Automatic Cat Feeder

Group 3

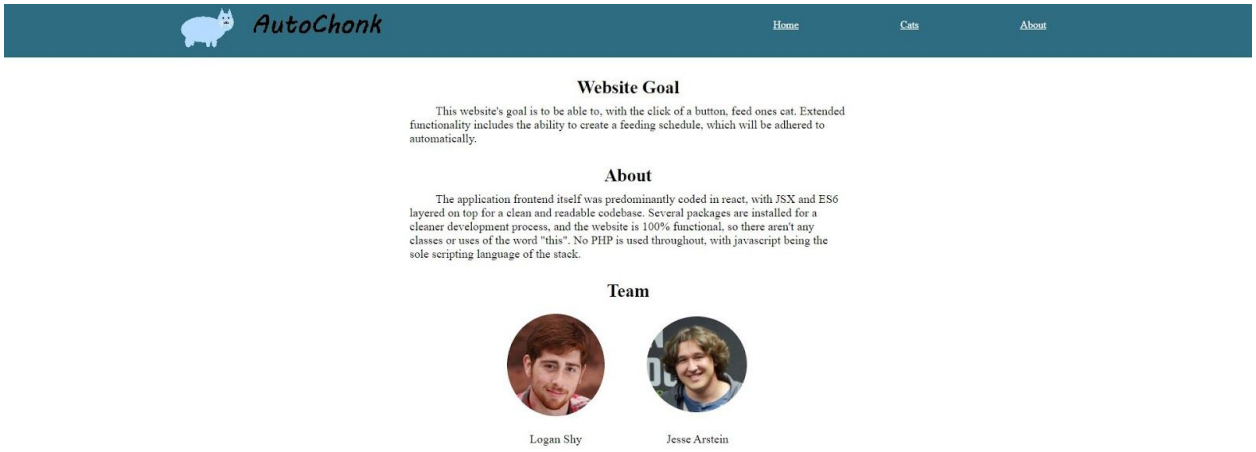
Logan Shy and Jesse Arstein

Github: https://github.com/Logan-Shy/CSCI491-final-project
Zip file containing node project submitted alongside PDF. Cloning the project is definitely the least painful option.

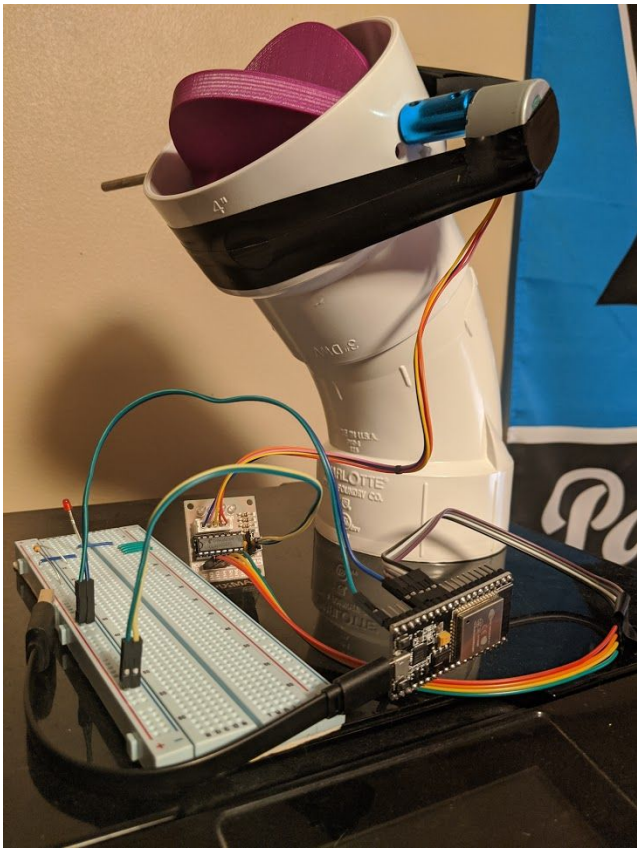
Objective

The main goal for this website was to help the owner of a feline friend to control their dry food intake. By providing an interface that allows a user to feed their cat with the push of a button, cat owners everywhere need no worry about hurrying home, or being out of town. By hosting it locally, a user simply needs to VPN into their home network, similar to interfacing with ones router.

