ECE 445 Senior Design

Final Report Spring ‘24

Automatic Ice Fishing Rod

Professor: Arne Fliflet
TA: Zicheng Ma
Team 60:
Luke Boelke
James Niewiarowski
Andrew Osepek

May 1, 2024

Abstract

The document details the steps we took to design and build our project throughout the Spring Semester. Our project consisted of developing a functioning automatic ice fishing rod which addresses many of the problems with traditional ice fishing. Our project consists of many unique mechanical, electrical, and software components that elevated the scope and functionality of our automatic ice fishing device. The following document will go into depth examining our high level requirements, block diagram and each of our subsystems and their requirements and verifications.

Table Contents

1. Introduction

1.1 Problem

There are many problems associated with traditional ice fishing that can ruin the fishing experience. Ice fishing can be a very tedious and labor-intensive process. While it is being performed, the fisherman must dedicate all of their attention to the task at hand, constantly jigging the rod, making multitasking impossible. It must be done in a harsh environment as well. The fisherman is exposed to extreme cold temperatures, high winds, and snow which can get uncomfortable after long periods of time. Additionally, there can be long stretches with little to no bites. If the fisherman did not have to constantly attend to the rod, these stretches of no activity would be perfect for taking a break, to warm up, eat a meal, etc., but the nature of ice fishing makes this impossible.

1.2 Solution

Our project aims to create an automated ice fishing rod that eases the challenges associated with traditional ice fishing. The user will have the ability to fish normally as they can spool any lb-test line onto the device as with any lure when fishing. The reel will be automatic, and the fisherman can set the length of the line out from the reel. The fishing rod will have the ability to automatically jig the attached lure in hopes of attracting fish. When a tug occurs at the line, the flex bend sensor mounted on the rod will alert the fisherman through the mobile application. The mobile application will allow the user to control the automatic fishing rod by being able to set preferences to the depth of the line and jigging frequencies. Additionally, the fisherman will be able to record their catches in the mobile application that will be stored in a MySQL database hosted in Google's cloud services.

1.3 High-Level Requirements

The following requirements were developed to hold our project's functionality to the highest standard:

1. The user will be able to set up to 3 different jigging frequencies, a lure depth of up to 50 feet (+/- 5 feet) in increments of 1 foot (+/- 0.25 feet).
2. When a bend angle of 30 (+/- 10) degrees is detected, the jigging will halt within 5 seconds of detection, a notification will be sent to the user application, and the line will be reeled in automatically to its zero position.
3. The user will be able to record their catches in the user application with 7 different data fields. Previous catch information can be viewed in the application.

2. Design

This section of our document expands on the design decisions of our automatic ice fishing rod project. It will go into the specifics of our mechanical design while also examining in detail each of our subsystems.

2.1 Initial Design Introduction

When initially designing the appearance and functionality of our automatic ice fishing rod, we envisioned a device similar to the one drawn in Figure 1. In the end, we created an automatic ice fishing rod that resembled our initial design sketches which is shown in Figure 10. We worked closely with the ECE machine shop to bring our initial sketch to a reality. We used an existing spinning reel rod holder for supporting our fishing rod. The fishing rod holder has a loose and flexible joint that allows our rod to move up and down to support our jigging feature. We cut down a normal length fishing rod which was about 6.5 feet to about 3 feet to create an ideal rod length for our ice fishing rod. The machine shop fabricated our shortened rod into our rod holder using aluminum and steel. To create the automatic reeling feature for our device, we mounted a stepper motor directly to the rod holder using brackets. The shaft of the stepper motor was placed in the hex drive position of the reel's hand crank using a coupler and a hex shaft. Finally, to create the jigging functionality of our device, we mounted a linear actuator between our plywood baseboard and the tail end of our rod. Figure 1 further demonstrates the individual design decisions well.

2.2 Design

2.2.1 Design procedure

1 Sensor subsystem

In order to detect whether or not a fish was on the line, we needed a sensor that could measure the amount of bend occurring in the rod. One of the possibilities we considered was using an IMU placed at the tip of the rod to track the tip's motion. Because we wanted our rod to jig, however, this would increase the complexity of the sensor subsystem, as the jigging itself would cause the tip of the rod to move. This would be detected by the IMU and falsely report a fish on the line. To prevent this, we would need an additional IMU to act as a reference point, which would make the subsystem unnecessarily complex. Instead, we decided to use a flex bend sensor for this subsystem. By placing this sensor along the rod, we would be able to accurately calculate the bend angle and determine whether or not a fish was on the line. This simplified the design of this subsystem greatly while maintaining accuracy.
Our original plan for the flex bend sensor was to use a voltage divider rule circuit to calculate the resistance of the flex bend sensor, consisting of the flex bend sensor and a 10kOhm resistor. The output of this system would be between these two resistors, and will be fed into the ESP32, which would then use voltage divider rule to calculate the resistance of the flex bend sensor and the corresponding bend angle. After looking at the datasheet for the flex bend sensor, however, we saw that it recommended using an op-amp in the voltage divider circuit, in order to act as an impedance buffer, preventing current from flowing out from V_{out} and into the ESP32, which will result in errors in our voltage divider calculation. In order to make our calculation of bend angle more accurate, we added this op-amp to our design of the sensor subsystem [10].

2 Application subsystem

When designing the application subsystem, we wanted to create a mobile application that was intuitively easy and clean to use while also providing the backend management to allow the user to save their catch data. We selected Flutter as our development framework for the mobile application because Flutter allows us to use a single codebase and deploy it on multiple different platforms such as IOS, Android, Windows, and Mac. Flutter's central class hierarchy called widgets streamlined our UI/UX development process by allowing us to design the layout and behind the scenes logic in the same place. Another reason we chose Flutter was for its strong package ecosystem that enabled us to build critical functionality directly in the app. In particular, the Bluetooth software we wrote on the application side was built on the flutter_blue_plus package. To develop the backend management system of our Application, a database and server were needed. We opted to use a MySQL database that would be hosted in Google Cloud Platform Services. The MySQL database is a relational database which efficiently organizes and relates data, therefore making the database ideal for our application. To develop our server which managed HTTP requests from our mobile application to our database, we elected to use Node.js along with Express.js web application framework for building RESTful APIs and user sessions. The framework abstracts away much of the complexity involved in handling HTTP requests and responses making it easier for us to focus on the mobile application (frontend).

3 Control subsystem

For the control subsystem, we needed a microcontroller to both send instructions to as well as receive and process data from the other subsystems. This subsystem would control the overall functionality of our device. We originally planned on using an STM32 microcontroller. We chose this because we noticed that it had both Bluetooth and WiFi capabilities, and we had not yet decided which we were planning to use to communicate with our application subsystem. However, while building the schematic using the STM32, we realized how many additional capacitors it required in order to solder it to the board. Because no one in our group had past experience soldering (with the exception of the soldering assignment), we wanted to keep the soldering complexity to a minimum. After discussing with the TAs and learning that the ESP32 microcontroller also had both Bluetooth and WiFi capabilities, we looked at the example board on the ECE445 Wiki and compared it to that of the STM32. The ESP32's schematic was much simpler, so we decided to use this in our control subsystem instead [9].

4 Mechanical subsystem

Our Mechanical subsystem consists of two main features: automatic reeling and automatic jigging. The automatic reeling feature is driven by a 3.3 V 1.8 A stepper motor along with a dual H-bridge motor driver by Texas Instruments, TI DRV8833. Initially, we planned to use a DC motor to drive the automatic reeling of our device, but we faced challenges of achieving precise depth setting and control. Consequently, we opted for a stepper motor, because it offered great precision and control as it can be controlled in a stepwise manner which makes it ideal for our fishing device. They also offer high levels of holding torque, which can withstand the stresses that will be placed on our ice fishing reel. Secondly, our

automatic jiggins is driven by a 12 V linear actuator and a H-bridge motor driver. Since our actuator is mounted at the butt end of our fishing rod, we chose a small stroke length of 0.8 inches to create our human-like jigging.

5 Power subsystem

The design procedure for the power subsystem began by looking at all the datasheets for the other components of our design, such as the motor [4] and actuator drivers [3] and ESP32 [9], and determining what voltage values they needed to function properly. Once we identified that we needed 3.3V, 5V, and 12V voltage supply lines, we began looking for voltage regulators to use to drop our 12V battery to 3.3V and 5V. One of the possibilities we considered was using linear regulators, due to their simple design. However, we ended up deciding to use AP63203 and AP63205 buck converters, as they are much more power efficient than linear regulators and have greater stability in the output voltage. After choosing these components, creating the schematic for the subsystem was relatively simple, as the datasheets provided diagrams of how to hook up the buck converters [7].
We updated our original design (which had only a single AP63203) to have two 3.3V voltage lines that use AP63203 buck converters, one 5V voltage line that uses a AP63205 buck converter, and a 12V voltage line that is connected directly to the positive terminal of the 12V battery. One 3.3V supply line goes to the ESP32 and another goes to the stepper motor driver. If these components had shared a voltage supply line, as was the case in our original design, whenever the stepper motor engaged, the stepper motor driver would draw a lot of current, causing the current to the ESP32 to drop significantly and possibly causing the microcontroller to reset, which would prevent our system from functioning properly. For our power supply, we originally planned on using a 12V lithium-ion battery. However, we then learned that lithium-ion batteries do not perform well in cold temperatures, making it unsuitable for an ice fishing device. We then switched our design to use a lead-acid battery, which would perform better in a cold environment like a frozen lake.

