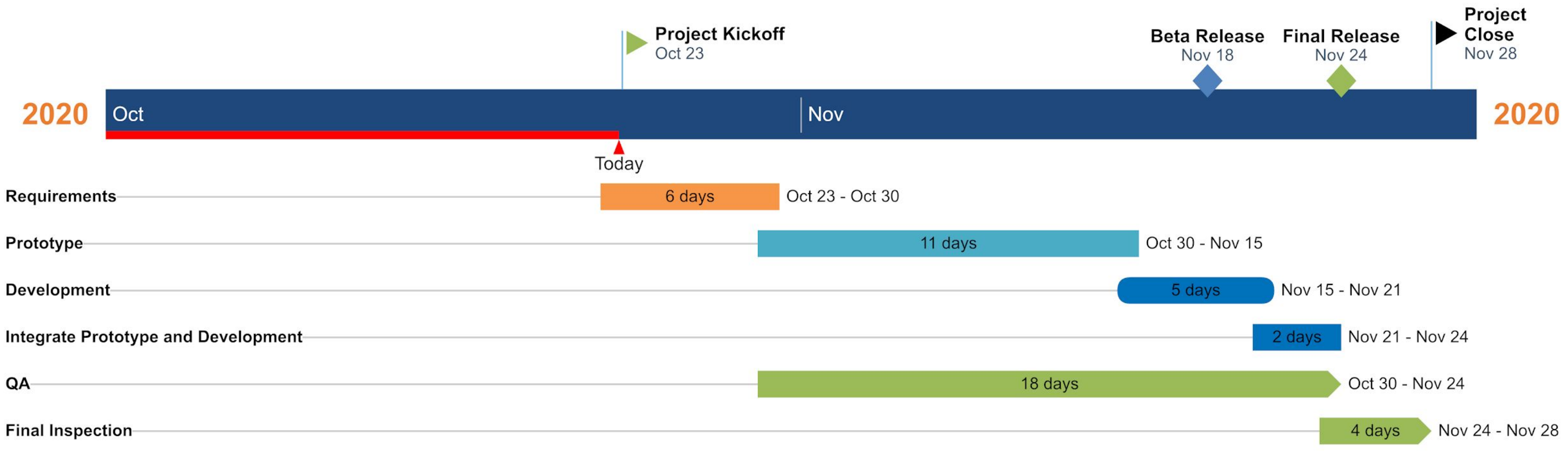
The React app and the google home would talk with AWS, AWS would talk with the Pi, the Pi will talk with the fridge, and then that data will flow the opposite direction back to the user - all in real time. Websockets are extremely fast, AWS's latency is extremely low. You would not be able to tell that your traffic is going from the dorm to Amazon and back to the dorm!



Budget

Development	$18/hr * 25 hours = $450
Engineering	$18/hr * 25 hours = $450
Raspberry Pi 4	$35
Mini Fridge	$60
Motors	2 x $15 = $30
3D printing	$0.5 / hr * 5 hours = $2.50
Wood	$20
Total	$1047.50

RootBeer Me Gantt Chart



Project Manager: Jared DiScipio

Target Market

The RootBeer Me is the ultimate solution for the college frat looking for high capacity liquid storage. Featuring voice recognition and a blazing fast response time, the RootBeer Me will keep the momentum of the party going. RootBeer Me utilizes a patented enveloped liquid (can) dispensing system designed to never jam, and it will never drop your drinks. Sick and tired of having to bend down and grab several cold ones to crack open with the boys? No worries, RootBeer Me comes with a detachable ramp capable of allowing auxiliary access to beverages from great distances!

References

NPM - https://www.npmjs.com/
ReactJS - https://reactjs.org/docs/getting-started.html
AWS - https://aws.amazon.com/lambda/
ParcelJS - https://parceljs.org/
GPIO - https://www.raspberrypi.org/documentation/usage/gpio/
TypeScript - https://www.typescriptlang.org/
Node - https://nodejs.org/en/
Stepper Motors - https://learn.adafruit.com/adafruits-raspberry-pi-lesson-10-stepper-motors