Summary



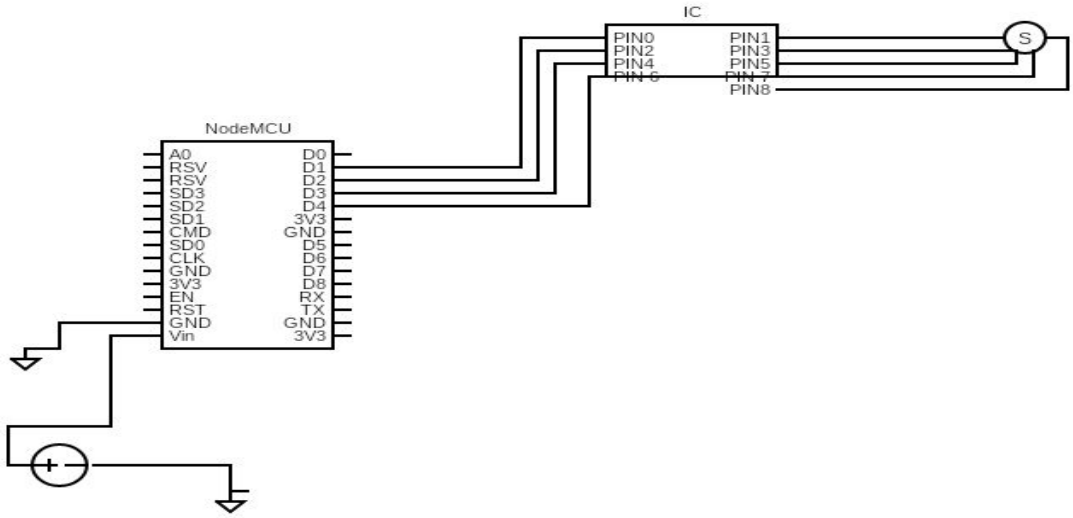
The frontend is a React[5] app, written almost entirely in javascript. The website is composed of an About, Cats, and Home Pages. However, only the first two are implemented. The About page contains info about the project, as well as links to myself and Jesse's Github[6] pages. On the cats page, there are several dummy cat profiles in place, each with their respective feed button. Since no API's exist that fulfill our needs, we went the hard route and made our own.



The backend was built on an ESP32[1] and was written in C. The backend contains code for connecting the microcontroller to a wifi network, hosting the website and handling all input and output control from the website to the stepper motor that drives the turning of the feeder. The Arduino[2] header libraries that were utilized are the Wifi, SPIFFS, ESPAsyncWebServer and Stepper libraries.

There is more to the backend that extends beyond just code. This project includes circuitry design and 3d modeling in order to build out and design the actual feeder portion of this project. A Monoprice Duplicator i3[3] was used to print the auger that was modeled to fit within a PVC pipe that will contain the

cat food. The 3d modelling was completed within Autodesk Fusion 360[4]. A 5v stepper motor was utilized to turn the auger when it was time to feed the cat. The design is simple but functional. Site says turn auger, auger turns, cat is fed.



Members

Logan Shy worked primarily on the frontend. The biggest initial hurdle simply selecting a framework to build from. We knew that we didn't want to have to use PHP, and Logan already had a bit of previous experience with React, so we decided to use create-react-app to start us off. From there, structuring and adding content to the website was trivial.

Jesse Arstein worked primarily on the backend and feeder design. The biggest hurdle within the backend was hosting a react framework within a microcontroller. With the way react runs, it was difficult for it to play nice within the ESP32. Getting C to send chunked javascript files proved to be a difficult task and if we were to design the project again we would utilize the simple client-server relationship with the ESP32 just being an extension of the server. Another hurdle was driving the stepper motor, the stepper motor has a library but to learn how to tweak the values so you get the correct amount of torque with little amperage and voltage you need to know how to time the bits being sent to your motor. A long time was spent learning how a stepper motor functions and when to time the transmission of bits to get the most out little electrical impulses. After a while of tweaking the library files and values I was able to achieve a functioning stepper motor with enough torque to move the large auger I mounted onto the motor.

Conclusion

I would say a big take away would be how the frontend and the backend meet. Working with APIs and storing data in databases was something I had never done in a class before, and getting the chance to implement a full stack application was invaluable. Learning javascript was

a unique experience, in that I've never written code like some of the .js files I produced. In terms of what worked, the React app functioned as expected, but our methods for hosting left us with unforeseen problems. Since React apps require some host to *serve* them, navigating to a static index.html will have little to no content. If I were to re-do this project, I would keep working with the React framework, however I would add a calendar function, to allow a user to schedule feeding times for their cat.

Ignoring troubles with the hosting of the website on an ESP32, designing the circuit and modeling the parts that went into putting a functional machine together was a rewarding experience. Understanding the design from a hardware level to a high level react application was well rounded project that utilized many different skill sets to create a true full stack experience.

References

[1] ESP32 http://esp32.net/
[2] Arduino https://www.arduino.cc/
[3] Monoprice Duplicator i3 https://www.monoprice.com/product?p_id=13860
[4] Autodesk Fusion 360 https://www.autodesk.com/products/fusion-360/overview#banner
[5] React https://reactjs.org/
[6] Github https://github.com/

Resources

Zybook https://learn.zybooks.com/
Professor Daniel Defrance https://www.cs.montana.edu/defrance/index.html
CSCI 491