2.2.2 Design details

1 Sensor subsystem

Once we had decided to use a flex bend sensor for our sensor subsystem, we began looking into how to hook it up to the ESP32 in order to calculate the resistance value, and then use this resistance to calculate the corresponding bend angle. The datasheet provided a lot of useful insights into this. In order to determine the resistance, it suggested using a simple voltage divider circuit [10]. The ESP32 is capable of performing analogRead operations from its GPIO ports, allowing it to read voltage values [9]. I decided to use one of the 3.3V supply lines as the high voltage for the voltage divider circuit. We could connect the voltage value between the sensor and known resistance to one of the ESP32's GPIO ports in order to read this voltage value, which could then be used to calculate the corresponding bend angle, as described below.

Variables:

- v: voltage drop across sensor
- y: voltage read by the ADC port
- X: sensor resistance
- R: known resistance value
- Xi: sensor resistance when straight
- Xf: sensor resistance when bent at a 180 degree angle
- a: bend angle of sensor

Calculating voltage across the sensor:

$$v = 3.3 - y$$

Using voltage divider rule:

$$v = 3.3 * X/(X + R)$$

Isolating X to solve for the resistance of the sensor:

$$X = v * R/(3.3 - v)$$

Using this calculated sensor resistance, we can estimate the bend angle with the knowledge (as mentioned in the datasheet) that there is a relatively linear relationship between the resistance value and the bend angle:

$$a = 180 * (X - X_i)/(X_f - X_i)$$

The above equation will allow our microcontroller to calculate the bend angle of the sensor using the voltage value output from the sensor subsystem. If this bend angle is determined to be above 30 degrees, a fish is on the line, and the control subsystem will communicate with the other subsystems to perform the necessary actions for this event.
The schematic for our sensor subsystem can be seen in Figure 2.

2 Application subsystem

a Database

Figure 3 illustrates our database design. A one-to-many relationship exists between the User and Fish table which means a user can have zero to multiple related catches in the database, but a fish can have only one related user [5]. Each attribute in the User table, which is shown in Figure 4, cannot be null. The primary key for the table is defined by the userID, and the username attribute must be unique. The Fish table in our database, which is described in Figure 5, is used to describe the fisherman's catches. The fisherman is not required to add values to all fields in the table to upload a catch. The primary key for the table is defined by the fishID. The userID is a foreign key that references a user in the User table.

If a user is deleted from the User table, all of their corresponding catch logs are removed from the Fish table.

b Server

Our server is responsible for handling HTTP requests and responses made from our mobile application to our MySQL database. It supports many create, read, update, and delete (CRUD) operations. Figure 12 shows the pseudocode that our server handles for each HTTP route. Initially, the user is asked to login or sign up for an account. At login, the user's credentials are sent to the server via HTTP, the server receives the information and queries the User table for the following user, it returns on success and creates a new user session. At signup, the user's new account information is delivered to the server, the server asks to insert the new user information into the database if the new username does not already exist, it returns on success and creates a new user session. Additionally, we have account delete, application logout, and account update operations that are handled by our backend server, see Figure 12 for more details. With respect to catch information, the user can insert new catch data, it is sent to the server, and the server inserts the catch data for that user in the database, it then returns on success. Additionally, the user can view catch information; a request is made to the server to query the user's catch data and then the data is sent back to the user. Examine Figure 12 to see and expand on all HTTP routes are server handles. Figure 13 demonstrates example requests made to our server.

c Mobile Application

The mobile application consists of multiple pages that streamlines the user's experience within the mobile application. On startup of the application, the user is prompted to login or sign up, see Appendix E. On the Sign Up page, the user is prompted to create an account by filling out a form [6]. The user is brought to the Home page when they sign up successfully. On the Login page, the user is prompted to enter their username and password into a form [6]. If an account exists with their entered credentials, the user is brought to the Home page. Otherwise, the user is asked to re-enter their username and password.

The Home page is composed of multiple pages that allow the user to adjust the mechanical settings of the ice fishing rod, upload catch data, view catch data, see fish alerts, and modify/delete their user account.

There is also a Bluetooth tab that turns on the bluetooth adapter for the iPhone. Switching to this tab initializes a scanner that starts looking for nearby devices. If the rod is powered it should appear on the list as RoboFisher. Tapping on this item connects you to the device and discovers the Bluetooth service broadcasted from the ESP 32 BLE server. After you connect the app creates a subscription to the Bluetooth characteristic that we set up to hold the fishOnHook state. Whenever the ESP32 notifies the subscriber (the app) the app updates its reading of the characteristic. This is similar to an interrupt where the ESP32 notifies the app when new data is ready. From the controls page, sending a jigging frequency and depth will simply write to an alternate characteristic on the BLE server that is different from the fishOnHook characteristic.

3 Control subsystem

Our control subsystem, the schematic of which can be seen in Figure 11, is responsible for controlling the overall operation of our device. The ESP32 executes instructions in order to engage the motor and actuator, calculate the bend angle of the sensor, and communicate with the user application via Bluetooth. In order to simplify the code in the microcontroller's loop function, we created two helper functions: sense and moveDepth.

The sense function reads the voltage output from the sensor subsystem, uses it to find the current resistance value of the sensor, and then uses that resistance to calculate the corresponding bend angle. If the bend angle is above 30 degrees, the function returns true. Otherwise, it returns false.

The moveDepth function is responsible for changing the depth of the line. This function uses Arduino's built-in Stepper class [11], which is designed to control stepper motors. moveDepth takes two arguments, new_depth and curr_depth, which are the new depth to set the line to and the depth the line is currently at, respectively. This function calculates the number of revolutions of the motor needed to change the depth from curr_depth to new_depth, and then uses the Stepper class's step function to rotate the motor that many revolutions [12].

The overall setup of our code's main loop depends on whether the jigging frequency is currently set to 0, meaning whether or not the jigging is off. If the jigging is off, the loop will first check if the user has changed the depth setting in the user application. If they have, the moveDepth function will be used to change the current depth to the newly set depth from the application. Whether or not the line depth has been changed, the code will then call the sense function to poll the sensor. If it returns true, meaning a fish is on the line, the moveDepth function will be used to reel the line in all the way. The loop will then begin again.

If the jigging is not off, the ESP32 will perform a digitalWrite to the actuator driver's inputs to extend the actuator. While the actuator is extending, the sense function will continue to be called to poll the sensor. If sense returns true, jigging will halt and the line will be reeled in all the way. If a fish is not detected during extension, the ESP32 will signal the actuator driver to retract the actuator. Similar to extension, the sense function will continue to be called while retracting. If it returns true, jigging will halt and the line will be reeled in. If a fish is not detected during retraction, the ESP32 will then wait a period corresponding to the jig frequency (10, 20, or 30 seconds), before repeating the loop and jigging again. During this waiting period, the sense function will be repeatedly called. If a fish is detected, the line will be fully reeled in. Pseudocode for the main loop of our function can be found in Figure 14.

4 Mechanical subsystem

Our Mechanical subsystem can be separated into two main components: our stepper motor and linear actuator. Our schematic for our stepper motor and its motor driver is shown in Figure 8 . When determining the maximum stress that would be exerted on our line, we used our reel's rated line. Since our reel has a maximum rated line test of 6 lbs, or 26.689 Newtons, and we can expect a spool radius between 12.7 mm - 19 mm, the expected torque on our reel is expected to be between:

$$\tau = 26.689\ Newtons * (0.0127\ m - 0.0190\ m) = 0.34\ Nm - 0.51\ Nm.$$

We selected a Nema17 stepper motor, because It can provide up to 0.65 Nm of torque and requires 3.3 V. Looking at the schematic in Figure 8, GPIO pins which are labeled STEP 1-4 control the stepper motor's direction and speed [4]. These pins are connected to GPIO pins within our ESP32 microcontroller. Depending on the values for STEP 1-4, in Figure 8, the motor's direction or speed will change. Unlike most motor drivers, the DRV8833 has one power input (VM) that supplies its internal chip logic and the stepper motor. Additionally, AISEN and BISEN are current limiting pins. The current limiting rule is:

$$LimitCurrent\ (amps)\ =\ 0.2\ V\ /\ RSEN\ \ [4]$$

Our stepper motor is rated at a maximum of 1.8 amps, so we selected RSEN = 0.11 ohms to protect our stepper motor from damage. Since our stepper motor should be expected to stall, this feature should be enabled as it protects our motor from drawing a significant amount of current. Our schematic for our linear actuator is shown inFigure 9 , GPIO pins labeled ACT 1-2 control the direction of the linear actuator. These pins are connected directly to GPIO pins on the ESP32 microcontroller. Depending on their values, the actuator will extend, retract, or stop. The L293D motor driver supports two voltage supplies. Port VCC1 is the power supply for the internal logic of the motor driver while VCC2 is the voltage supply for the linear actuator [3].

