ECE 445 Spring 2025

SENIOR DESIGN LABORATORY

Final Report

Schedulable Automatic Fish Feeder

TEAM #18

BRANDON MACINTOSH(bm53@illinois.edu)

COLBY STEBER(csteber2@illinois.edu)

JEREMY RICHARDSON(jrr13@illinois.edu)

TA: Sanjana Pingali

7 May 2025

Abstract

This paper presents the design, development, and implementation of the Schedulable Automatic Fish Feeder (SAFF) aimed at improving the convenience and reliability of feeding aquarium animals. The system integrates a microcontroller-based PCB, a user interface for scheduling feed times, and manual feeding through an app, manual feeding on the unit, a mechanical mechanism to dispense the food, a backup battery for temporary power loss, and a camera to view the aquarium. Key design goals included cost-efficiency, commercializability, ease of use, modular construction, and system reliability. The final prototype features a working camera module, backup battery, a web app interface to schedule times, and feed accessible at *auto-fish.food*, a micro-metal gear motor to control rotation of the feeder, and a manual feed button on the physical device. Testing demonstrated consistent and accurate feed dispensation across multiple trials, validating the design's functionality and effectiveness. The project offers a scalable and practical solution for hobbyists and aquarium enthusiasts seeking an automated, customizable, ethical feeding system.

Contents

1. Introduction

1.1 Purpose

Fish feeders currently on the market are limited on how much convenience they give fish owners when they are away from their tank. If the user wants to feed their fish at a certain time, they usually have to set a timer for 12 or 24 hours in advance to feed them. There is also no reassurance that their fish is actually being fed and eating. Owners just have to assume that the machine is working as intended. This poses a major problem when gone for extended periods of time, such as spring break.

With our fish feeder, the user is able to feed their fish from any location by using a web app, which also has the ability to schedule up to four exact times they want the feeder to dispense food, allowing them to have customized feeding times. In addition, the feeder has a sensor that detects when the food container rotates and sends a notification to the web app so the user can ensure that their fish was fed. The feeder is connected to power through a USB-C cable and brick supplied by the user. If the power goes out or if the feeder is not being supplied with DC power from the USB-C port, it switches to battery power. This solution required a PCB, ESP32, rotating canister, motor, hall effect sensor, web app, power system, and camera.

1.2 Functionality

The SAFF has a couple of high-level functionalities. The user is able to manually activate the feeder via the physical manual feed button or the manual feed button on the app. The user is able to view the camera feed via the view button on the app. The user is able to schedule times to feed through the app to activate the feeder. The user receives a notification once the feeder has completed a 360-degree revolution. The user is able to view the state of the feeder, whether it is

in standby or feeding. These relate to the overall purpose of the project by activating the feeder with scheduled times or manually, along with viewing the camera feed from the feeder itself.

1.3 Subsystem Overview

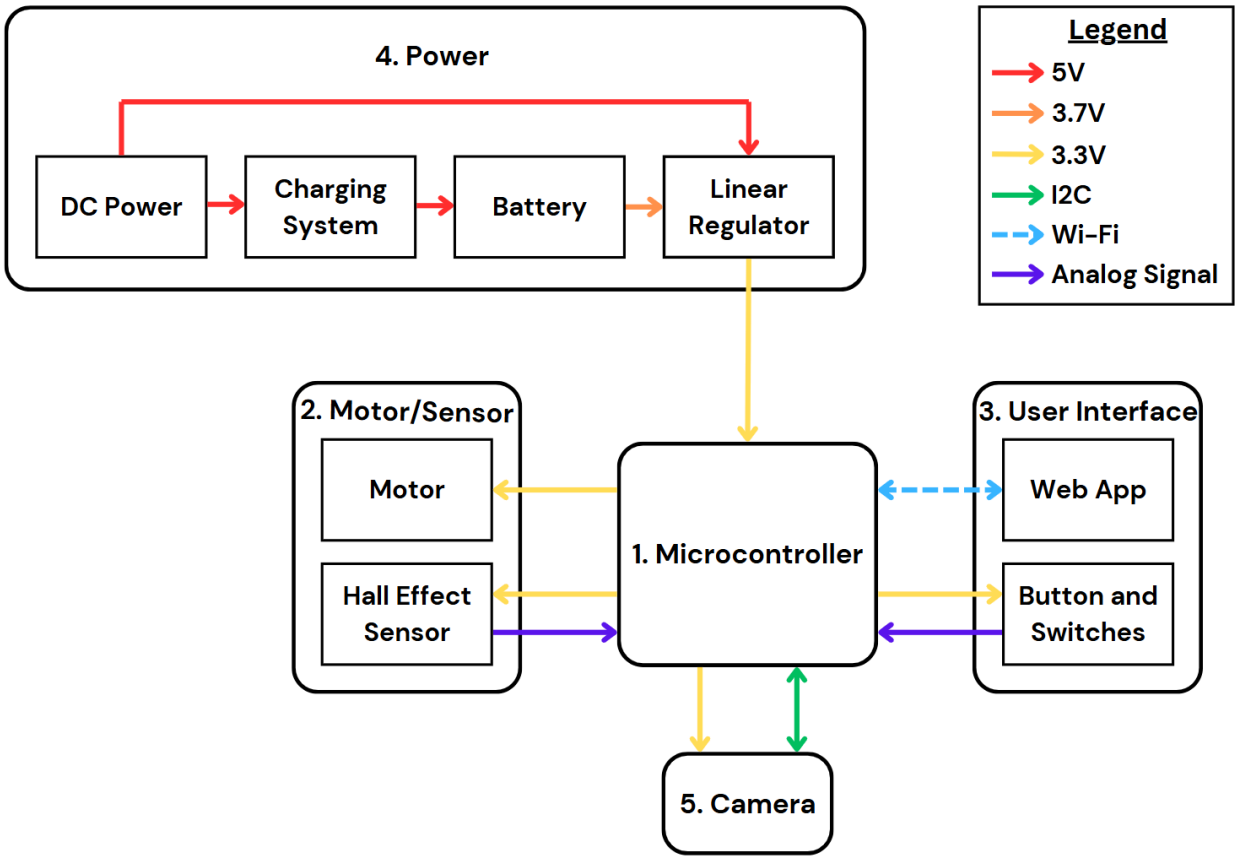

Figure 1: Block Diagram

1.3.1 Microcontroller Subsystem

The microcontroller subsystem manages voltage control for the motor, camera, and hall effect sensor while also handling data communication with the app and camera, as can be seen in *Figure 1* above. The microcontroller subsystem uses an ESP32 microcontroller with Wi-Fi capabilities to communicate with the User Interface system. It sends 3.3 V to the sources of two 2SJ652 transistors. The drains of the transistors are connected to the motor and camera. When the camera button is selected on the app, a signal is sent from the ESP32 to the gate of the

