Louie Hernandez
CS437: IoT
UIUD: lgherna2
Team Name: Public Safety
Final Project: Automated Pet Feeder

Video Link: https://www.youtube.com/watch?v=us706tWY3ec

Repo: https://github.com/louiehernandez95/CS437-IoT-Final-Project

Motivation

This semester I had the pleasure of taking care of 5 dogs (3 chihuahuas, 1 husky, 1 german shepherd), but making sure they're all well fed and have eaten at least breakfast and dinner can get stressful. A pet owner's worst fear is the possibility of their pet starving and getting sick. We treat our pets as part of the family so it is imperative that proper care is given to them. Because of these reasons I want to take the skills I have learned for this class and create an IoT based Pet Feeder to help pet owners with their furry best friends. The feeder system for the first MVP (minimum viable product) should be able to automate dispensing food or you should be able to dispense food remotely. The feeder system will be designed to dispense precise amounts of food at a time, reducing the amount of time owners spend on feeding and monitoring household pets. I was thinking the automated feeder system gives can be programmed in such a way that it can be controlled via a web application with a good user friendly interface

Reference paper:

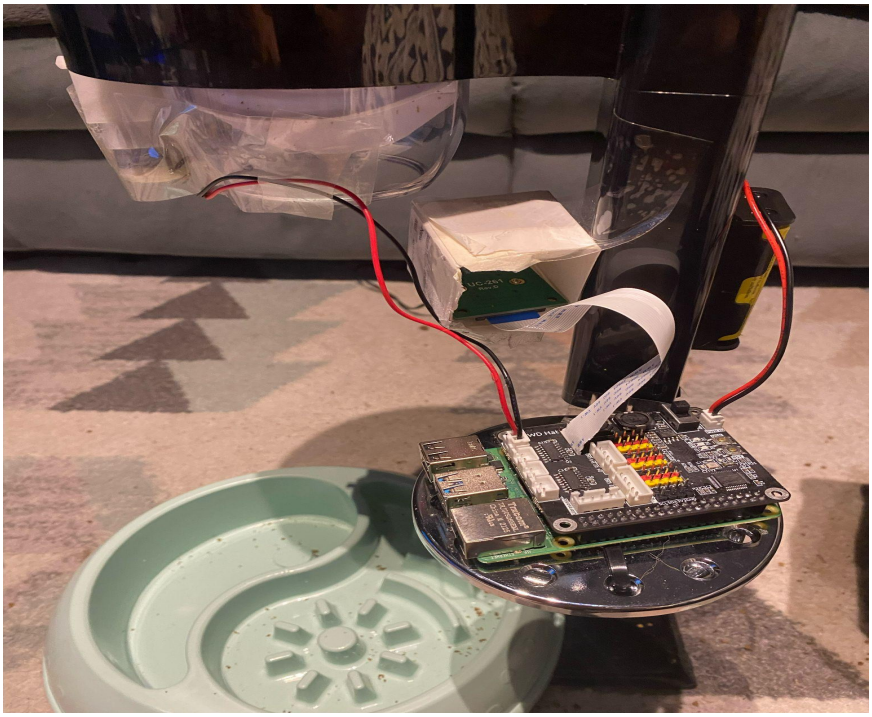
https://www.ijser.org/researchpaper/Internet-of-Things-based-Pet-Feeder-Automation-using-Raspberry-Pi.pdf

https://medium.com/@shehanb/how-to-build-your-own-pet-feeder-diy-83d738b6f931

Technical Approach

Final Product

Back View



Side View

Items Used:
x1 Honey-Can-Do Dry Food Dispenser, Single Control, Black/Chrome

https://www.amazon.com/dp/B00013K8O4?ref=ppx_yo2ov_dt_b_product_details&th=1

x1 Laboratory Grade Metalware Set - with Support Stand (8.3"x5.5"), 2 Retort Rings (Dia. 2.2"/2.6"), Rod (Length 19.7") and Burette Clamp

https://www.amazon.com/dp/B07THX1NB5?psc=1&ref=ppx_yo2ov_dt_b_product_details

x1 Lens Board OV5647 Sensor for Raspberry Pi Camera, Arducam Adjustable and Interchangeable Lens M12 Module, Focus and Angle Enhancement for Raspberry Pi 4/3/3 B+

