University of Maryland, College Park

Department of Electrical and Computer Engineering

Figure 0: Image of Team SQUID submarine final design.

ENEE408V Section: 0101
Fall 2025

Instructor: Professor Romel Gomez
TA: Gavishta Liyanage

Team S.Q.U.I.D
(Somewhat Qualified Underwater Investigation Division)
Olivia Adams, Marco Albano, Margaret Gouker, Brian Wu

Table of Contents

I. Approval of Team Members

I, Olivia Adams, ensure that I approve of all information on this document as well as the contributions of all team members.

I, Marco Albano, ensure that I approve of all information on this document as well as the contributions of all team members.

I, Margaret Gouker, ensure that I approve of all information on this document as well as the contributions of all team members.

I, Brian Wu, ensure that I approve of all information on this document as well as the contributions of all team members.

Figure 1: Team picture. From left to right: Marco Albano, Olivia Adams, Brian Wu, Marg Gouker.

II. Design and Development

Introduction

The engineering design philosophy of Team SQUID centers on creativity, communication, collaboration, and critical thinking as the foundational principles for developing the autonomous submersible vehicle. Creativity will drive the exploration of innovative propulsion systems, compact sensor integration, and adaptive control algorithms for the exploration of underwater environments. Communication will ensure consistent, reliable interaction between hardware, software, and sensing subsystems, enabling accurate movement, object detection, and navigation data flow. Collaboration will allow the team to form a cohesive and resilient system architecture and collaborative team environment, bringing together diverse skillsets to achieve a common goal. Critical Thinking will enable the evaluation of sensor feedback, anticipation of failure, refinement of navigational logic, and self correction. These principles will support the creation of the best underwater vehicle ever.

In the real world these submarines could be employed for any shallow water needs. As the submarine is not designed for deep ocean or any sort of pressurization of the capsule, it does limit what underwater work they can do. For example, these subs could be used to monitor coastal wetlands and shallow reefs to monitor and determine environmental specifications over time. They could be used to record coral bleaching events as well. Alternatively, they could be used to monitor the stability and structure of oil rigs around the surface of the water. They could also be used to monitor any oil leaks in the drilling process, as oil spilling in the ocean would eventually rise to the surface due to buoyancy. Additionally the submarine could be used as a scrubber for the rigs, removing biogrowth to maintain the integrity of the structure for longer periods of time as well as improve the safety of the rig for divers. There are a plethora of uses for this technology, since it can monitor places humans can't normally go for long periods of time.

From reading previous lab reports, there was a lot of inspiration from those teams' successes as well as failures. First, the battery was downsized a lot, which helped with space issues as well as practicality issues. Because a 7.4 V battery was used, there was less of a step down for voltage needed, and a more compact design for the electronics. Additionally two downward motors were used rather than four, as more motors required a larger battery, and it was unnecessary in order to control and maintain depth. Other teams noted and warned about issues with sealing, so minimizing points of water ingress were integral to the mission. The design itself was supposed to be modeled after a squid (but ultimately was not) since the team is Team SQUID.

Mission Statement

The goal for this project is to build a fully functional autonomous underwater vehicle that will be able to scan, find, and navigate to a small object within 1 cm of the target in the tank, within the given design constraints: full autonomy, watertight up to one meter, a maximum total volume of 20x20x20 cm cube, and ability to reach a target, and all within the semester timeframe. (15 labs total!)

Design Overview

The aim for this project is to design and develop an intelligent submersible vehicle capable of autonomous underwater communication, propulsion, sensing, and actuation. The vehicles must meet specific design challenges, including but not limited to: precise and autonomous motion control, underwater communication, and location and depth sensing.

Team SQUID has constructed a vehicle capable of these functions, and consists of a 13” length hollow plexiglass cylinder that is watertight with machined plastic ‘gland’ plugs on either exposed end. These machined plugs are hollow with an o-ring diameter that is the same as the inner diameter of the tube (3.25”). One end also features four ‘watertight’ cylindrical wire glands so that the vehicle can be self-contained and wireless, with electronic components stored within the inner cylindrical compartment connected to the external mounts of the vehicle. Connected to the outside of the submersible vehicle are external mounts to house the motors and underwater sonar distance sensors.