transistor, allowing for voltage to be delivered to the camera subsystem. Similarly, if the manual feed button is pressed, the feed button on the app is pressed, or the scheduled feed time is reached, a signal is sent to the gate of the other transistor, and 3.3 V is sent to the motor. When the camera is powered, it will send data to the ESP32 through I2C. The hall effect sensor is always powered and read by the ESP32.

1.3.2 Motor and Sensor Subsystem

This subsystem consists of a 3 V micro metal gearmotor that is connected to the main PCB via a PMOS transistor [1][2]. The transistor takes input power from the linear regulator and a signal to switch on from the ESP32. The output shaft holds the container of food. This container has a magnet glued on the rotating section so that a hall effect sensor can detect when it rotates to ensure that the food is actually dispensed.

1.3.3 User Interface Subsystem

The user interface subsystem is responsible for performing different actions based on user inputs. The two main parts of this subsystem are the app and the physical button and switch on the feeder. The feeder has a manual feed button on it that, if pressed, will give the motor power, and food will be dispensed. The feeder also has two power switches. One turns on and off the DC power, and the other turns on and off the battery power. The app has the ability to manually feed as well, trigger the camera feed, and set the feeding schedule. When the manual feed button is pressed on the app, a signal is sent from the app to the ESP32, which triggers the motor signal, which initiates the feeding procedure. When the camera icon is selected on the app, a signal is sent from the app to the ESP32, which triggers the camera signal and delivers the camera power. The camera sends pixel data back to the ESP32, and the ESP32 sends that data to the app continuously until the camera feed is closed. When a new feeding schedule is set, that data is

sent to the ESP32, and the feeding time is updated in the chip's memory. The app also displays information about the feeder, such as whether or not the feeder is connected to Wi-Fi and the estimated amount of food left in the canister.

1.3.4 Power Subsystem

This subsystem consists of two Schottky diodes that are used to switch between USB-C power and battery power, and another IC to control the charging of the battery. The battery is a 4.63 Wh, 3.7 V battery that is used as a backup power source for when power is lost through the USB-C connector. There are two switches on the unit: one to shut off USB-C power and one to shut off battery power. When USB-C power is restored, the charge IC checks the voltage of the battery and begins to charge the battery if needed. When USB-C power is available, the unit uses USB-C power. The battery is solely for backup and can provide ten hours of power under heavy use.

1.3.5 Camera Subsystem

The camera subsystem supplies the camera feed that is viewed in the app. The camera module being used is the OV2640. The OV2640 receives 3.3 V from the 2SJ652 transistor drain when the ESP32 sends a signal. When the OV2640 receives power, it sends data back to the ESP32 through the I2C Bus.

2. Design

2.1 Equations and Simulations

2.1.1 Power Switching Simulation

Shown below in *Figure 2* is the simulation that was used to determine if the Schottky diode that the self-service electronics shop had would work for our project. We can see that the node after the diode is still producing 3.5 V, which is over our 3.3 V threshold.