https://www.amazon.com/dp/B013JTY8WY?psc=1&ref=ppx_yo2ov_dt_b_product_details

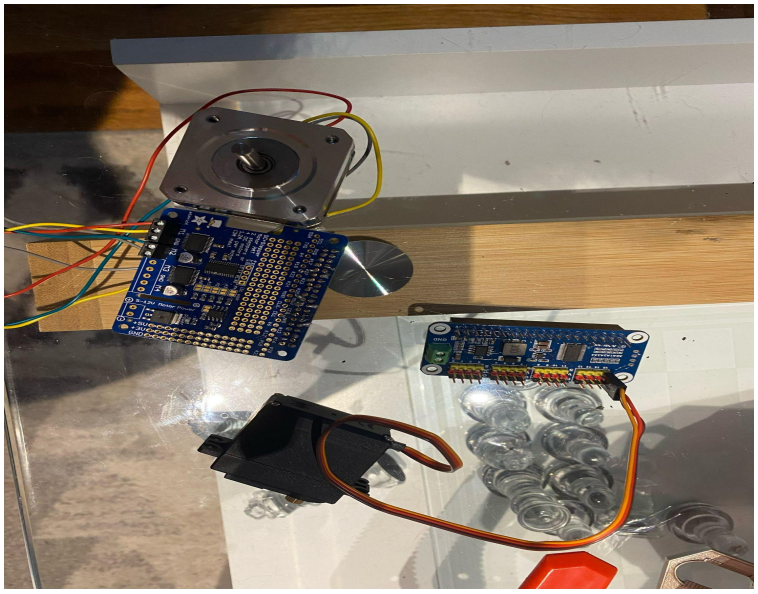
x1 SunFounder Raspberry Pi Car Robot Kit, 4WD HAT Module, Ultrasonic Sensor, Velocity Measurement Module etc. Electronic DIY Robot Kit for Teens and Adults, Raspberry Pi/TF Card/Battery not Included

https://www.amazon.com/dp/B087QKRX5J/ref=sspa_dk_detail_0?psc=1&pd_rd_i=B087QKRX5J&pd_rd_w=8RQQn&pf_rd_p=0c758152-61cd-452f-97a6-17f070f654b8&pd_rd_wg=qV0gq&pf_rd_r=QDDRDB3RBAD8FVRC6YP3&pd_rd_r=f6d5e77c-5fcd-484a-b6fa-1bb3363db09a&s=toys-and-games&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyMUpYMUZFQUVUR1hQJmVuY3J5cHRlZElkPUEwNjMxNjQ5MkpJNjlQREtaTlNLSCZlbmNyeXB0ZWRBZElkPUEwNzg3NTIyMkFaQlQwRDlPOUE5SyZ3aWRnZXROYW1lPXNwX2RldGFpbCZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=

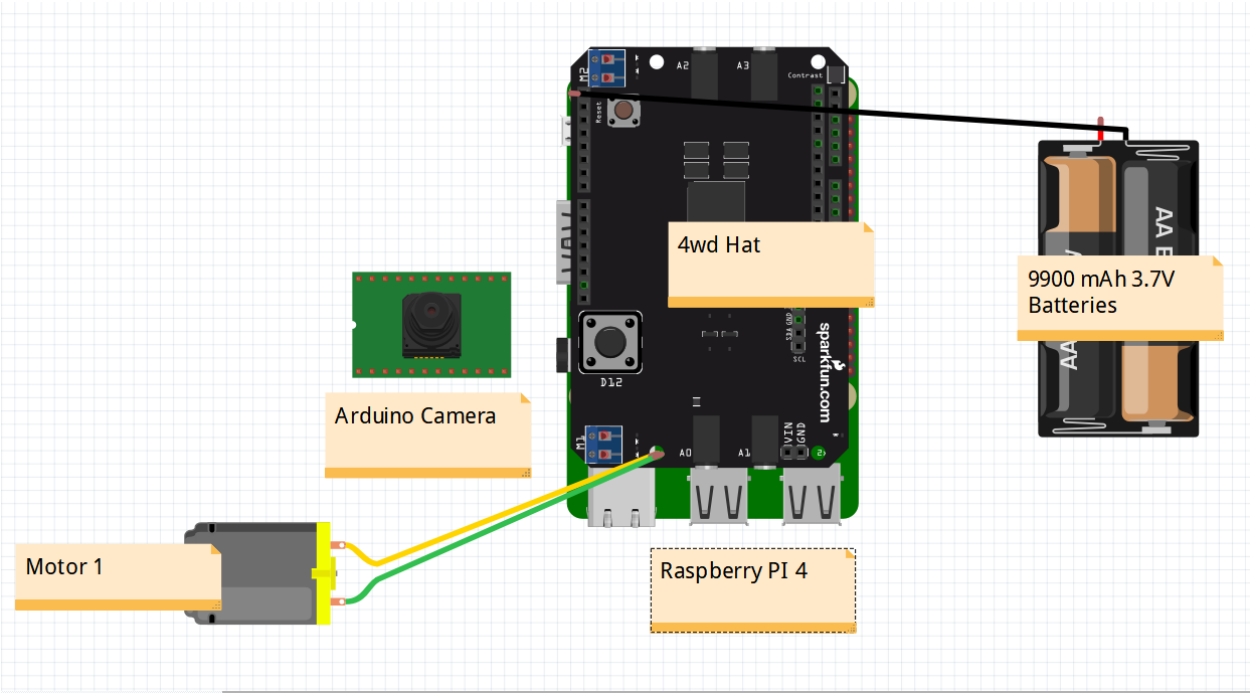
x1 Raspberry PI 4
https://www.amazon.com/Raspberry-Model-2019-Quad-Bluetooth/dp/B07TC2BK1X/ref=sr_1_3?crid=1XZ87RX0NIZ2U&keywords=raspberry+pi+4&qid=1652652346&sprefix=raspberry+pi+4%2Caps%2C97&sr=8-3