5 Power subsystem

The schematic for the power subsystem is shown in Figure 7. As seen here, the subsystem consists of three buck converter circuits, two of which are AP63203 buck converters to step the 12V battery down to 3.3V, and one of which is an AP63205 buck converter to step the battery down to 5V. One of the 3.3V voltage lines will be connected to the control subsystem's ESP32 and the high voltage in the sensor subsystem's voltage divider circuit. The other 3.3V voltage line will be connected to the DRV8833 stepper motor driver. The 5V voltage line will be connected to the L293D actuator driver to supply its internal logic. In addition to these voltage outputs, we will also have a 12V output coming directly from our battery to be delivered to the L293D driver. This value will be used by the driver to power our 12V actuator.

For our power supply, we use a 12V, 10Ah lead-acid battery. We wanted to ensure that our battery would be able to power our device for at least 5 hours of fishing before needing to be charged. After connecting our circuit to a 12V power supply in the lab room, we noticed that even while the motor was engaged, the current being drawn from the supply never surpassed 1.5A. Using this value to calculate the amount of hours a 10Ah battery would be able to power our device (by dividing Amp-hours by Amps), we see that our battery would operate for

$$t\ =\ (10Ah)/(1.5A)\ =\ 6.67\ hours$$

6.67 hours of operation will provide the user with plenty of time for an ice-fishing session.

2.3 Verification

2.3.1 Sensor subsystem

In order for our device to function properly, the sensor subsystem would have to accurately detect the presence of fish. This means that the calculated bend angle using the voltage output of our sensor subsystem must accurately reflect the sensor's actual bend angle (within our tolerance of 10 degrees.) In order to ensure that this was the case, we programmed the ESP32 to repeatedly poll the sensor subsystem's output voltage, calculate the bend angle, and then print this value. While the sensor was near straight, the calculated angle was around 3 degrees, whereas when the sensor was bent, the printed angle increased to over 30 degrees. Additionally, our sensor had to be able to detect when the bend angle exceeded 30 degrees, so that the microcontroller could then instruct the other subsystems to perform the required actions for a fish on the hook (halting jigging, reeling in the line, etc.) We were able to test this once our device was fully built and programmed. When we bend the rod so that the sensor angle is greater than 30 degrees, the jigging halts, the line is fully reeled in, and the user application is notified. This meant that our sensor subsystem was able to meet all its requirements. The R&V table for our sensor subsystem can be seen in Table 1.

2.3.2 Application subsystem

To verify the success of our Application Subsystem, we broke our testing up into testing of the database, server, and mobile application. To verify the success of our MySQL GCP database, we queried the Fish and User tables from a python file, local to our computer, printing the queries to the console, verifying the printed rows matched the data in the database. Likewise, we inserted data into the Fish and User tables and checked the database for newly inserted values. To verify the success of our server, we simulated simple HTTP POST/GET requests from our user application to our server and printed out the HTTP message bodies on both sides to verify its success. We also initiated two user sessions in different mobile applications and verified that the users' session ID and data was kept active and private during the user's login by printing out the user's session data on Server and mobile application. Finally, to verify the success of our mobile application, we demonstrated its functionality by logging in, signing up, adding catch data, viewing catch data, and modifying the user's account details. We also shared our mobile application with peers and friends to gain feedback on its UI/UX. The R&V table for our application subsystem can be seen in Table 2.

2.3.3 Control subsystem

In order to test the functionality of our control subsystem's code, we needed to test its interactions with each of the other subsystems. To ensure that the ESP32 could communicate with the application subsystem via BLE, we would program the microcontroller to read data sent from the application and print the value to the serial monitor. If the value received matched the value sent, this meant that the test was successful. We then wanted to make sure these received values could be communicated to the mechanical subsystem to set the jigging frequency and reel in/out the line. To test this, we edited our

microcontroller code to change the line depth and jig frequency to the values received from the application subsystem. Once these values were sent to the ESP32, we ensured that the line depth and jig frequency were updated accordingly. Once our control subsystem had passed this test, we then wanted to test that the ESP32 could send a notification to the application subsystem. To verify this, we bent the sensor to an angle of greater than 30 degrees, and ensured that the application subsystem displayed "fish present on hook." When we straightened the sensor, we ensured that it displayed "no fish present." We then wanted to ensure our control subsystem's code could properly carry out a typical fishing session, involving all other subsystems. To do this, we used the application subsystem to set the depth and frequency, and ensured that the mechanical subsystem carried out these actions. We then changed these settings and ensured that they were updated accordingly. Once this was successful, we bent the sensor to signal a fish on the line to ensure the ESP32 sent the proper instructions to the other subsystems. We ensured that the jigging was halted, the line was reeled all the way in, and the application subsystem displayed "fish present on hook." Because this test had passed, it meant our control subsystem was functioning properly. The R&V table for our control subsystem can be seen in Table 5 .

2.3.4 Mechanical subsystem

To verify the stepper motor and its motor driver's functionality, we programmed the ESP32 using Arduino's Stepper.h library and connected our stepper motor to our microcontroller. We programmed it to perform a series of simple movements, such as rotating clockwise and counterclockwise, at various speeds and step increments. We would verify the amount of line let out/in by our reel using a tape measure to see if our reel was functioning properly. Then, we integrated the sensor subsystem with our stepper motor to verify the automatic reeling-in feature of our device. When a large bend angle occurred on the sensor, we verified the line was reeled back to its original zero position. Additionally, we placed an object of 1lb on the end of our line to verify that our rod and reel could support the stress exerted by a fish on our device. Finally, we programmed our ESP32 to verify the success of our linear actuator for our automatic jigging functionality. We wrote simple code in the microcontroller to control the extending and retracting of the linear actuator at different frequencies. This code enabled us to send commands to the actuator to extend and retract as required for the jigging component's functionality. We integrated the sensor subsystem with our linear actuator to verify the automatic halting feature of our device when a fish is detected. When a large stress and bend was placed on our sensor, the linear actuator would halt in its current position. The R&V table for our mechanical subsystem can be seen in Table 4.

2.3.5 Power subsystem

To ensure that our power subsystem was delivering the proper voltage values to the other components of our design, we used a voltmeter to measure the voltage output of the buck converter circuits. The voltage being delivered to the ESP32 and sensor was 3.374V, the voltage delivered to the stepper motor driver was 3.324V, and the voltage delivered to the actuator driver was 4.994V. All our measured values were within 0.1V of their desired values (3.3V, 3.3V, and 5V), meaning that our power subsystem passed

the verification tests. A more detailed description can be seen in the R&V table for the power subsystem, shown in Table 3 .

2.4 Costs

We spent approximately $166.16 in parts to build the necessary functionality and success for our automatic ice fishing rod. Table 6 shows an itemized list of all our costs when building our project.

We spent on average 12 hours/week working on developing our automatic ice fishing rod during our project's 9 week timeline. Therefore, we spent about 108 hours in total per person over the course of our project. A comparable industry salary for this type of position would expect $45/hour.

Each team member will cost:

$$\$45/hour\ x\ 2.5\ x\ 108\ hours\ =\ \$12,150$$

The total labor costs, consisting of three people, will cost $36,450 using the value above.

Thirdly, when estimating the machine shop's cost, we estimate it to be $475.
We calculated the expected machine shop labor cost by using:

$$\$45/hour\ x\ 10\ hours\ =\ \$450$$

Table 6 lists the itemized expected costs for their parts and labor.

In total, we expect our automatic ice fishing rod project to cost $37,091.16.

2.5 Conclusions

2.5.1 Accomplishments

We were able to get our automatic ice fishing rod fully functional by the end of the semester. Our project is able to fulfill all three high-level requirements we set when initially proposing the idea. Using the settings in the user application, the fisherman is able to set the jigging function to 3 different frequencies: 1 jig every 10 seconds, 1 jig every 20 seconds, and 1 jig every 30 seconds. The fisherman can also use the application to set the line depth in increments of 1 foot, up to a maximum depth of 50 feet. The sensor subsystem is able to accurately reflect the bend angle of the flex bend sensor and signal whether or not a fish is on the line, using a threshold angle of 30 degrees. When this threshold is exceeded, the rod stops jigging (even if the actuator is in the middle of extending or retracting), the line is fully reeled in to a depth of 0 feet, and the user application will display "fish present on hook" rather than "no fish present," allowing the fisherman to be notified that there is a fish on the line. Within the user application, the fisherman is able to store information regarding catches in a MySQL database. These stored entries can then be viewed within the application for future reference, and can also be

deleted from the database. If deleted, they will no longer be shown in the application. Because our project was able to satisfy all high-level requirements, it is able to solve many of the problems associated with ice fishing.