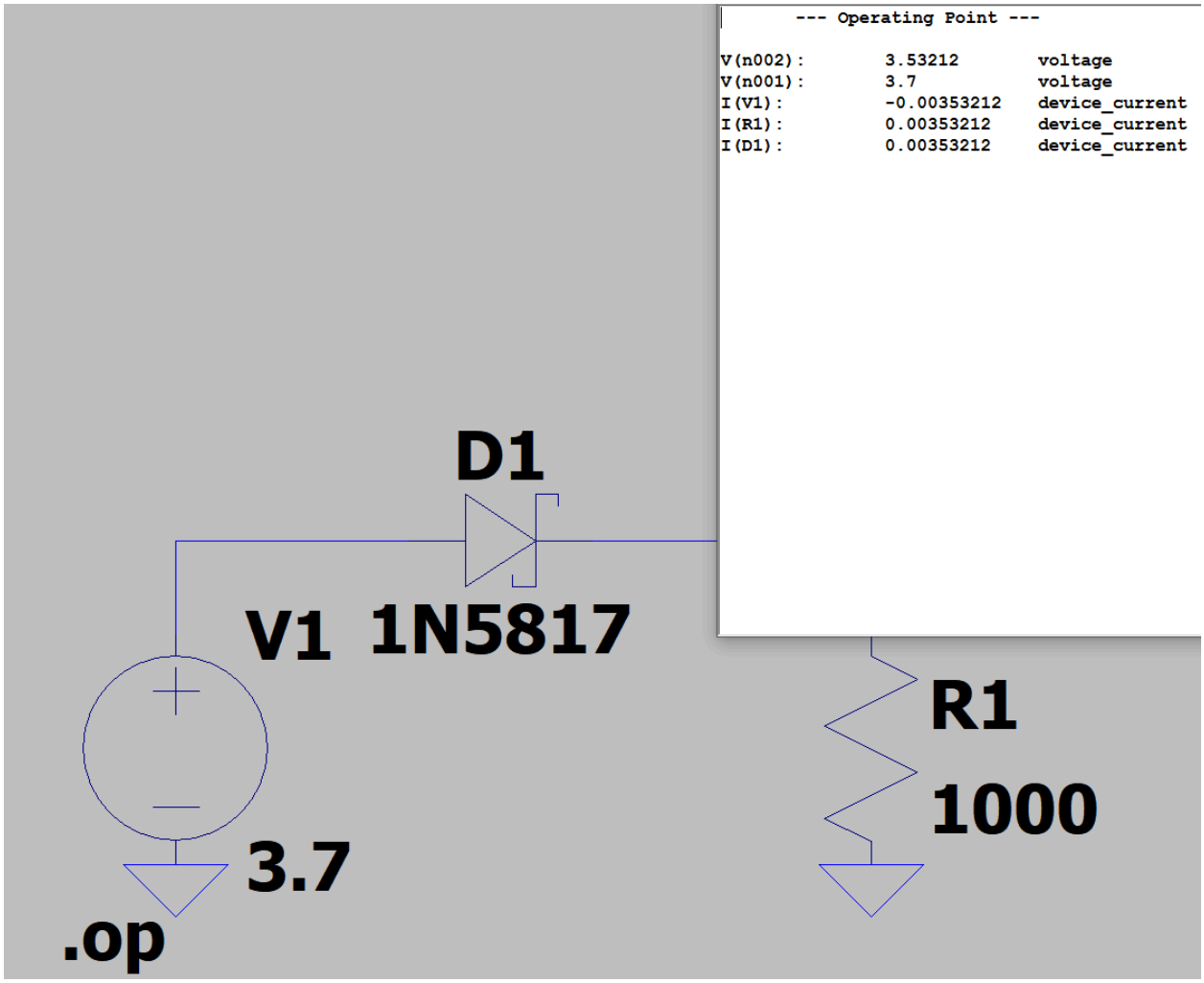

Figure 2: Schottky Diode Simulation

2.1.2 Estimating the Remaining Food in the Rotator

A motor's RPM can be affected by the amount of load it is carrying. Due to this, we figured we could estimate how much food was in the feeder due to the difference in mass that the food would have depending on how much was in the container. To do this, we wanted to record and log two different times, t_1 and t_2, explained below.

t_1: *The time it takes the motor to complete a full revolution with no food.*

t_2: *The time it takes the motor to complete a full revolution with max capacity.*

To develop a linear equation to predict the capacity of the feeder, we took these times and created two sets of coordinates, (x_1, y_1) and (x_2, y_2) and plotted them, as shown in *Figure 3*.

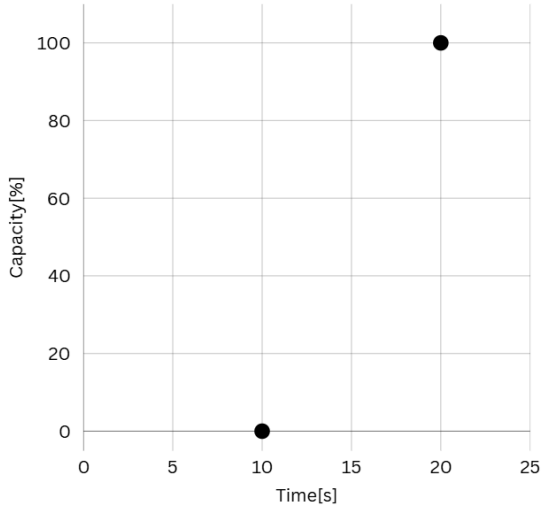

Figure 3: (x_1, y_1) *and* (x_2, y_2)

The x-coordinates were the times values that we logged and the y-coordinates corresponded to the food capacity, 0% for empty and 100% for full. If we form a line fit, the slope of that line is:

$$\frac{1}{t_2 - t_1} \tag{1}$$

If we account for the adjusted x-intercept, we get the equation:

$$f(x) = \frac{x - t_1}{t_2 - t_1} \quad (2)$$

used to calculate the percentage of food remaining with x being the amount of time the motor took to spin fully when feeding. An example of a linear fit of this equation is shown in *Figure 4*.

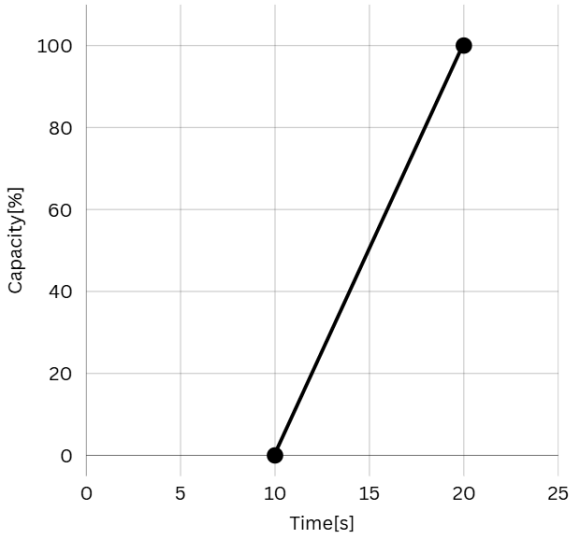

Figure 4: Linear Fit Between Points

2.2 Design Description & Justification

2.2.1 Microcontroller Subsystem

The ESP32 serves as the central controller for the feeder, managing all major components, as shown in *Figure 5* below. Its extensive documentation and wide range of available libraries made development straightforward. Both the camera and motor are controlled via 2SJ652-1E P-channel MOSFETs. The ESP32 toggles these MOSFETs by driving their gates, pulling the gate to 0 V turns the MOSFET on, allowing power to flow to the components; pulling the gate to 3.3 V turns the MOSFET off, cutting power. This setup ensures that the camera and motor only draw current when needed, preventing unnecessary load on the ESP32 and avoiding any potential impact on its processing performance.

In addition to output control, the ESP32 continuously reads analog signals from both a hall effect sensor and a physical feed button mounted on the enclosure. It also manages communication with the mobile app, enabling user interaction and system feedback.

