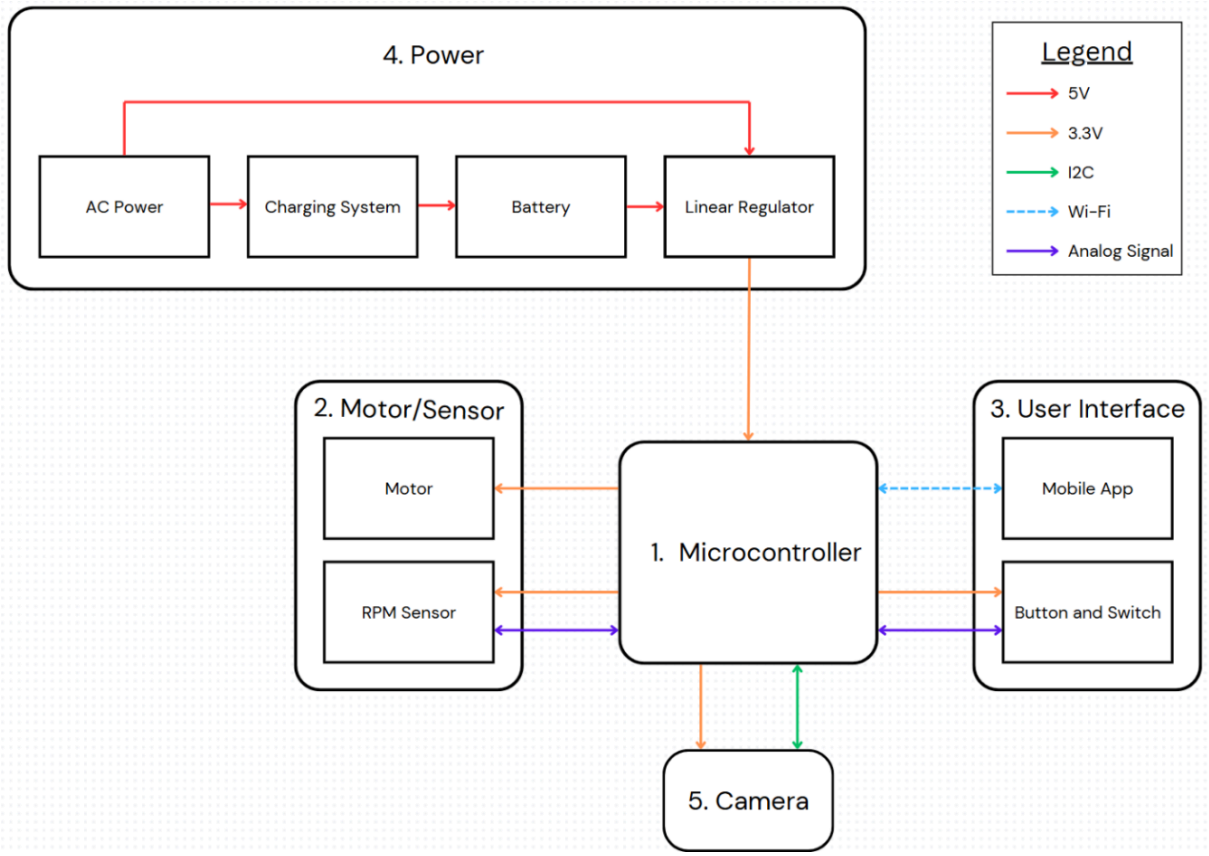
2/2/25 - 2 hours

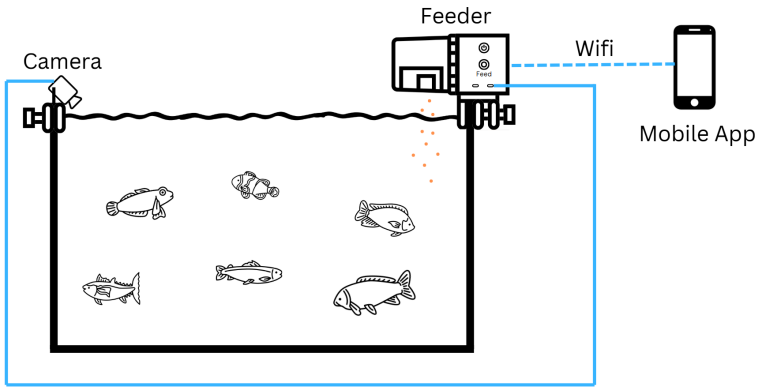
Decide what microcontroller would be best for our project.