x1 Dog food bowl (any is fine)

Prototype:
Before I decided on the DC motor from Lab 1, I tried using a servo motor and stepper motor. Normal servos can move an arm typically 270 degrees, but not a full rotation. After realizing I did not really care about the absolute position of the wheel; I just wanted it to spin a certain amount of times so I decided on using a stepper motor. I bought the following: https://www.adafruit.com/product/324 . With the food dispenser that I bought I soon realized the stepper motor was too big and too heavy to reasonably attach to the propeller in the food dispenser. A stepper motor is a sound solution, but because of my lack of resources after buying two motors I decided to reuse the DC motor from Lab 1 alongside the 4wd hat and battery holder.



Architecture:
The circuit diagram for the hardware of the project. Besides the 4-WD Hat that I attached, there were only three connections made with the Raspberry Pi: 1) Arduino Camera 2) DC Motor 3) Battery Pack



How Data Flows:
The whole system is user driven. I took a top down approach when designing the system for this IoT device. In terms of user experience I figured users would want to know how much food there already is in the bowl so that's why I added a livestream of the dog bowl in real time as a must have feature for the MVP. I wanted the user to feel empowered that they control how much is being dispensed so that is why I did not make it so that it dispenses food in exact portions. It is all controlled by the push of a button. A future feature could be keeping track of how much food is being dispensed at a time and setting "alarms" so that that exact amount can be dispensed at let's say 1:00 pm everyday.

This is how data flows in the system:
UI (electron app) -> API endpoint gets hit -> API communicates to server running in raspberry PI using a open ip socket connection -> raspberry PI server gets hit -> command processed -> response message to API through socket connection -> HTTP response to UI

Hardware Details:
In my proposal I thought I wanted to use a servo motor but like I mentioned earlier getting a full rotation would be impossible/difficult so I decided to either use a stepper motor or DC motor. I

ended up using the DC motor from Lab 1. I did not want to attach the motor to the propeller so I used the pipe from the picar wheel to attach the two. I made an elevated stand so that the raspberry Pi wire between it and the DC motor did not have to be lengthened, I bolted the battery holder to the back of the dispenser holder. I positioned the picamera so that it would be pointing downwards to the food bowl. Everything else was just connecting the wires to what was specified in the diagram.