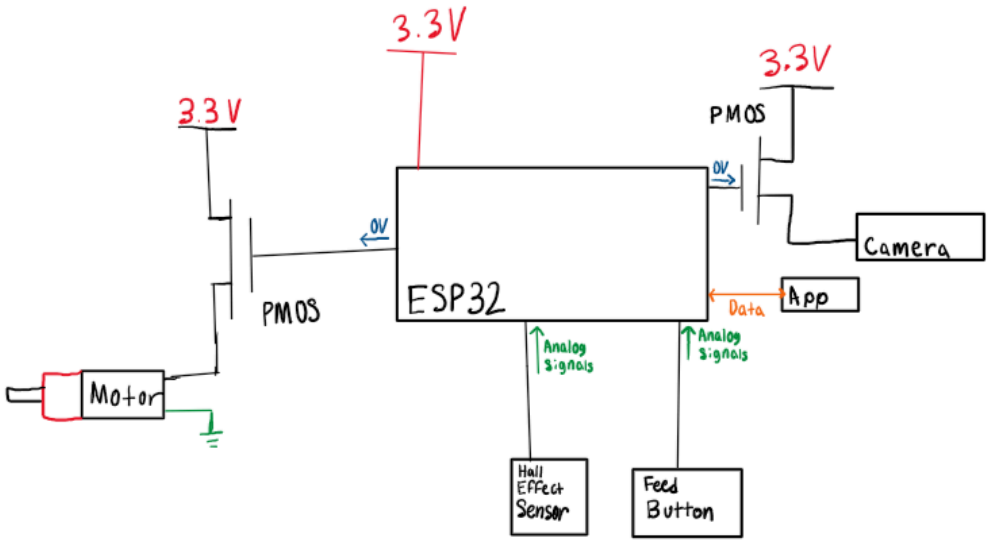

Figure 5: Microcontroller Diagram

2.2.2 Motor and Sensor Subsystem

As mentioned above, the ESP32 reads the analog value from the hall effect sensor and is set at a certain bit value. At no magnetic field, the hall effect sensor outputs half of the input voltage. In this case, that would be 1.15 V. When the magnet gets closer, the magnetic field from the north-facing pole of the magnet facing the hall-effect sensor drives the voltage higher. This is the setting for when the opening of the container is facing straight up. This signal is sent to the ESP32 to determine that the motor needs to stop rotating.

Using the same PMOS transistor as the camera, when the manual feed button is pressed, a logic zero is sent to the gate of the MOSFET to turn on the MOSFET. Once the hall effect sensor reads the magnet, the gate of the MOSFET is driven to logic one. These connections are visualized below in *Figure 6*.

The hall effect sensor was selected due to being an analog value; the motor was selected due to having a low voltage and low RPM in testing; and the transistor was selected because it was readily available at the self-service electronics shop and had a correct turn-on voltage for our application.

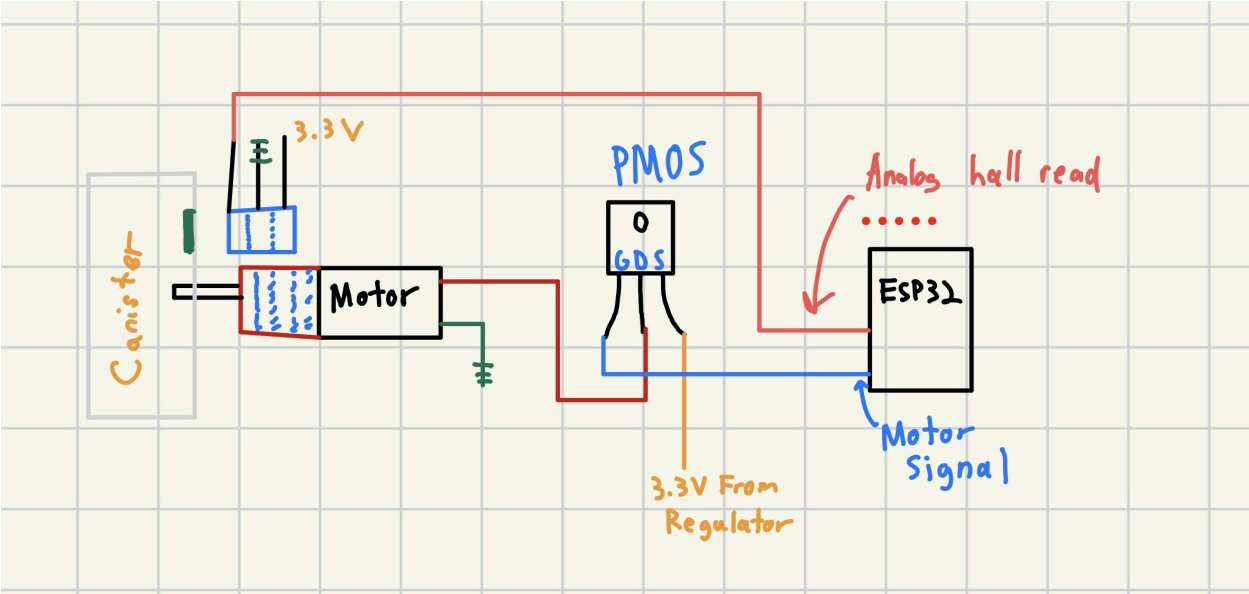

Figure 6: Motor and Sensor Schematic

2.2.3 User Interface Subsystem

The user interface has several components, including the physical feed button, green power LED, and the web app. We were originally going to create a mobile app, however, we opted to instead create a web app using Google's Firebase as there is a lot of documentation about it in relation to using an ESP32 microcontroller. Using Firebase, it is simple to create a mobile app from a web app, so in the future, we can create a mobile app as well. Written in C++, JavaScript, and HTML, the web app allows the user to control the feeder from anywhere by going to *auto-fish.food*. After the user logs in with their email and password, they are presented with a screen that includes the manual feed button, feed times, camera, and food remaining. When the user presses a button on the app, it sends data to the real-time database on Firebase.

The ESP32 then receives this data through a Wi-Fi connection and stores it locally on the chip. This allows the feeder to continue to feed at the scheduled times even when it is not connected to Wi-Fi. As can be seen in *Figure 7* below, when feeding is triggered either through the web app or the physical button, the motor starts and the state is changed accordingly. There is a 1.5 second delay before we check the hall effect sensor output again to allow for the food container to rotate away from the magnet. Then, it continues to check if the magnet is detected, and if it is, then the motor stops.