We decided to use an ESP32 because my teammate Colby's roommate had a hard time using the STM32 and recommended using the ESP32. Still figuring out how we want to estimate how much food is remaining (either sensor or scale).

2/10/25 - 3 hours

Worked on the project proposal and created both the block diagram and visual aid, showing the interactions between all the subsystems and how the design will look when used on a fish tank, respectively.





2/10/25 - 2.5 hours

Worked on the project proposal and decided to use the ESP32C3MINI1 for our microcontroller as there is a lot of documentation on it and is compact and cost effective.

https://www.espressif.com/sites/default/files/documentation/esp32-c3-mini-1_datasheet_en.pdf

2/16/25 - 3 hours

Got Platformio set up with Visual Studio Code to program the ESP32 dev board. Tried connecting it to Wi-Fi and seeing if it responds to basic inputs.

pinMode(pinNumber, OUTPUT) and then set the pin high using digitalWrite(pinNumber, HIGH)

2/24/25 - 1 hour

Decided to instead go with the ESP32-S3-WROOM-1 as it was readily available through the E-Shop and still has plenty of documentation.

2/25/25 - 2 hours

Researched and determined what GPIO pins on the dev board to use for our project. Mainly focusing on what pins we can use for the OV7670 camera.

OV7670	ESP32 Dev board	
GND	GND	
3.3V	3V3	
SCL	D22 (IO38)	
SDA	D21 (IO37)	
VS	D34	
HS	D35	
PCLK	D33	
MCLK	D32	
D7	D4	
D6	D19	D12
D5	D13	
D4	D14	
D3	D18	D15
D2	D16	
D1	D17	
D0	D27	
RST	3V3	
PWNN*		

*Suggested we connect PWNN to ground with a 10k-ohm resistor

https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

https://www.espressif.com/sites/default/files/documentation/esp32-s3_technical_reference_manual_en.pdf#iomuxgpio

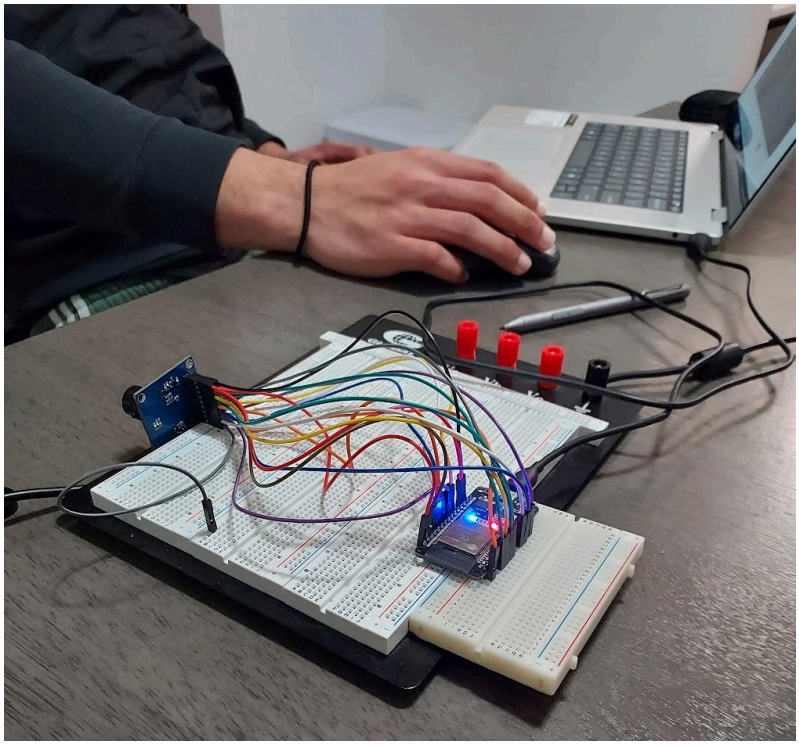
https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

https://lastminuteengineers.com/esp32-pinout-reference/

https://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf

2/26/25 - 3 hours

Helped Jeremy with the camera using the dev board and tried to get it to work (no luck).