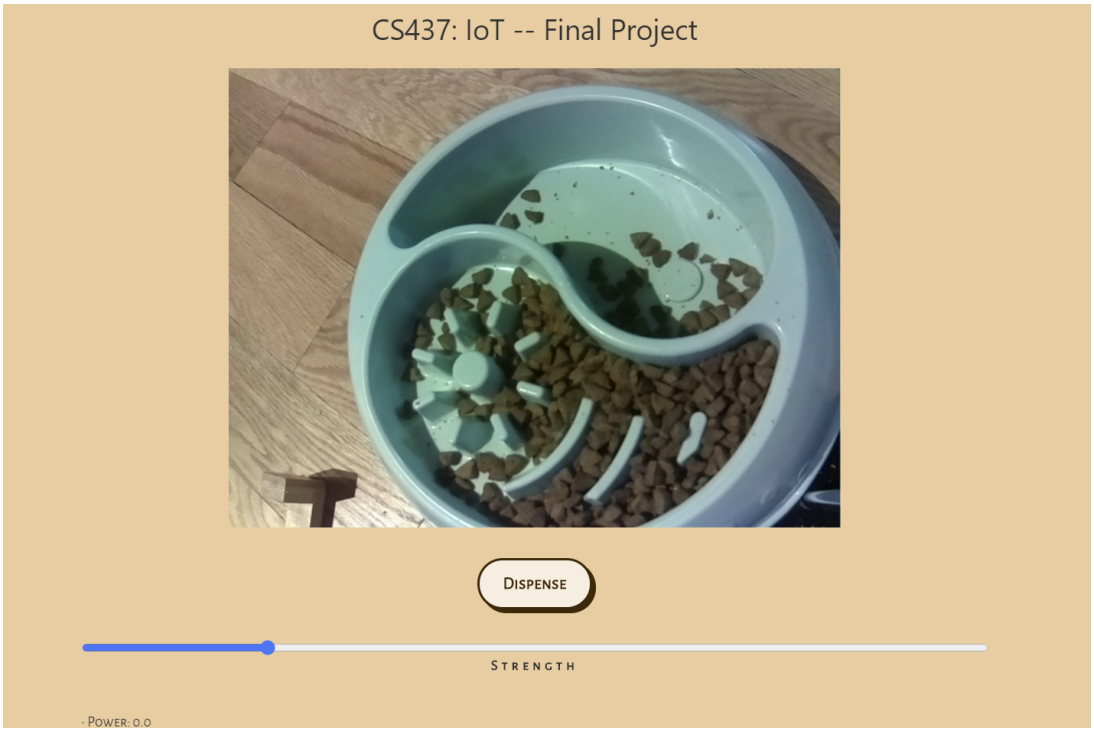
Implementation Details

All of the backend code was written in Python. While the UI was constructed using HTML/CSS/JS using an electron app shell. The only package I use in the frontend is the electron package, everything else is written using vanilla HTML and Javascript. I use the hypertext transfer protocol (HTTP) so that the UI can communicate with the backend api server. The backend API is written in Python and I use the micro web framework Flask to construct my API. Depending on what API endpoint gets hit it will construct a message and send that data over to my raspberry PI over wifi using a socket. I use my raspberry pi's IP address to open that connection, I use the socket library for this. I use the same IP address and port when I am running the server in my raspberry pi. For the wifi-server.py file to operate the motor I reuse the code from the picar-4wd repository https://github.com/sunfounder/picar-4wd/tree/master/picar_4wd . The files I needed were: i2c, pin, pwm, motor, and utils. I refactored the code so that when I initialize the wifi server file it will only initialize one motor which I only needed. The GPIO pin I used for motor 1 was "D4". Once the server in the raspberry pi gets hit it will decode the message and execute code depending on what the command was. After that it will send a message back to the client based on the command hit. For the livestream, I used the following open sourced code to help do that. https://gist.github.com/n3wtron/4624820

How it basically works is it uses the following package to start up a http server "http.server", when the endpoint gets hit (GET) the picamera module that it uses will begin to continuously capture video from the picamera as a mjpg file. What will be returned is HTML, we set the content type to text/html and we will return a byte array with html with an img element with the cam.mjpg file attached to it. One limitation is that it can only do one connection at a time. When all of the cogs are put together the end result is a user controlled food dispenser with a video livestream attached to it, you can also adjust the strength of the propeller in case of a jam.

The following medium article was super helpful, it helped give me a vision of what I wanted out of my final project.
https://medium.com/@shehanb/how-to-build-your-own-pet-feeder-diy-83d738b6f931

I expanded on what was already there by adding a livestream feature and a nice UI the user can use.



UI

Results

Overall, I think the project went well. The biggest hiccup I had was that I did not drill a hole to the center of the propeller enough. Drilling a couple millimeters off made me have to position the DC motor at an angle. If I were to do this project again I would do more accurate measurements. If I were to do this project again I would try and see how I can minimize costs. If this was a commercial product I would need to find a way to reduce costs since a raspberry pi is about 150 dollars. But then again this was just a POC. The key takeaway I took from this project was that I had an idea, I investigated multiple trade offs on all of the pieces I chose, you don't know how many hours I spent on product design. Figuring out what food dispensers I can easily modify, also evaluating tradeoffs on motors to use. If I had a saw I would have also put the motor to the back of the dispenser. Crafting the UI was also tricky, I ran into multiple instances where I got into an infinite loop because the motor stopped and had no way of ending. That's why I decided to send a stop() message when the user lets go of the button click.

Overall

I am proud of this project. I got to work on something from top to bottom! I got complete say on what pieces I wanted to use, I felt like a true IoT engineer. How I expanded upon all of the research projects I looked up was that I added a live streaming feature and a UI as well as fleshing out an API. If I had more time I would have incorporated some AWS tools like a lambda

function. A future feature could have been calculating the average time spent dispensing so there could have also been an “auto-dispense” button.

I had a great time in this class and I wish you guys well!