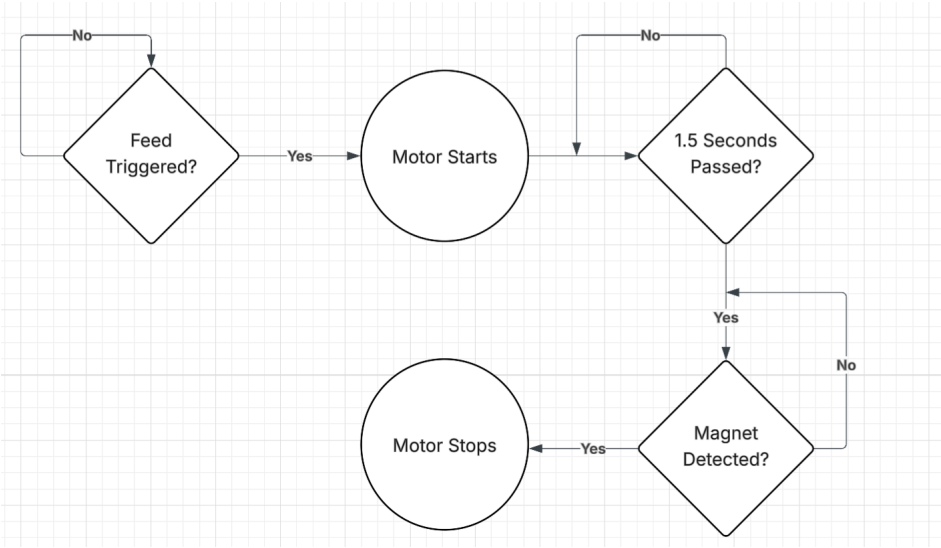

Figure 7: Feeding State Diagram

2.2.4 Power Subsystem

The power subsystem has five major components that need to be decided: how to input voltage from the DC wall adapter, how to charge the battery, what battery, the switching circuit, and the linear regulator. All connectors and switches were gathered from the self-service electronics shop, since those items were readily available.

The linear regulator was the easiest choice out of all of them. The self-service electronics shop had the AP2112K-3.3 and this met our needs for the PCB. The maximum current that the

ESP32 would draw, which is the major power factor, was 240 mA. This regulator produced up to 600 mA, so it worked well for this application. The whole system on the PCB is powered by 3.3 V.

The next part was the battery and charger. The idea was to have the largest battery possible for the enclosure size that we were going to build, which was based on the feeders available on the market. On Digi-Key, a battery was selected that met these measurements, was over 4 Wh, and had a nominal voltage of 3.7 V. This would work well because the linear regulator would produce 3.3 V, as long as there was an input of at least 3.3 V. As far as the battery charger, when looking at batteries, Digi-Key recommended a single-cell battery charging IC shown below in *Figure 8*. As long as the input was 4.3 V or greater, the charging IC would work. This was the choice selected as it was a small package.

Finally, the connection to the DC wall adapter and a switching circuit was needed. Many devices are shifting to USB-C, so a USB-C connector was chosen. The connector only needed to have VBUS and Control Channels since it was only to be used for power. The cheapest, right-angle mount USB-C connector was selected. As for the switching circuit, the simple solution chosen was Schottky diodes. Since the device only needed to switch to battery power (3.7 V) at the loss of USB-C power (5 V), since the potential was higher on the USB-C port, diodes would work well. Schottky diodes were chosen due to their low voltage drop.

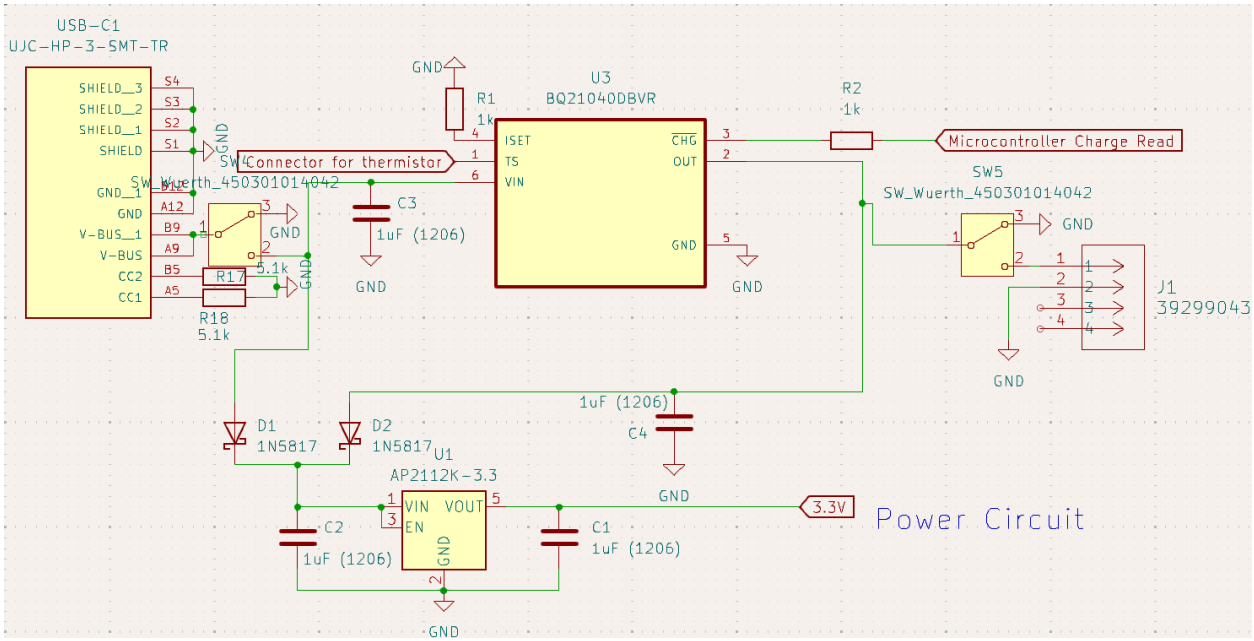

Figure 8: Power Schematic

2.2.5 Camera Subsystem

When the "View Feed" button on the app is selected, the ESP32 is triggered to ground the gate of a 2SJ652-1E PMOS by sending it 0 V. This PMOS has a threshold voltage of 3.3 V, so this will trigger it to allow 3.3 V to the drain which is connected directly to our OV2640. This PMOS was chosen because it was available at the E-shop, and since the threshold voltage is 3.3 V, we could send 3.3V to the gate to close the drain when we do not want to use the camera [2].