2/28/25 - 1 hour

Made revisions to proposal. I accidentally put AC power in our power subsystem, but it should be DC (power from a wall outlet is AC, but by the time the power reaches the board, it's DC)

3/2/25 - 3 hours

Came up with a pinout for the ESP32-S3-WROOM-1, including the camera and USB-to-UART bridge (motor and sensor just need one pin; doesn't matter which ones).

OV7670	ESP32 Dev board	ESP32-s3-wroom-1 (Use this on KiCad)
GND	GND	1 (GND)
3.3V	3V3	2 (3V3)
SCL	D22 (IO38)	39 (IO1)
SDA	D21 (IO37)	38 (IO2)

VS	D34		17 (IO9)
HS	D35		18 (IO10)
PCLK	D33		19 (IO11)
MCLK	D32		20 (IO12)
D7	D4		4 (IO4)
D6	D19	D12	5 (IO5)
D5	D13		6 (IO6)
D4	D14		7 (IO7)
D3	D18	D15	8 (IO15)
D2	D16		9 (IO16)
D1	D17		10 (IO17)
D0	D27		11 (IO18)
RST	3V3		2 (3V3)
PWNN*			

*Suggested we connect PWNN to ground with a 10k-ohm resistor

(USB-to-UART Bridge)

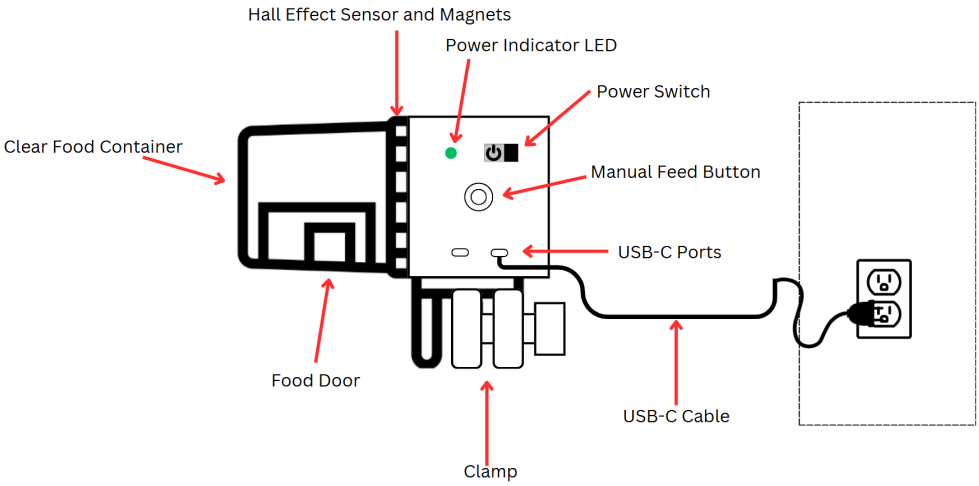
RX	37 (TXD0)
TX	36 (RXD0)
GND	1 (GND)
VCC	2 (3V3)
DTR	DTR (From programming circuit)
CTS	RTS (From programming circuit)

3/5/25 - 4 hours

Worked on the design document and created our schedule for completing the project. Also, signed up for a breadboard demo time.