Each gland is capable of six wire connections through one seal. In order to prevent water ingress, the glands are sealed with epoxy upon completion, and the wire connections going through them are sealed in silicone and covered with a silicon sock.

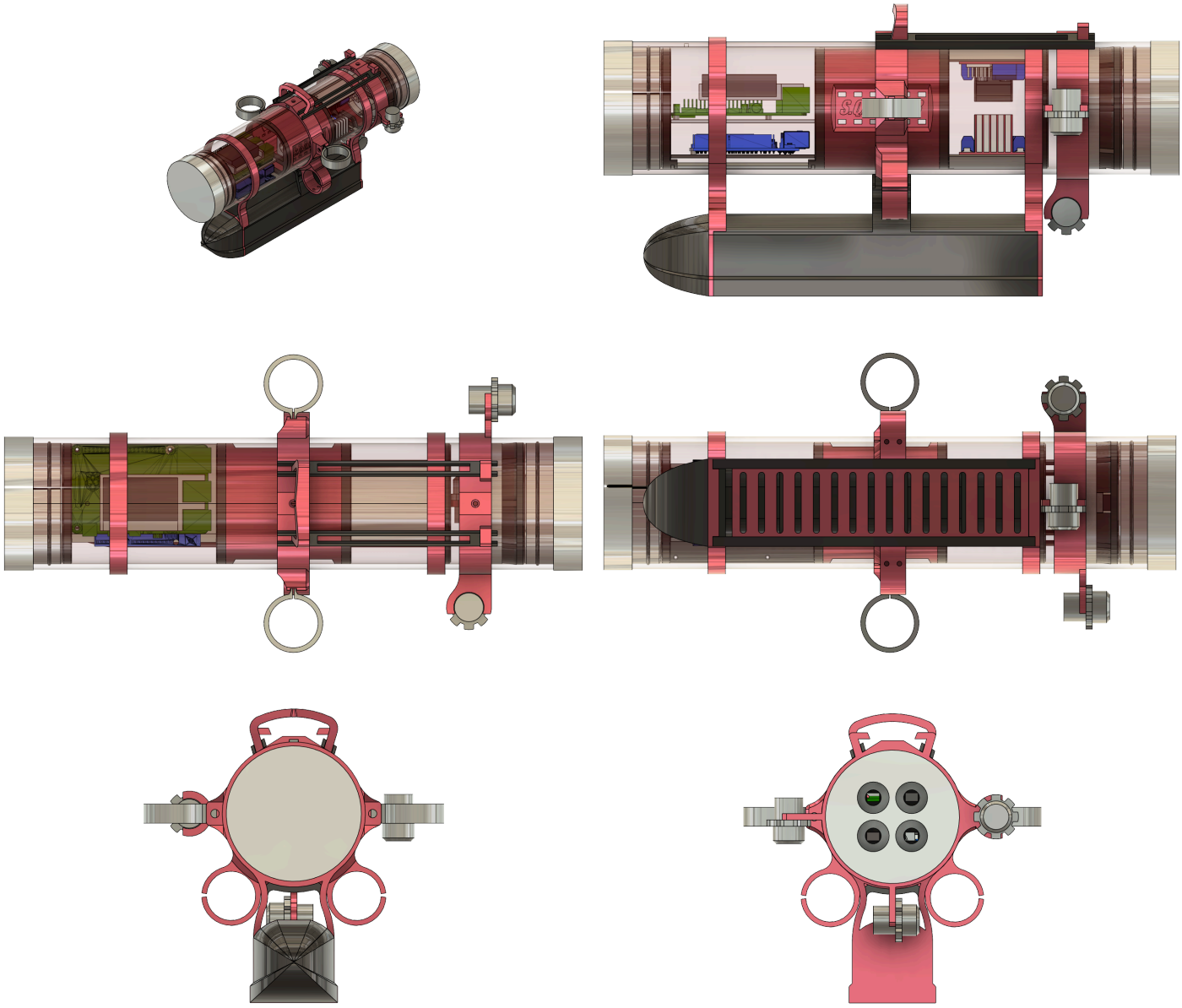

Figure 2A-F: Left to right: technical renders of isometric, side, top, bottom, front, and back views respectively.