When the camera gets powered, it initializes, starts capturing frames, and sends them straight back to the ESP32. The ESP32 will take these frames and send them to a virtual machine set up on Google Cloud. The virtual will create a web address that displays a video that constantly updates with the new frames. Our web app will connect to this address in order to view the feed. Google Cloud was used because it was the easiest option that did not cost money. These interactions with the camera are visualized in *Figure 9*.

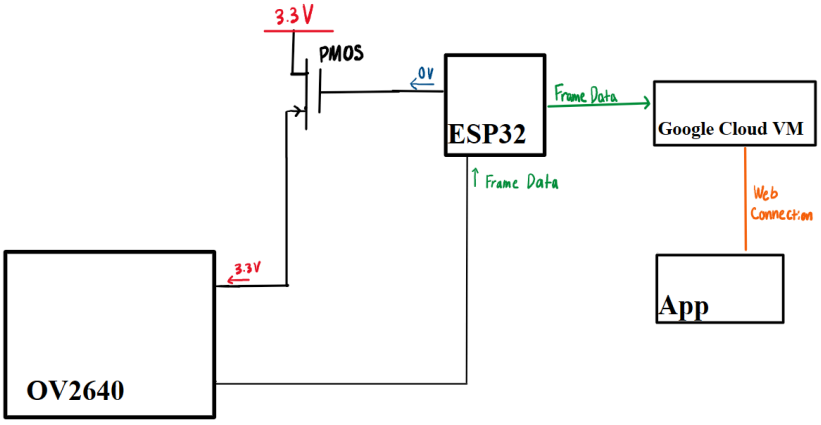

Figure 9: Camera Diagram

2.3 Design Alternatives

2.3.1 Power Subsystem

In the second revision of our PCB design, the USB-C adapter failed to deliver 5 V to the board. Upon probing the connector, we observed that both CC1 and CC2 lines were floating, with only one of the two lines sourcing 5 V, depending on the orientation of the USB-C plug. This indicated that the USB-C source was unable to detect a valid pull-down on the CC lines, preventing 5 V from being delivered. To address this, we added 5.1 kΩ pull-down resistors on both CC1 and CC2. This allowed proper negotiation and enabled the 5 V VBUS line to power the rest of the board.

2.3.2 Camera Subsystem

Originally, we planned to use the OV7670 camera module for this project, and it was successfully integrated into our breadboard demo. However, after assembling our first PCB, we realized that the ESP32 chip used on the board—the ESP32-S3-WROOM-1—differed from the one used during initial prototyping. The ESP32-S3 lacks native support for the OV7670, making integration much more difficult. In contrast, the OV2640 offered better compatibility with the ESP32-S3 and supported higher image quality. As a result, we transitioned to using the OV2640 for the final design [7][8].

3. Conclusion

3.1 Accomplishments

The project was able to be completed successfully with every requirement met. The goal of the project was to create a fish feeder that the user was able to view the camera feed of their fish tank, along with different feeding functions. The user is able to view the camera feed, manually feed, and schedule feeding times for the feeder to activate through the web app. The user is also able to manually feed via the button on the feeder itself. The unit draws power from the USB-C connector, but can also switch to an internal backup battery for short-term use. The battery is also charged on board with a battery charging IC. All design requirements and original functionality that were intended for the feeder were accomplished. The project was a complete success.

3.2 Uncertainties

We were very happy with the results of the project, however, the framerate of the camera was lower than we ideally would want if we were to commercialize the product. This is discussed in the future work section. Other than that, there were no unsatisfactory results to the project.

3.3 Future Work

Although the project delivered satisfying results, we identified several improvements that would be necessary to elevate the SAFF to a commercial-grade product. First, replacing the basic Schottky diodes sourced from the E-Shop with better alternatives would reduce voltage drop. Combined with a larger-capacity battery, this change would significantly extend battery life during power outages.

Also, the ESP32 chip used in this prototype was the ESP32-S3-WROOM-1-N16, which lacks integrated PSRAM. Upgrading to a variant such as the ESP32-S3-WROOM-1-N16R8,

which includes 8MB of PSRAM, would enable more memory-intensive operations, including higher-resolution image capture and improved camera framerates [4].

We selected 2SJ652-1E P-channel MOSFETs due to limited availability in the E-Shop, but N-channel MOSFETs would be a better choice. They offer better efficiency and would allow for simpler logic-level control—eliminating the need to continuously drive 3.3V to the gates to keep the devices off.

While the ESP32 supports Bluetooth, we did not leverage this feature in this project. For a commercial version, enabling Bluetooth would provide an alternative connection method between the feeder and the user's mobile app, improving usability and setup flexibility.

Lastly, this project was designed with the idea of adding optional accessories in the future. In a commercial version, the feeder could include multiple USB-C ports, making it easy for users to plug in additional features as they buy them. To support this, we would need to switch the camera to a USB-C connector so it can be easily connected and recognized by the system.

3.4 Ethical Considerations

One ethical concern with our fish feeder is that it takes away the hands-on interaction between the owner and their fish. When people feed their fish manually, they are not just giving them food; they are also checking in on their health, noticing any changes in behavior, and building a sense of responsibility. This fish feeder removes that connection, allowing oversight when it comes to issues like illness or poor water conditions. Also, feeding fish can be a calming, enjoyable routine for the owner that alleviates feelings like anxiety or stress [5][6]. However, the implementation of the camera in our design will hopefully mitigate some aspects of this concern as the owner will be able to check in on the fish when they are away.