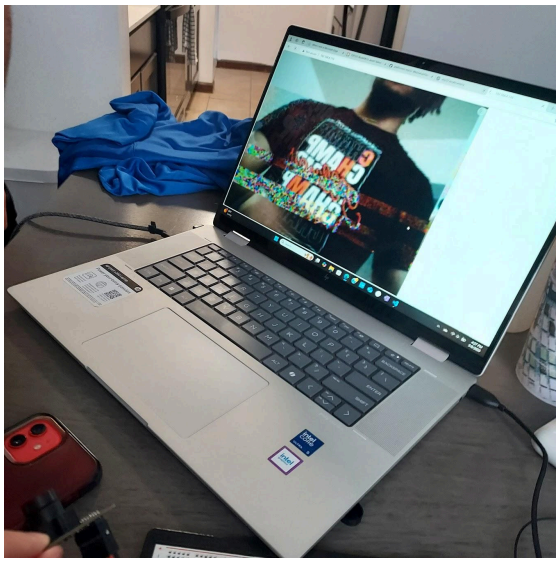
3/6/25 - 2 hours

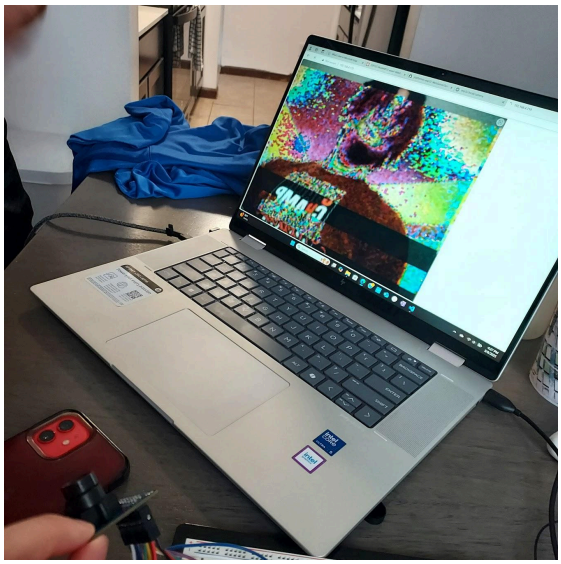
Wrote about the physical design in the design document and created a diagram of the physical design concept of the feeder, including the power LED, power switch, manual feed button, USB-C ports for power and the camera, etc.



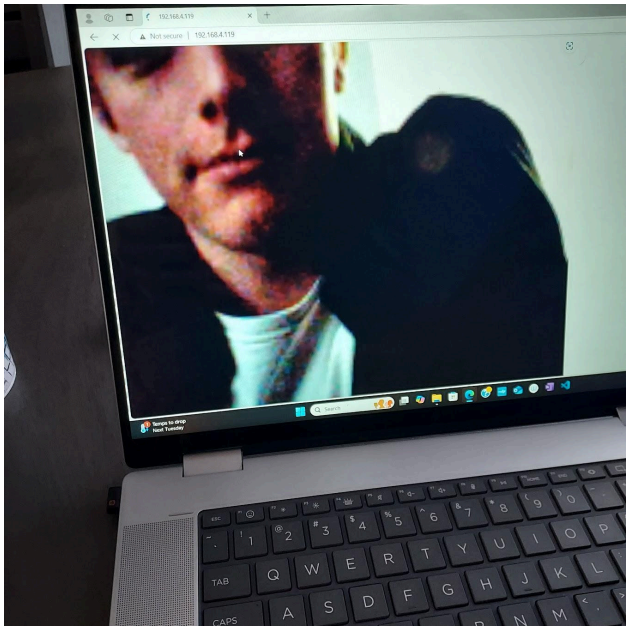
3/8/25 - 4 hours

Come up with logic for the motor functionality and what pins we are gonna use for the motor, sensor, and button. Helped Jeremy with the camera(one that Colby had from ECE385) and eventually got it to display on a server (although the quality was not the best).





We then replaced the camera with a new one that Jeremy bought and the rainbow artifacts went away, but the quality was still not the best.



3/10/25 - 2 hours

Wrote code for the motor functionality logic. We found that the hall effect sensor outputs values over about 1900 when it detects a magnetic field, so we set this value accordingly in the code.

```
#include <Arduino.h>
#include <WiFi.h>

void setup() {
  Serial.begin(115200);
  pinMode(26, INPUT); // Hall Effect Sensor value
  pinMode(25, OUTPUT); // Motor pin
  pinMode(12, INPUT); // Feed button
}

bool isConnected = false;

void loop() {
  uint16_t hallOutput = analogRead(26); // Read the hall effect sensor value
  uint16_t D12 = analogRead(12); // Read the value of the feed button

  printf("Hall Output: %d\n", hallOutput);
  if (hallOutput <= 1900) { // Motor rotates at default
    digitalWrite(25, HIGH);
  } else if (D12 != 0) { // Else if the feed button is pressed, activate the motor
    digitalWrite(25, HIGH);
  } else { // If the hall effect sensor detects the magnet, stop the motor
    digitalWrite(25, LOW);
  }
}
```

3/13/25 - 1 hour

Research ways to control the ESP32 remotely. I don't know about a mobile app (maybe a web server for now, but not for long).

https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

3/24/25 - 2.5 hours

Figure out how I want to create an app. Web app will most likely be easier (TA suggested). Looks like using Firebase will be a good way to create a web app (and we can create a mobile app if we want later), and it can host it. Code written in C++, Javascript, and HTML

https://randomnerdtutorials.com/firebase-control-esp32-gpios/

https://randomnerdtutorials.com/control-esp-gpios-firebase-web-app/

Set up a Firebase project called SAFF (Schedulable Automatic Fish Feeder). Create a login page for our app and show the manual feed button on the app once you log in.