2.5.2 Challenges and uncertainties

Though we saw many successes with our project this semester, there are still some uncertainties that we encountered or that still remain. We ran into various issues while designing our PCB, as it was the first time any of us had used KiCad, with the exception of the soldering assignment at the beginning of the semester. We were able to meet the first PCB order deadline, but in order to do so, we needed to rush through our design. Because of this, we ran into various issues when we began soldering, such as pins not being grounded correctly and incorrect footprints. In order to prevent this issue from occurring again, we wanted to make sure we got our project fully functioning on the breadboard before we began designing another PCB, and ensure all our connections and components were correct. However, after dealing with many setbacks with things like the motor driver, we were not able to get our project functioning on the breadboard until the end of the semester. We were able to fix our original PCB design and meet the last order deadline, but by the time it arrived, we were not left with enough time to move our project from the breadboard to the PCB, as we were still making minor bug fixes.

Another challenge we encountered this semester was burning out our stepper motor drivers. We would connect them to the microcontroller and the stepper motor and then program the ESP32 to rotate the motor. This would work for a brief period of time before the motor would simply stop moving. We used a voltmeter to ensure that the ESP32 was delivering the correct voltages to the driver, thinking it may be an issue with our code. However, we noticed that the driver was receiving the proper input voltages, but all its output voltages were 0V. This meant that it was an issue with the motor driver itself. After reviewing the datasheet, we figured out that there was too much current running through the driver, causing it to burn out. We were able to fix this by adding current limiting resistors, preventing the driver from delivering too much current to the stepper motor. Though we were able to fix this issue, because the current to the stepper motor was decreased, the motor cannot produce as much torque, and so will not be able to reel in large amounts of weight.

A final challenge that we encountered this semester dealt with the bluetooth communication between the Flutter application and the ESP32. In order to test the bluetooth communication, we needed to upload the Flutter application to a mobile device, rather than simply using an iPhone or Android emulator. However, to upload it to an iPhone without publishing it to the app store, we needed a Mac. James is the only member of our group who has a Mac, meaning that he was the only one able to upload our Flutter app to his iPhone. Because of this, any time we wanted to test the bluetooth communication, we needed to meet together as a whole group. This was not a major issue as we were already meeting frequently, but if something came up and we needed an impromptu meeting to test our device, we would need to work around everyone's school schedules to find a time to meet.

2.5.3 Ethical considerations

When developing an automatic ice fishing rod, it's essential to consider various ethical and safety issues, both during the development process and in terms of potential misuse. The IEEE and ACM (Association for Computing Machinery) Codes of Ethics provide general guidelines that can be applied to our project. The issues listed below ensure our project upholds to the highest standards established:

1 Privacy Concerns (ACM Code 1.6) [1]

Any time personal data is being collected and stored, there is a risk of a breach of privacy. Our application will allow users to upload data to a GCP database, which will contain two tables: one for personal information and one for catch information. The main privacy concern deals with the former, as this table will contain the user's first and last name. In the application's privacy policies, we will clearly state that the information they are submitting will be stored in a database, and ask that they only proceed if they give us consent to store their data. In the event that this database is hacked, this information could be used for malicious purposes that could potentially harm the user. In order to minimize these consequences in the event of an attack, this risk will be clearly stated in the privacy policies, and will inform the user that they may use an alias when submitting this information if they wish to take extra precautions.

2 Transparency and Honesty (ACM Code 1.3) [1]

When developing a new product, it is important to fully document the entire design and implementation processes. Data should not be tampered with and no values should be altered to ensure honesty and transparency. The claimed capabilities of our design must accurately reflect its actual capabilities, so users receive the quality they expect. In order to make sure we are transparent throughout our development, we will thoroughly document the process in our lab notebooks. We will also make our code open sourced so that users can understand our system and how it works. The claimed values will be realistic and accurate, and will be reported with tolerance values (e.g., +/- 5 N) to account for slight differences in conditions, equipment, etc.

3 Mechanical and Electrical Safety (IEEE 1.1) [2]

These codes involve safety issues related to mechanical and electrical failures in the rod or its deployment mechanisms.To mitigate these issues we will employ the following precautions. The battery and PCB will be enclosed in an element-proof box to minimize the risk of electric shock and mechanical/electrical damage to the system. The system components will also be properly grounded. The automatic reeling system will operate at a frequency that is safe for the user as the fisherman's line will not be reeled in excessively fast. We will rigorously test the mechanical stability of the rod to ensure that it can withstand the stress of catching multiple fish of varying sizes.

4 User Training and Guidelines (ACM Code 1.2) [1]

This code relates to safety issues surrounding Injuries due to improper use or lack of understanding of the equipment. In our user manual we will have instructions for using the rod. To mitigate this issue we will provide clear user manuals, safety guidelines, and potentially implement features like emergency stop mechanisms. These instructions will include how to turn on and off the system. There will also be an analog off switch on the power delivery circuit to cut power in cases of extreme malfunction. The user manual will also have safety guidelines such as how far to stand from the rod and advise against putting your hands near the hook. Our user manual will also advise against tampering and have information about how to handle component breakdown such as battery corrosion or leakage.

2.5.4 Future work

To continue and improve upon our automatic ice fishing rod, we would first integrate all our hardware onto the PCB. By moving everything from the breadboard to the PCB, we could make our design much more compact and professional-looking. Because the components would be soldered to the PCB, it would also make our hardware more durable, as the breadboard components and wires can very easily be removed. Once we have integrated everything onto the PCB, we would then make our design look less like a prototype by covering up any exposed wire connections, and creating an enclosure for the PCB and battery that we could mount to our device's stand. This would help protect our project from the conditions it would be exposed to at a frozen lake, such as water and wind, increasing the safety of the product. We would then want to look into alternatives for our stepper motor driver and find one that could handle higher currents. Our stepper motor would have been able to produce a lot of torque if provided enough current. However, because we needed to add current limiting resistors to avoid burning out the motor driver, our stepper motor was unable to reel in heavy weights. By replacing the stepper motor driver, our motor would be able to reel in bigger and heavier fish, even if they were fighting against the fisherman. Once we ensured that our improved device could reel in a greater weight, we would then take the fishing rod out to an actual frozen lake and test it in a real-world setting. We would set the jigging frequency and lower the line into the water using the user app, and if a fish bites the hook, we would make sure our device could successfully detect this fish and reel it in. This would ensure that our project would be able to solve the problems typically associated with ice fishing. We could also improve the application subsystem by placing it within a Docker container image.

References

[1] ACM, “ACM Code of Ethics and Professional Conduct,” *Association for Computing Machinery*, Jun. 22, 2018. https://www.acm.org/code-of-ethics

[2] “IEEE Governing Documents,” www.ieee.org. https://www.ieee.org/about/corporate/governance/index.html.

[3] “L293x Quadruple Half-H Drivers,” Sep. 1986. https://www.ti.com/lit/ds/symlink/l293d.pdf?ts=1710069624078&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FL293D.

[4] “DRV8833 Dual H-Bridge Motor Driver,” Jan. 2011. https://www.ti.com/lit/ds/symlink/drv8833.pdf?ts=1710011492587.

[5] “Relationships in SQL - Complete Guide With Examples,” Devart Blog, Oct. 26, 2021. https://blog.devart.com/types-of-relationships-in-sql-server-database.html.

[6] “Build a form with validation,” docs.flutter.dev. https://docs.flutter.dev/cookbook/forms/validation

[7] Diodes Incorporated. “AP63200/AP63201/AP63203/AP63205 3.8V TO 32V INPUT, 2A LOW IQ SYNCHRONOUS BUCK WITH ENHANCED EMI REDUCTION Datasheet.” Diodes Incorporated, Jan. 2019, https://www.diodes.com/assets/Datasheets/AP63200-AP63201-AP63203-AP63205.pdf.

[8] “flutter_blue_plus 1.32.4," pub.dev. https://pub.dev/packages/flutter_blue_plus#using-ble-in-app-background.

[9] Espressif Systems. “ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet.” Espressif Systems, 2023, https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

[10] Spectra Symbol. “SpectraFlex FLX Datasheet.” Spectra Symbol, 2014, https://cdn.shopify.com/s/files/1/0578/4128/7283/files/SPECTRAFLEX_DATA_SHEET_v1.0.pdf?v=1691015077.

[11] “Stepper.” Stepper - Arduino Reference, Arduino, www.arduino.cc/reference/en/libraries/stepper/.

[12] “ESP32 - Stepper Motor.” ESP32 Tutorial, ESP32IO.com, esp32io.com/tutorials/esp32-stepper-motor.