Appendix A: Cost and Schedule

Table 1: Cost of Parts

Description	Manufacturer	Part Number	Quantity	Extended Price
USB-C Port	CUI Devices	UJC-HP-3-SMT-TR	1	$0.75
LiPo Charger	Texas Instruments	BQ21040DBVR	1	$0.52
LiPo Battery	Jauch Quartz	LP503562JU+PCM+2 WIRES 50MM	1	$11.40
PMOS Transistor	Onsemi	2SJ652-1E	2	$4.25
Camera	OmniVision	OV7670	1	$4.50
Linear Regulator	Diodes Incorporated	AP2112K-3.3TRG1	1	$0.56
BJT	Comchip	SS8050-G	2	$0.48
ESP-32	Espressif Systems	ESP32-S3-WROOM-1	1	$5.06
Hall Effect Sensor	SING F LTD	49E	1	$0.59
1kohm Resistor	Stackpole Electronics Inc	RMCF0805JT1K00	5	$0.50
10kohm Resistor	Stackpole Electronics Inc	RMCF0805JG10K0	8	$0.80
100kohm Resistor	Stackpole Electronics Inc	RMCF0805JT100K	2	$0.20
1uF Capacitor	Samsung Electro-Mechanics	CL21B105KBFNNNG	6	$0.60
10uF Capacitor	Murata Electronics	GRM21BR61H106ME43L	1	$0.26
0.1uF Capacitor	Samsung Electro-Mechanics	CL21F104ZAANNNC	1	$0.10
Rare Earth Magnet	TRYMAG	X003716VW9	1	$0.05
Micro Metal Gear Motor	Fielect	GA12-N20	1	$10.35

Push Button	C&K	PTS645SL43-2 LFS	3	$0.66
PCB Bare Connector	Wurth Elektronik	61300811121	2	$0.72
4-pin Connector Female	Molex	0039299043	1	$0.54
4-pin Connector Male	Molex	0039012040	1	$0.27
10 Pin Connector	Amphernol	10056845-110LF	2	$2.62
Misc Wire	N/A	N/A	1	$5
Printer Filament	OVERTURE	850006233403	1	$15.99

Total Cost of Parts: $66.70
Quoted Machine Shop Labor Hours and Cost: 7.5hrs per day 3 days - $1262.70
Estimated Engineering Man Hours: 187 hours @ $100/hour: $18,700

Grand Total: $20,029.40

Table 2: Schedule

Week	Task	Person
2/2/25 - 2/8/25	Research power and hall effect sensors	Colby
	Research app control of ESP32	Brandon
	Talk to Machine Shop about motor and container	Colby
2/9/25 - 2/15/25	Project Proposal (2/13 11:59 PM)	Everyone
	Team Contract (2/14 11:59 PM)	Everyone
	Research hall effect sensors and USB-C Ports	Colby
2/16/25 - 2/22/25	Proposal Review (2/17 10:00 AM)	Everyone
	Work on camera communication with ESP32	Jeremy/Brandon
	Begin designing power delivery systems	Colby
2/23/25 - 3/1/25	Start PCB design and order parts	Colby
	Obtain parts from self-service	Everyone

3/2/25 - 3/8/25	Finalize initial PCB design & design rotator to deliver to PCB shop. Test hall effect sensors and motors.	Colby
	First-Round PCBway Orders (3/3 4:45 PM)	Colby
	Design Document (3/6 11:59 PM)	Everyone
	Get design working on breadboard	Everyone
3/9/25 - 3/15/25	Finish camera communication with ESP32	Jeremy
	Start developing the app	Brandon
	Make revisions to PCB design	Colby
	Second-Round PCBway Orders (3/13 4:45 PM)	Colby
3/16/25 - 3/22/25	Spring Break	Everyone
3/23/25 - 3/29/25	Continue work on the app	Brandon
	Bake/Solder First-Round PCB	Colby
	Bake/Solder Second-Round PCB	Jeremy/Brandon
	Debugging Second-Round PCB	Jeremy
	Make revisions to PCB design	Colby
3/30/25 - 4/5/25	Third-Round PCBway Orders (3/31 4:45 PM)	Colby
	Individual progress reports (4/2 11:59 PM)	Everyone
	Bake/Solder Third-Round PCB	Jeremy/Brandon
	Debugging Third-Round PCB	Jeremy
	Get camera feed to display on app	Jeremy
	Finish app	Brandon
4/6/25 - 4/12/25	Design and finish all enclosures and mounting devices	Colby
	Final Debugging of Third-Round PCB	Jeremy
	Fourth-Round PCBway Orders (4/7 4:45 PM)	Colby
4/13/25 - 4/19/25	Fix existing bugs and assemble	Everyone
	Team Contract Assessment (4/18 11:59 PM)	Everyone
4/20/25 - 4/26/25	Mock Demo (4/21 1:00 PM)	Everyone

4/27/25 - 5/3/25	Final Demo	Everyone
	Work on final presentation	Everyone
	Mock Presentation	Everyone
5/4/25 - 5/7/25	Final Presentation	Everyone
	Work on final paper	Everyone
	Final Papers (5/7 11:59 PM)	Everyone

Appendix B: Requirements & Verification

Table 3: Microcontroller Subsystem

Requirements	**Verification**	**Verification Status (Y/N)**
The ESP32 microcontroller must be able to communicate over I^2C serial protocols with the camera.	• Connect an oscilloscope that reads I2C signals in between one of the data pins connecting the OV7670 and ESP32. • Ensure that the oscilloscope reading confirms I2C communication. • Repeat for each data pin on the OV7670.	Y
The ESP32 microcontroller must be able to communicate over Wi-Fi to the app.	• Turn the feeder on by flipping the power switch. • Open the app. • Verify the “Wi-Fi Connected” symbol is visible on the app.	Y
The ESP32 microcontroller must be able to communicate through analog signals from the manual feed button and hall effect sensor.	• Connect a voltmeter on the output pin of hall effect sensor • Ensure that the voltage changes when a magnetic field is present • Connect a voltmeter on the output pin of the manual feed button • Ensure that the voltage is 3.3V once the manual feed button is pressed	Y