The primary external mount is the motor mount. It is designed to smoothly slide on and off the cylindrical body of the vehicle, with a set screw retention lock to hold it in place. It is located in the very middle of the vehicle, and features inlaid nuts for simplified installation of the vertical motors. The horizontal motors are merely friction fit into the motor mounts, but feature holes for set screw retention if needed. There is a margin that allows for wiring to slip through for easy installation.

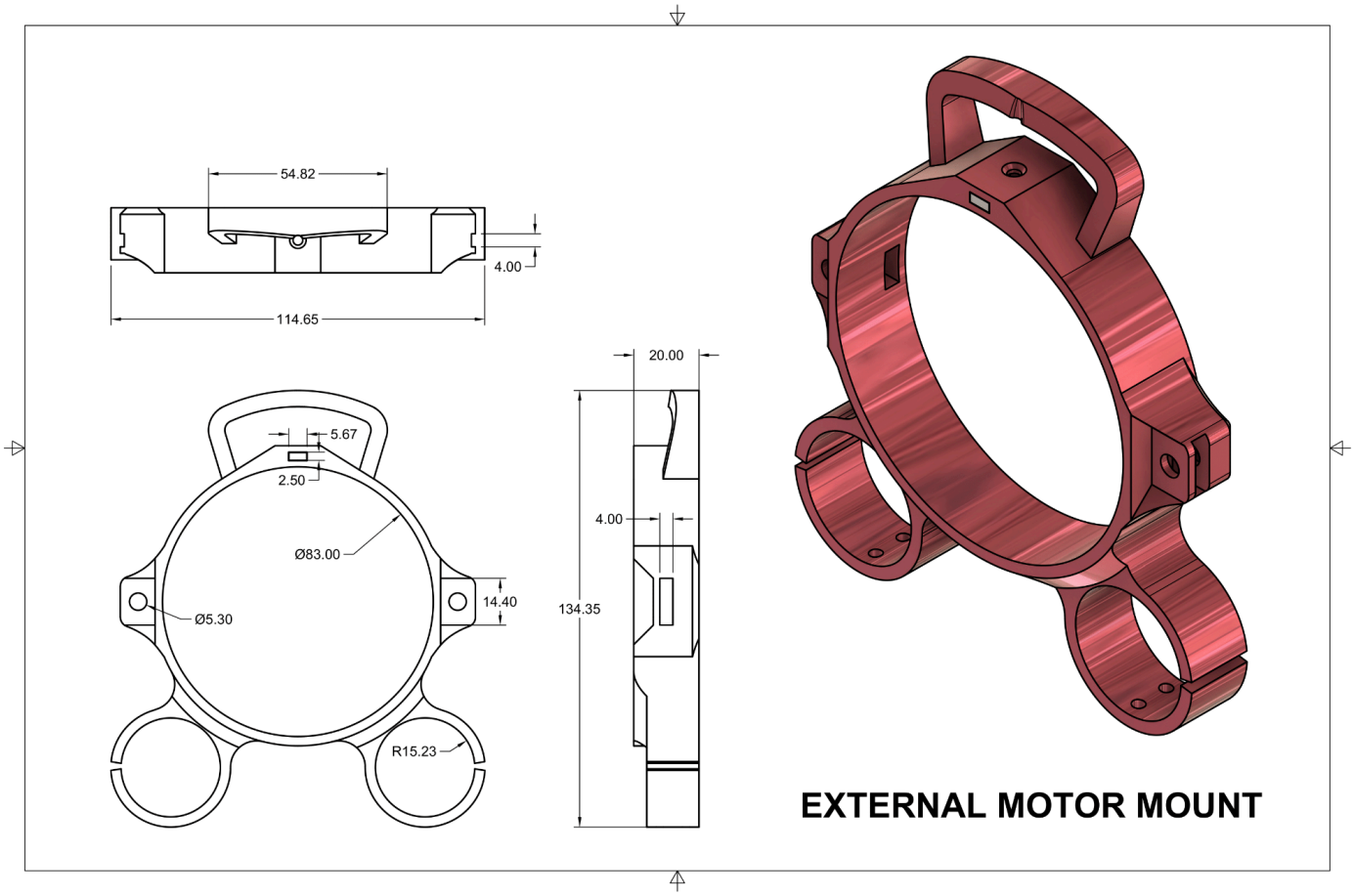

Figure 3: External motor mount working drawing.

The secondary external mount is the sonar sensor mount. It features holders for the three sonar sensors being used to measure the current location, and is easy to clip on and install each sonar. It is designed to smoothly slide on and off the cylindrical body of the vehicle, with a set screw retention lock to hold it in place. Each sonar is facing a different cartesian direction (x, y, z) for thorough distance measurement. There is also a rail system that connects both external mounts together so they are aligned and in the correct position at all times.