Appendix A Visual Aids

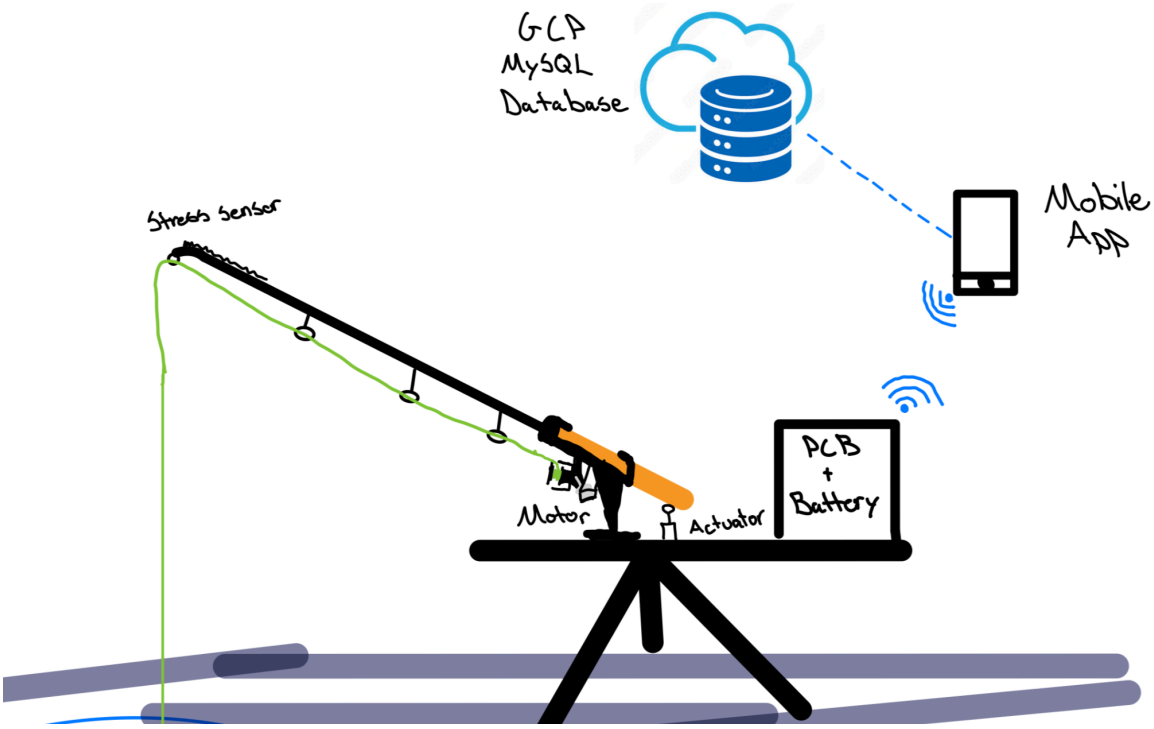


Figure 1: Initial Visual Design of Automatic Ice Fishing Rod

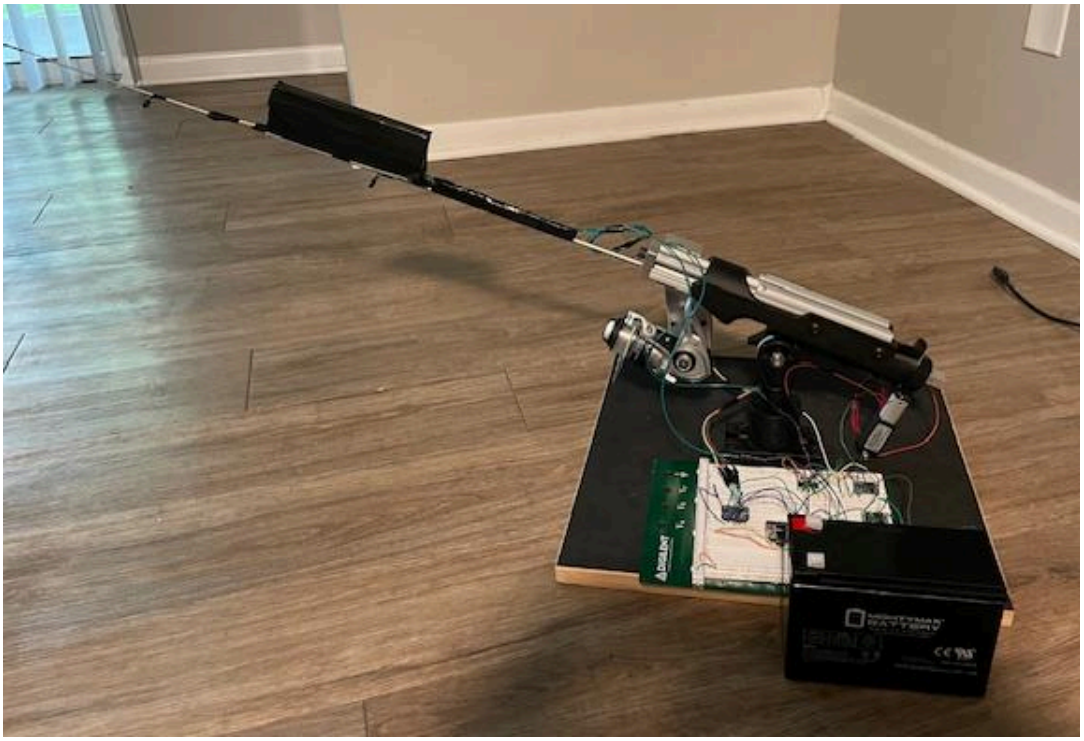


Figure 10: Final Design of Automatic Ice Fishing Rod

Appendix B Schematics

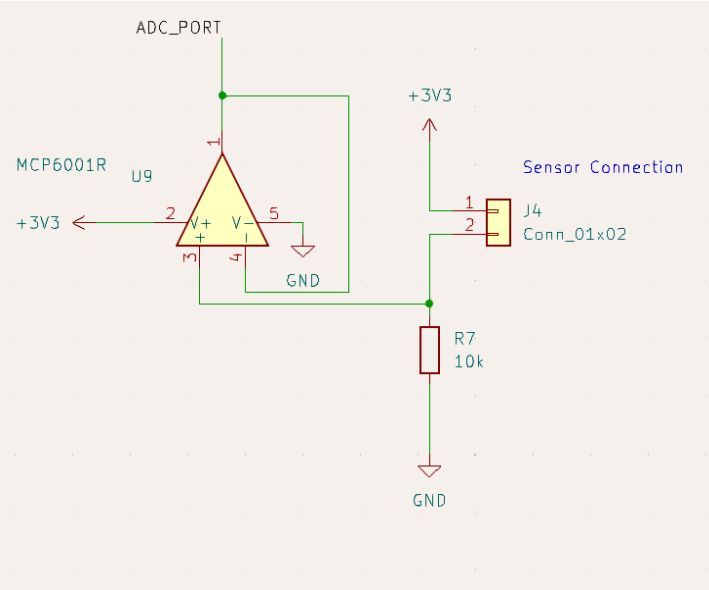


Figure 2: Schematic for Sensor Subsystem

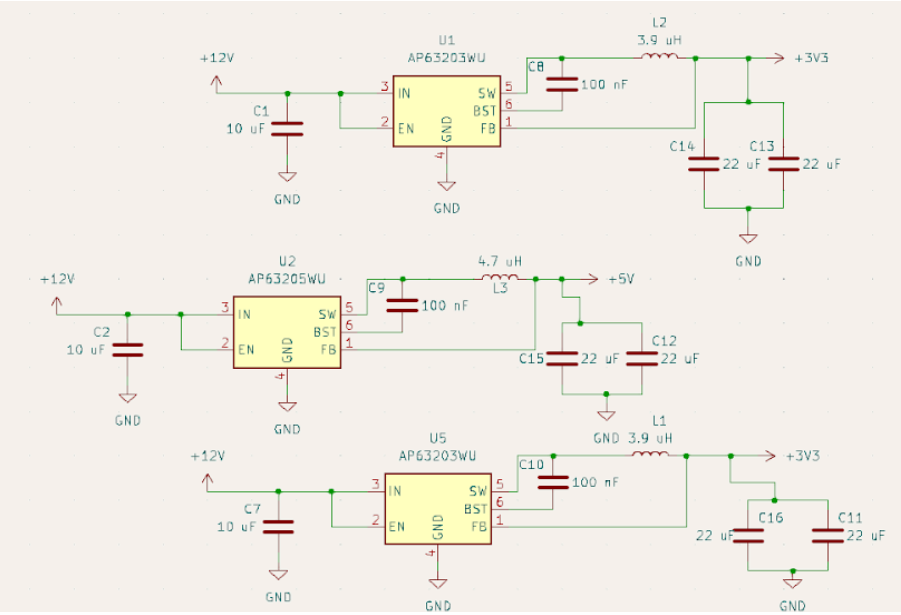


Figure 7: Schematic for Power Subsystem

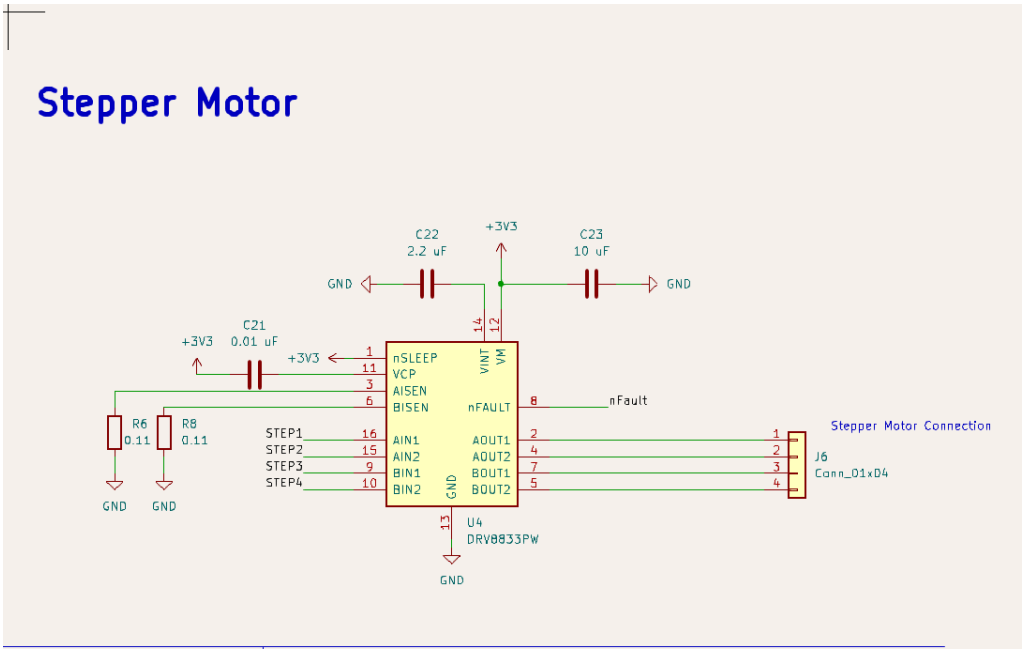


Figure 8: Schematic for Stepper Motor within Mechanical Subsystem

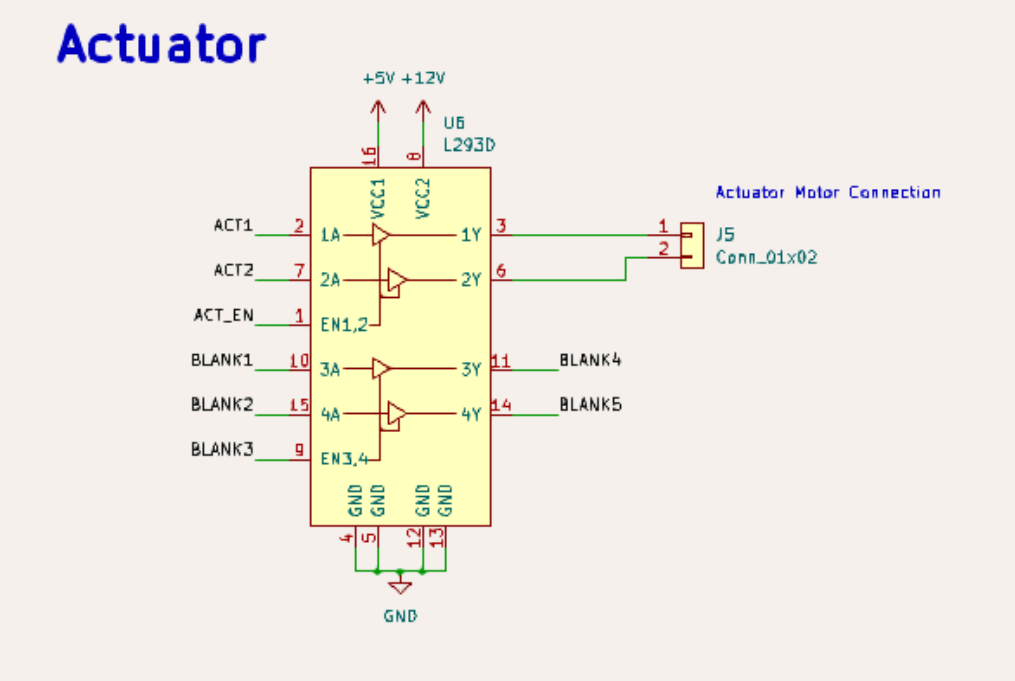


Figure 9: Schematic for Linear Actuator within Mechanical Subsystem

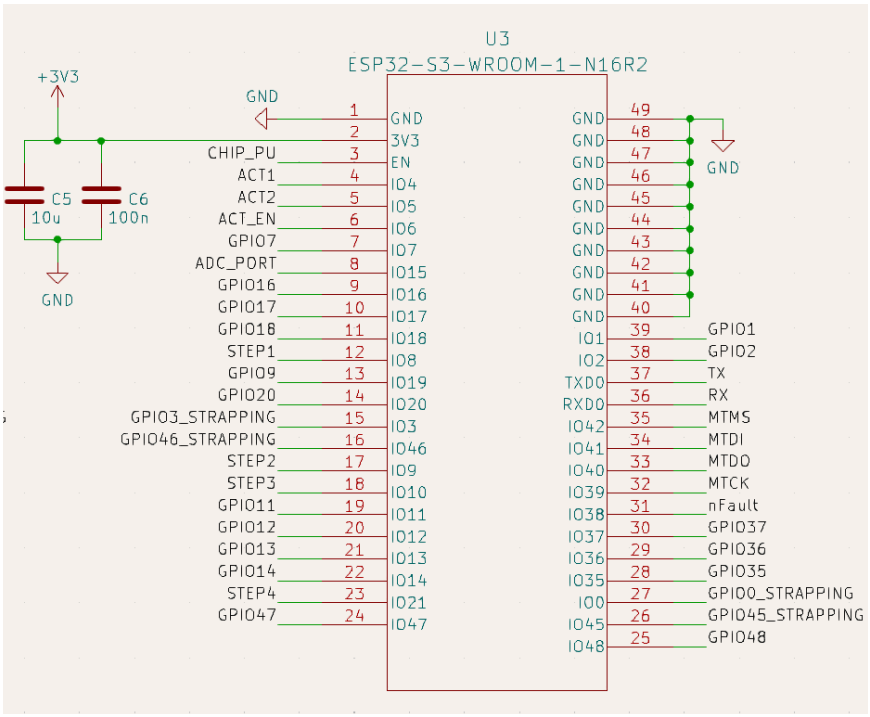


Figure 11: Schematic for Control Subsystem

Appendix C Database

AUTO ICE FISHING UML DIAGRAM

Figure 3: UML Diagram of Auto Ice Fishing Database

```
mysql> describe User;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| userID    | int          | NO   | PRI | NULL    |       |
| firstName | varchar(255) | NO   |     | NULL    |       |
| lastName  | varchar(255) | NO   |     | NULL    |       |