The ESP32 microcontroller must only send 3.3V +/- 0.1V to the camera when the option is selected on the app.	• Connect a voltmeter to the drain of the transistor connected to the camera. • Ensure that 0V +/- 0.2V is being read on the voltmeter. • Select the camera icon on the app. • Ensure that 3.3V +/- 0.2V is being read on the voltmeter.	Y
The ESP32 microcontroller must only send 3.3V +/- 0.1V to the motor when the manual feed button is selected or when the scheduled time is reached.	• Connect a voltmeter to the drain of the transistor connected to the camera. • Ensure that 0V +/- 0.2V is being read on the voltmeter. • Select the camera option on the app. • Ensure that 3.3V +/- 0.2V is being read on the voltmeter.	Y

Table 4: Motor and Sensor Subsystem

Requirements	**Verification**	**Verification Status (Y/N)**
3.3 V +/- 0.1 V is successfully supplied to the motor from the ESP32 when the feed button is pressed.	• Connect a multimeter on the gate of the transistor. • Press the feed button. • Ensure that 3.3V +/- 0.1V is being read on the voltmeter.	Y
3.3 V +/- 0.1 V is successfully supplied to the motor from the	• Connect a multimeter on the gate of the transistor. • Set a scheduled feed time in the app. • Ensure that 3.3V +/- 0.1V is being read on the voltmeter.	Y

ESP32 when the scheduled feed times are reached.		
The motor successfully completes a minimum rotation of 360-degrees once triggered from the ESP32.	• Mark starting position of container with a piece of tape. • Select the manual feed button and allow to complete the feed cycle. • Measure end point. End point should be the same as the start point.	Y
The motor successfully stops within two seconds once the hall effect sensor is triggered via the magnet inside the rotating cylinder.	• Connect multimeter to hall effect sensor output. • Press the manual feed button. • Once hall effect value raises or lowers from the value at any degree other than 0 degrees, start stopwatch.	Y

Table 5: User Interface Subsystem

Requirements	**Verification**	**Verification Status (Y/N)**
Feeder successfully activates within three seconds of the manual feed button press on the app via Wi-Fi.	• Open the app. • Press the manual feed button on the app. • Ensure that the feeder dispenses the food within three seconds.	Y
The ESP32 microcontroller must only send 3.3V +/- 0.1V	• Connect a voltmeter to the drain of the transistor connected to the camera. • Ensure that 0V +/- 0.2V is being read on the voltmeter. • Select the camera icon on the app.	Y

to the camera when the option is selected on the app.	• Ensure that 3.3V +/- 0.2V is being read on the voltmeter.	
Feeder successfully activates within three seconds of the scheduled time set using the app.	• Open the app. • Choose the Schedule Feeding Time icon. • Schedule a new feeding time. • Ensure that the feeder dispenses within three seconds of the new feeding time.	Y
Feeder successfully activates within three seconds of the manual feed button being pressed on the feeder.	• Press the manual feed button on the feeder. • Ensure that the feeder dispenses food within three seconds.	Y

Table 6: Power Subsystem

Requirements	**Verification**	**Verification Status (Y/N)**
5V drawn from the wall is converted to 3.3V +/- 0.1V after going through the linear regulator	• Connect a multimeter to the output of the linear regulator. • Unplug the battery and plug in wall adapter. • Ensure that 3.3V +/- 0.1V is being read on the voltmeter.	Y
When wall power is unavailable, the feeder successfully switches to battery power.	• Connect a multimeter to the output of the linear regulator. • Unplug the wall adapter and plug in a fully charged battery. • Ensure that 3.3V +/- 0.1V is being read on the voltmeter.	Y

The power subsystem supplies a stable voltage of 3.3 V +/- 0.1 V and 400 mA to the system, even with the temporary loss of DC power.	• Connect a multimeter to the output of the linear regulator. • Have both the wall adapter and battery sockets plugged in. • Simultaneously unplug and plug in wall adapter. • Ensure that 3.3V +/- 0.1V is being read on the voltmeter and all the motor is still functioning properly.	Y

Table 7: Camera Subsystem

Requirements	**Verification**	**Verification Status (Y/N)**
3.3 V +/- 0.1 V is successfully supplied to the camera module from the ESP32 when the camera icon is selected on the mobile app.	• Connect a voltmeter to the drain of the transistor connected to the camera. • Ensure that 0V +/- 0.2V is being read on the voltmeter. • Select the camera option on the app. • Ensure that 3.3V +/- 0.2V is being read on the voltmeter.	Y
The camera feed is displayed through the mobile app.	• Open the app. • Choose the Camera Icon. • Ensure that the camera feed is properly displayed.	Y

References

[1] “Pololu - Micro Metal Gearmotors,” *www.pololu.com*. https://www.pololu.com/category/60/micro-metal-gearmotors

[2] “2SJ652.” *Mouser.com*, mouser, www.mouser.com/datasheet/2/308/2SJ652-D-255514.pdf.

[3]S. Santos, “Firebase: Control ESP32 GPIOs from Anywhere | Random Nerd Tutorials,” *Random Nerd Tutorials*, Jul. 21, 2022. https://randomnerdtutorials.com/firebase-control-esp32-gpios/

[4] “ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet 2.4 GHz Wi-Fi (802.11 b/g/n) and Bluetooth ® 5 (LE) module Built around ESP32-S3 series of SoCs, Xtensa ® dual-core 32-bit LX7 microprocessor Flash up to 16 MB, PSRAM up to 8 MB 36 GPIOs, rich set of peripherals On-board PCB antenna.” Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[5] H. Clements *et al.*, “The effects of interacting with fish in aquariums on human health and well-being: A systematic review,” *PLOS ONE*, vol. 14, no. 7, p. e0220524, Jul. 2019, doi: https://doi.org/10.1371/journal.pone.0220524.

[6] “PetCoach - Ask a Vet Online for Free, 24/7,” *Petcoach.co*, 2019. https://www.petcoach.co/article/why-overfeeding-fish-is-a-problem-and-how-to-avoid-it/

[7] *Advanced Information Preliminary Datasheet OV2640 Color CMOS UXGA (2.0 MegaPixel) CAMERACHIP TM O Mni Ision® with OmniPixel2 TM Technology General Description*. 2006, www.uctronics.com/download/cam_module/OV2640DS.pdf.

[8] “ESP32 Camera Driver.” *GitHub*, 25 Oct. 2022, github.com/espressif/esp32-camera.