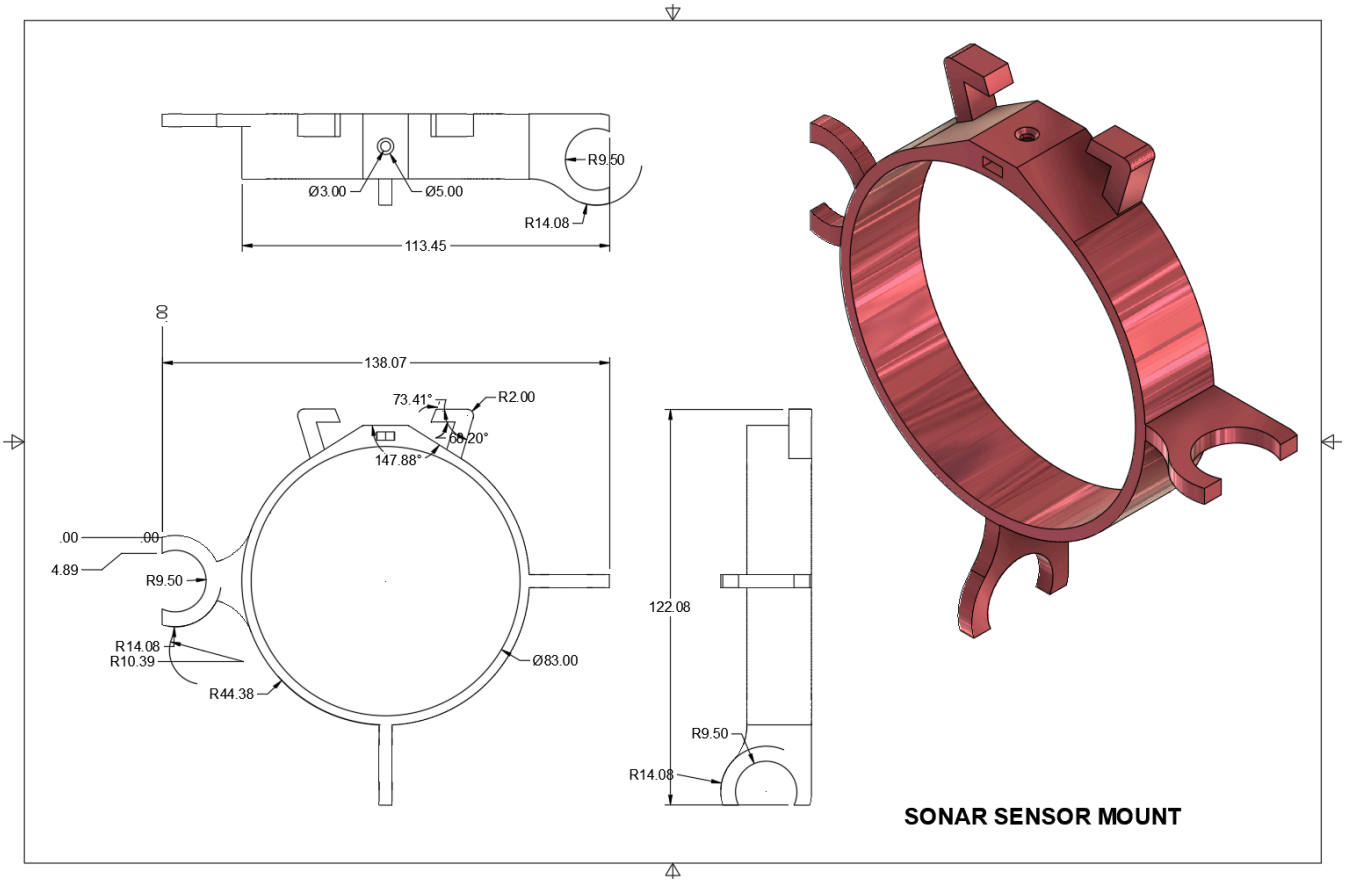

Figure 4: Sonar sensor mount working drawing.

The external tube housing and O-ring pressure fitting system is tedious to assemble and disassemble. To remedy this, the tube has a pressure valve screw hole that can be plugged by a screw in order to maintain water-impermeability. The hole acts as an escape route for any compressed air trapped within the tube during assembly, allowing for less time spent and elbow grease when it comes to assembling and disassembling the sub.