| username  | varchar(255) | NO   | UNI | NULL    |       |
| password  | varchar(255) | NO   |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figure 4: Description of User table in Database

```
mysql> describe Fish;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| fishID     | int          | NO   | PRI | NULL    |       |
| userID     | int          | NO   | MUL | NULL    |       |
| catchDate  | datetime     | YES  |     | NULL    |       |
| location   | varchar(255) | YES  |     | NULL    |       |
| depth      | int          | YES  |     | NULL    |       |
| lure       | varchar(255) | YES  |     | NULL    |       |
| typeOfFish | varchar(255) | YES  |     | NULL    |       |
| length     | double       | YES  |     | NULL    |       |
| weight     | double       | YES  |     | NULL    |       |
| other      | varchar(255) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
10 rows in set (0.01 sec)
```

Figure 5: Description of Fish table in Database

Appendix D Server

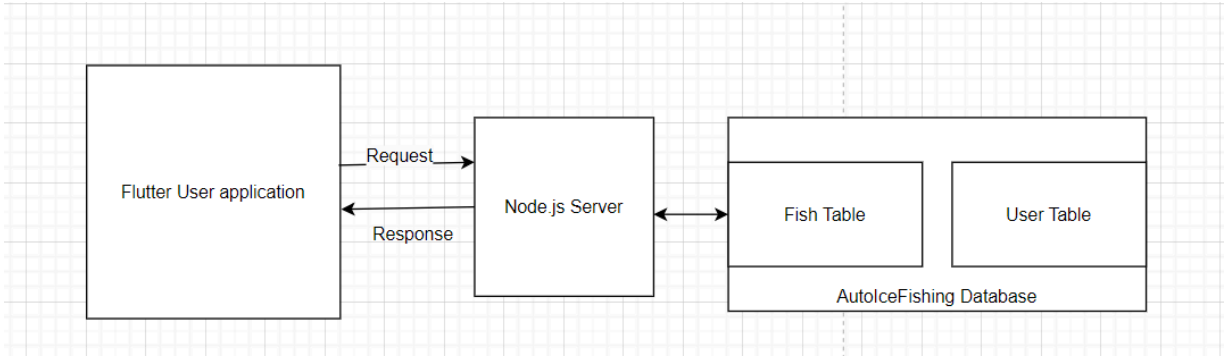


Figure 6: Overview of HTTP Response/Request Flow in Application Subsystem