3/26/25 - 6.5 hours

Set up the Realtime Database on Firebase for the GPIO pins for the motor and sensor. Researched how easy it would be to stream a video through Firebase, if possible.

https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/

https://randomnerdtutorials.com/esp32-cam-display-pictures-firebase-web-app/#more-107976

3/27/25 - 2 hours

Created a favicon (the image that's next to the title of the web page on the tab) for our web app. And also added icons next to the titles of the login title, control page, and manual feed button (and make the user interface look nice)

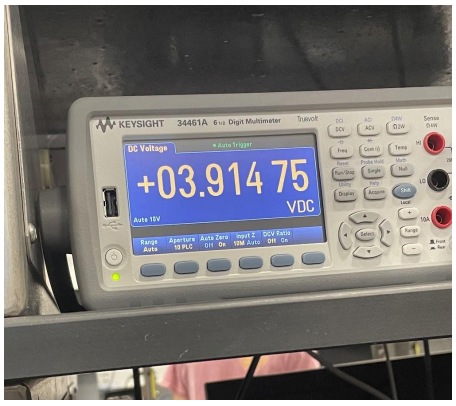
https://fontawesome.com/

3/28/25 - 1 hour

Research ways to be able to display a Wi-Fi symbol on the app only when the microcontroller is actually connected to Wi-Fi.

3/29/25 - 5 hours

Help Jeremy solder our second-round PCB since the first-round one that Colby soldered yesterday wasn't working. We decided to bake on all of the surface mount components as that would be easier and quicker. The power LED on the PCB for some reason wasn't lighting up when we plugged it in. The USB was only giving 3.91475 V and the wire for the power switch wasn't getting any voltage. Jeremy thinks it's because of the transistors.



3/30/25 - 3 hours

We tried to program the board that's getting power now, but it isn't working. It may be the USB-to-UART bridge that is bad. Jeremy ordered a bunch of new bridges to see if that's the problem.

4/2/25 - 3 hours

Worked on individual progress report.

4/3/25 - 2 hours

Worked with Jeremy and tried some of the new bridges to see if they would work (but they didn't). We think we might have bricked them because the website for the bridges says that they only work on older drivers, so if we have newer ones it'll brick the bridge.

4/6/25 - 3 hours

Went to the lab to test the rest of the bridges that we ordered (still no luck). I tried getting different drivers, using Arduino IDE instead of PlatformIO. A friend gave me another bridge that worked for him. Apparently some don't work with Windows 11 though.

4/7/25 - 4 hours

We checked with the TA to make sure our programming circuit was correct and that that wasn't the problem (hers was the same as ours). Jeremy and I tried programming a PCB after Colby used flux on the ESP32. We finally got it to work with one of the bridges, but we had to press the reset button really quickly for it to work. We think it's a soldering issue.

4/11/25 - 2 hours

Worked on the app more, now that the board is able to be programmed.

4/14/25 - 2 hours

Figure out how to schedule times for the ESP32 to specifically do things in order to set up the schedule feeding times feature of the app.

https://randomnerdtutorials.com/epoch-unix-time-esp32-arduino/

4/17/25 - 3 hours

Get scheduled feeding times to work. Get real time from NTP server. Also, figured out how to get input from the user for the feeding times and store the values on the database, so that feeding can be triggered when the real time matches.

https://randomnerdtutorials.com/esp32-date-time-ntp-client-server-arduino/

4/21/25 - 4 hours

Make the user interface look nice and format things correctly using javascript and HTML.

https://www.w3schools.com/js/

https://www.w3schools.com/html/html_computercode_elements.asp

4/23/25 - 3 hours

Try and get the scheduled times to display after the user enters them. And tried to make it so they can remove them (which also removes them from the database).

4/26/25 - 2 hours

Add more data outputs on the database for the food remaining values that Jeremy came up with. Displayed the food remaining percentage on the app.

4/27/25 - 4 hours

Figured out how to display the Wi-Fi symbol and finished the app.