```
/* LOGIN USER - POST */
app.post('/login', {
   Read username and password from HTTP request body
   Query MySQL database for user with matching username and password
   If (user in database)
```

```
      Create user session with user data
      Send success message in return HTTP response
   Else
      Send fail message in return HTTP response
  });
/* CREATE USER ACCOUNT - POST*/
  app.post('/create_account',{
      Query MySQL database for max userID
      Read user account data from HTTP request body
      Try to insert user account data into database with max userID + 1
      If (no error)
            Create user session with user data
            Send success message in return HTTP response
      Else if (username in database already)
            Send duplicate error message in return HTTP response
      Else
            Send fail message in return HTTP response
 });
/* DELETE USER - POST */
app.post('/delete_profile', {
   Delete user from MySQL database
   Destroy user session
   Send success/fail message in return HTTP response
  });
app.post('/add_catch', {
      Query MySQL database for max fishID
      Read fish catch data from HTTP request body
     Insert catch data into database with max fishID + 1 & session userID
      If (no error)
            Send success message in return HTTP response
      Else
            Send fail message in return HTTP response
  });
app.get('/get_catch', {
      Query MySQL database for catch data with current userID
      Send catch data in return HTTP response
 });
app.post('/logout, {
      Destroy user session
      Send success/fail message in return HTTP response
 });
app.post('/update_account', {
      Read user account data from HTTP request body
      Set user session data to the newly passed in user data
```

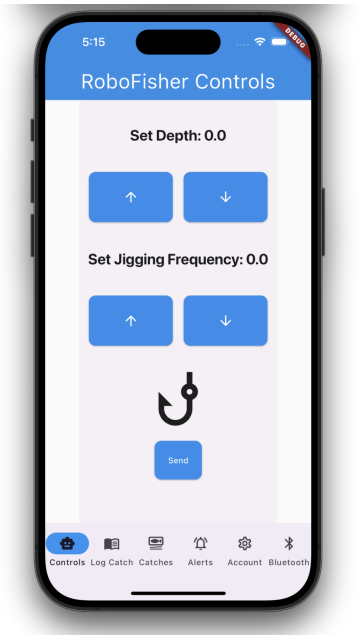
```
      Update MySQL database for user with current userID
      Send success in return HTTP response
 });

app.post('/delete_catch', {
      Read fishID from HTTP request body
      Delete catch data from MySQL database for catch with the userID for that
user session and fishID
      Send success in HTTP return message
 });
```

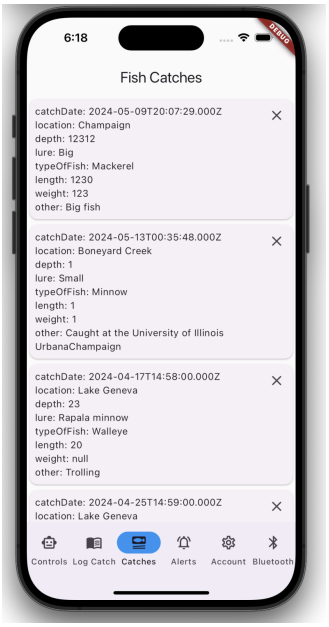
Figure 12: Pseudocode for middleware handling client requests in App.js:

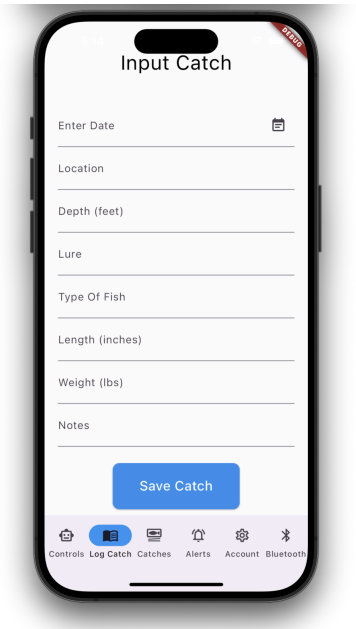
```
> nodeserver@0.0.0 start
> node app.js

AutoIceFishing app is running on port 3000
Connected to Auto Ice Fishing DB
POST /login 200 27.398 ms - 16
POST /add_catch 200 19.739 ms - 22
GET /catch 200 21.665 ms - 1505
GET /account 200 2.369 ms - 81
POST /add_catch 200 16.326 ms - 22
POST /add_catch 200 18.471 ms - 22
POST /add_catch 200 16.046 ms - 22
POST /add_catch 200 17.245 ms - 22
POST /add_catch 200 15.345 ms - 22
GET /account 200 2.260 ms - 81
POST /update_account 200 16.367 ms - 22
POST /logout 200 2.446 ms - 20
```

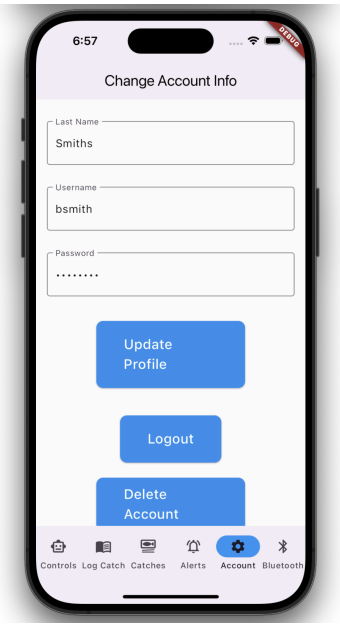
Figure 13: Example of Incoming Requests to Server

Appendix E Mobile Application









Appendix F Cost Table

Description	Manufacturer	Quantity	Extended Price ($)	Link
Flex Sensor 95mm Male Pins	Spectra Symbol	1	25.45	LINK
Nema17 Stepper Motor 3.3 V 95oz-in 60mm 1.8A,Hybrid 2-Phase 0.65NM 4-Wire 17HS4218	CNCTOPBAOS	1	17.50	LINK
L293D Quadruple Half-H Drivers	Texas Instruments	2	8.11	LINK
DRV8833PW Dual H-bridge motor driver	Texas Instruments	1	2.84	LINK
NORJIN 12V Mini Electric Linear Actuator 0.8" Stroke, 32N/7.2lbs, Speed 15mm/s, Linear Motion Actuators with Mounting Brackets	Poweka	1	22.00	LINK
AP63203WU-7 Buck Switching Regulator IC Positive Fixed 3.3V 1 Output 2A	Diodes Incorporated	4	3.48	LINK
AP63205WU-7 Buck Switching Regulator IC Positive Fixed 5V 1 Output 2A	Diodes Incorporated	2	1.74	LINK
ESP32-S3-WROOM-1-N8	Espressif Systems	1	3.20	LINK
12v 10Ah SLA Rechargeable Battery - F2 Terminals	ExpertPower	1	24.50	LINK
360 Degrees MiniPort Fishing Rod Holder	METER STAR	1	14.97	LINK
Ugly Stik Complete Spincast Reel and Fishing Rod Kit	Ugly Stik	1	32.37	LINK
Machine Shop - Parts	N/A	N/A	25	N/A
Machine Shop - Labor	N/A	N/A	450	N/A
Miscellaneous Circuit Elements (Resistors, Etc.)	N/A	N/A	10	N/A

Table 6: Project Cost

Appendix G Control Pseudocode

```
void loop() {
  If (jigging frequency == 0) {
    If (new depth in user settings != depth the line is currently at) {
      Engage the motor to reel in/out the line to the new depth
      Update the current depth (global variable)
    }
    If (sense function returns true) {
      Reel in the line all the way
      Set the terminate flag to 1 (to signal a fish has been caught)
    }
  } else {
    If (terminate flag is 0 (fish has not been detected)) {
      Signal the actuator to extend
      For (i from 0 to 4) {
        If (sense function returns true) {
          Signal the actuator to halt jigging
          Reel in the line all the way
          Set the terminate flag to 1
          Break
        }
        delay(500)
      }
    }
    If (terminate flag == 0) {
      Signal the actuator to retract
      For (i from 0 to 4) {
        If (sense function returns true) {
          Signal the actuator to halt jigging
          Reel in the line all the way
          Set the terminate flag to 1
          Break
```

```
    }
    delay(500)
  }
}
If (terminate flag == 0) {
  Signal the actuator to retract
  For (i from 0 to 4) {
    If (sense function returns true) {
      Signal the actuator to halt jigging
      Reel in the line all the way
      Set the terminate flag to 1
      Break
    }
    delay(500)
  }
}
If (terminate flag is 0) {
  Signal the actuator to halt jigging
  For (i from 0 to the period (corresponding to the jig frequency)) {
    If (sense function returns true) {
      Reel in the line all the way
      Set the terminate flag to 1
      Break
    }
    delay(1000)
    If (new depth in user settings != depth the line is currently at) {
      Engage the motor to reel in/out the line to the new depth
      Update the current depth (global variable)
    }
  }
 }
}
If (terminate flag == 1) {
  Set depth and frequency values to 0
  Clear the terminate flag
  Call the sense function again to signal the user application if a fish is still on after reeling in
}
}
```

Figure 14: Pseudocode for Control Subsystem's Main Loop

Appendix H Requirements & Verifications

Requirements	Verifications	Success (Yes or No)
• The bend angle calculated using the sensor subsystem's output voltage value is within 10 degrees of the actual bend angle of the rod	• Connect one end of a voltmeter to the sensor subsystem's voltage output and the other end to ground • Bend the rod to an angle of 45 degrees • Use a protractor to measure the actual bend angle of the rod and take note of this value • Take note of the voltage being read by the voltmeter • Use this voltage to calculate the resistance of the flex sensor, and then use this resistance to calculate the corresponding bend angle • Compare the calculated bend angle with the actual bend angle and ensure the difference is less than 10 degrees	Yes
• When auto-reeling is enabled the line is reeled in when the rod is bent at an angle of 30 (+/- 10) degrees	• Ensure that auto-reeling is enabled in the user application settings • Place the fishing device on an elevated surface (at least 5 ft) with the lure hanging over the edge • Change the settings in the user application to a line depth of 3 ft • Bend the rod to an angle of 30 (+/- 10) degrees to signal a fish on the line • Ensure that the stepper motor fully reels in the line	Yes

Table 1: R&V for Sensor Subsystem

Requirements	Verifications	Success (Yes or No)

• The user is able to create a unique account and login to the system	• Turn on the DB instance and application server • Have the user create an account in the system • Check the database to see that the users' first name, last name, username, and password have been stored • Have server print out when a request to login is made and a user session is made	Yes
• The user is able to store catch information between 7 different attributes while logged in.	• Ensure the instance is running the same as above • Have the user logged in already • Input a catch information entry into the app • Query DB to see successful store for user • Log out of the application • Log back in and go to view catch information to see that the user can still view ONLY their catch information	Yes
• The application can allow the user to set the lure depth up to 50 feet and set 3 different jigging settings.	• Rod controls in application can enable depth and jigging • Print packet sent from application to microcontroller and ensure data passed matches input • Ensure that microcontroller has received the packet/request from the mobile application	Yes

Table 2: R&V for Application Subsystem

Requirements	**Verifications**	**Success (Yes or No)**
• The power subsystem is able to output 3.3V (+/-	• Connect one end of a voltmeter to the 3.3V	Yes

0.1V) to be used as the sensor subsystem's VDD	output voltage line of the power subsystem (the one that will be used by the sensor and microcontroller) and the other end to ground • Note this voltage value and ensure it is between 3.2V and 3.4V	
• The power subsystem is able to output 3.3V (+/- 0.1V) to be delivered to the ESP32	• Connect one end of a voltmeter to the 3.3V output voltage line of the power subsystem (the one that will be used by the sensor and microcontroller) and the other end to ground • Note this voltage value and ensure it is between 3.2V and 3.4V	Yes
• The power subsystem is able to output 3.3V (+/- 0.1V) to be delivered to the stepper motor driver (DRV8833PW)	• Connect one end of a voltmeter to the 3.3V output voltage line of the power subsystem (the one that will be used by the mechanical subsystem) and the other end to ground • Note this voltage value and ensure it is between 3.2V and 3.4V	Yes
• The power subsystem is able to output 5V (+/- 0.1V) to be delivered to the actuator driver's (L293D) VCC1 port	• Connect one end of a voltmeter to the 5V output voltage line of the power subsystem and the other end to ground • Note this voltage value and ensure it is between 4.9V and 5.1V	Yes
• The power subsystem is able to output 12V (+/- 0.1V) to be delivered to the actuator driver's (L293D) VCC2 port	• Connect one end of a voltmeter to the 12V output voltage line of the power subsystem (which is connected directly to the battery)	Yes

	and the other end to ground • Note this voltage value and ensure it is between 11.9V and 12.1V	

Table 3: R&V for Power Subsystem

3.4 Mechanical Subsystem

Requirements	**Verifications**	**Success (Yes or No)**
• The stepper motor is able to lower the fishing line to 50 ft (+/- 5 ft) when fully reeled out	• Go into a stairwell with a height of at least 50 ft and place the fishing device so that the lure is hanging above the stairwell • Measure the length of line out with tape measure.Note this length and ensure that it is between 45 ft and 55 ft	Yes
• The stepper motor is able to **lower/raise** the fishing line in increments of 1 ft (+/- 0.25 ft)	• Place the fishing device on an elevated surface (at least 5 ft) with the lure hanging over the edge • Use a tape measure to measure the actual length between the start and end of the line and note this length • Ensure that this difference is between 0.75 ft and 1.25 ft	
• The actuator is able to jig the rod up and down within 0.1 Hz of the desired frequency	• Change the settings in the user application to one of the three possible frequencies • Start a stopwatch at one of the jigs • Stop the stopwatch at the next jig • Note the time from the stopwatch (this is the period of the jigs) • Calculate 1 divided by the measured period to	Yes

	get the actual jig frequency • Compare this with the frequency set on the user application and ensure the values are within 0.1 Hz of each other • Repeat the above steps for the other two jigging frequencies	
• When a fish is detected by the sensor subsystem, the stepper motor fully reels in all of the line currently out, returning to the original position of a depth of 0 ft (+/- 1 ft)	• Place the fishing device on an elevated surface (at least 5 ft) with the lure hanging over the edge • Bend the rod to an angle of 30 (+/- 10) degrees to signal a fish on the line • Wait for the stepper motor to reel in the line • Measure the length of the remaining line out by placing one end of a tape measure at the tip of the rod and the other end at the lure and note this length (Ensure that this length is less than 1 ft)	Yes
• The stepper motor is able to fully reel in a weight of 1 lbs (+/- 0.5 lb)	• Place the fishing device on an elevated surface (at least 5 ft) with the lure hanging over the edge • Attach a 1 lb (+/- 0.5 lb) weight to the end of the line • Bend the rod to an angle of 30 (+/- 10) degrees to signal a fish on the line • • If the stepper motor is able to reel in the weight, the test succeeded	Yes
• The jigging motion halts	• Bend the rod to an angle	Yes

within 5 seconds when the bend angle of the rod is 30 (+/- 10) degrees	of 30 (+/- 10) degrees to signal a fish on the line and start a stopwatch • Wait until the actuator stops the jigging motion and stop the stopwatch • Ensure that the time on the stopwatch is less than 5 seconds	

Table 4: R&V for Mechanical Subsystem

Requirements	**Verifications**	**Success (Yes or No)**
• The microcontroller can receive requests from the application via Bluetooth	• Send requests from user application such as setting the line depth or jigging frequency • Program microcontroller to print out data taken from application • Ensure the data is consistent and that correct values exist in microcontroller functions	Yes
• The microcontroller is able to send a notification to the user application via Bluetooth when the sensor subsystem detects a bend angle of 30 (+/- 10) degrees	• Open the user application • Bend the rod to an angle of 30 (+/- 10) degrees to signal a fish on the line • Ensure that the user application has received a notification from the microcontroller	Yes
• The microcontroller is able to accurately read the voltage value output from the sensor subsystem within 0.1V	• Place one end of a voltmeter between the flex sensor and the 10k Ω resistor and connect the other end to ground • Take note of this voltage value (which is the output of the sensor subsystem) • Run code on the microcontroller to perform an analogRead from the appropriate	Yes

	GPIO port (GPIO15) • Take note of this voltage value (this is the voltage value being read by the ESP32) • Compare this with the previous voltage value and ensure the values are within 0.1V	
• When the bend angle of the rod is 30 (+/- 10) degrees the control subsystem will signal the mechanical subsystem to halt the jigging motion within 5 seconds of detecting the fish	• Bend the rod to an angle of 30 (+/- 10) degrees to signal a fish on the line and start a stopwatch • Wait until the actuator stops the jigging motion and stop the stopwatch • Ensure that the time on the stopwatch is less than 5 seconds	Yes
• When the bend angle of the rod is 30 (+/- 10) degrees and auto-reeling is enabled the control subsystem will signal the mechanical subsystem to reel in the line	• Ensure that auto-reeling is enabled in the user application settings • Place the fishing device on an elevated surface (at least 5 ft) with the lure hanging over the edge • Change the settings in the user application to a line depth of 3 ft • Bend the rod to an angle of 30 (+/- 10) degrees to signal a fish on the line • Ensure that the stepper motor fully reels in the line	Yes

Table 5: R&V for Control Subsystem