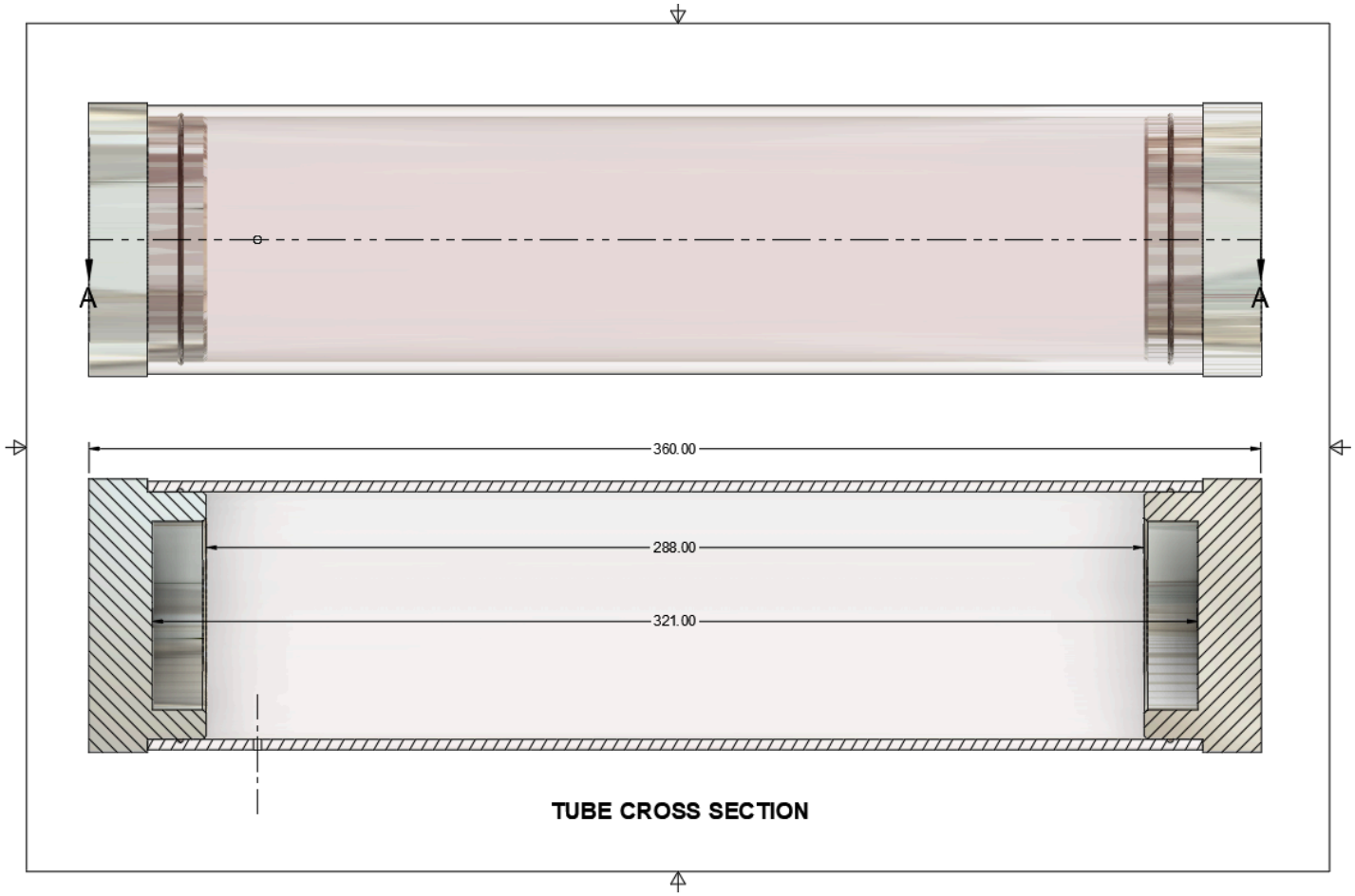

Figure 5: Tube cross section.

The sub features a proprietary interlocking electronics housing system for the simplified organization and installation of all internal electronics components. There are three main sections that all interlock via dovetail mounts and are reinforced with 5mm circular neodymium magnets.

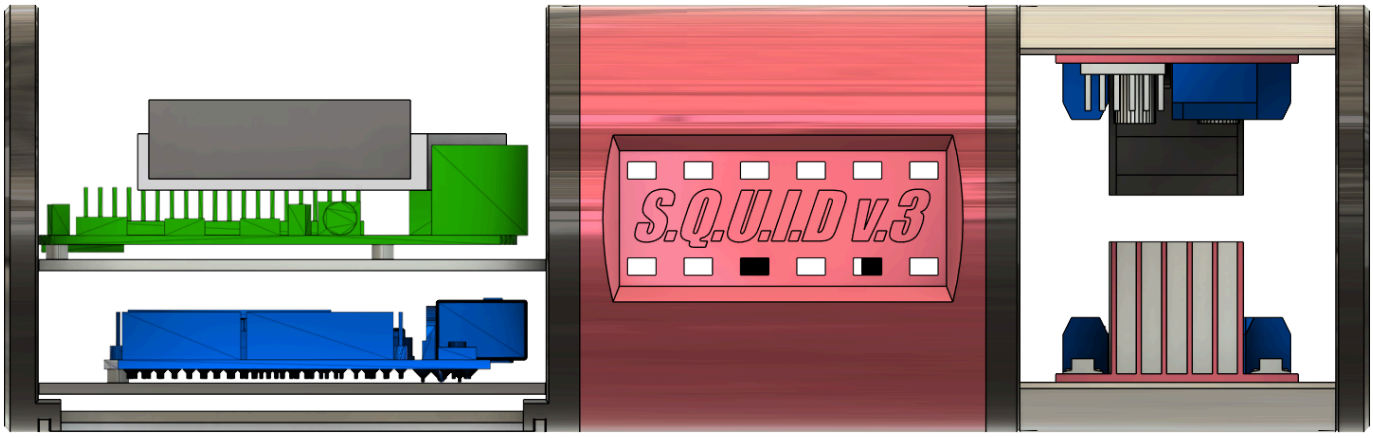

Figure 6: Internal housing with electronics.

The front section is composed of four pieces, two of which are the modular circular connectors. It has two plates that connect in between them and feature screw-in holes to mount the RaspberryPi and Arduino Uno boards respectively. All wiring then flows through the circular connectors through the other sections to get to their respective electronic components.

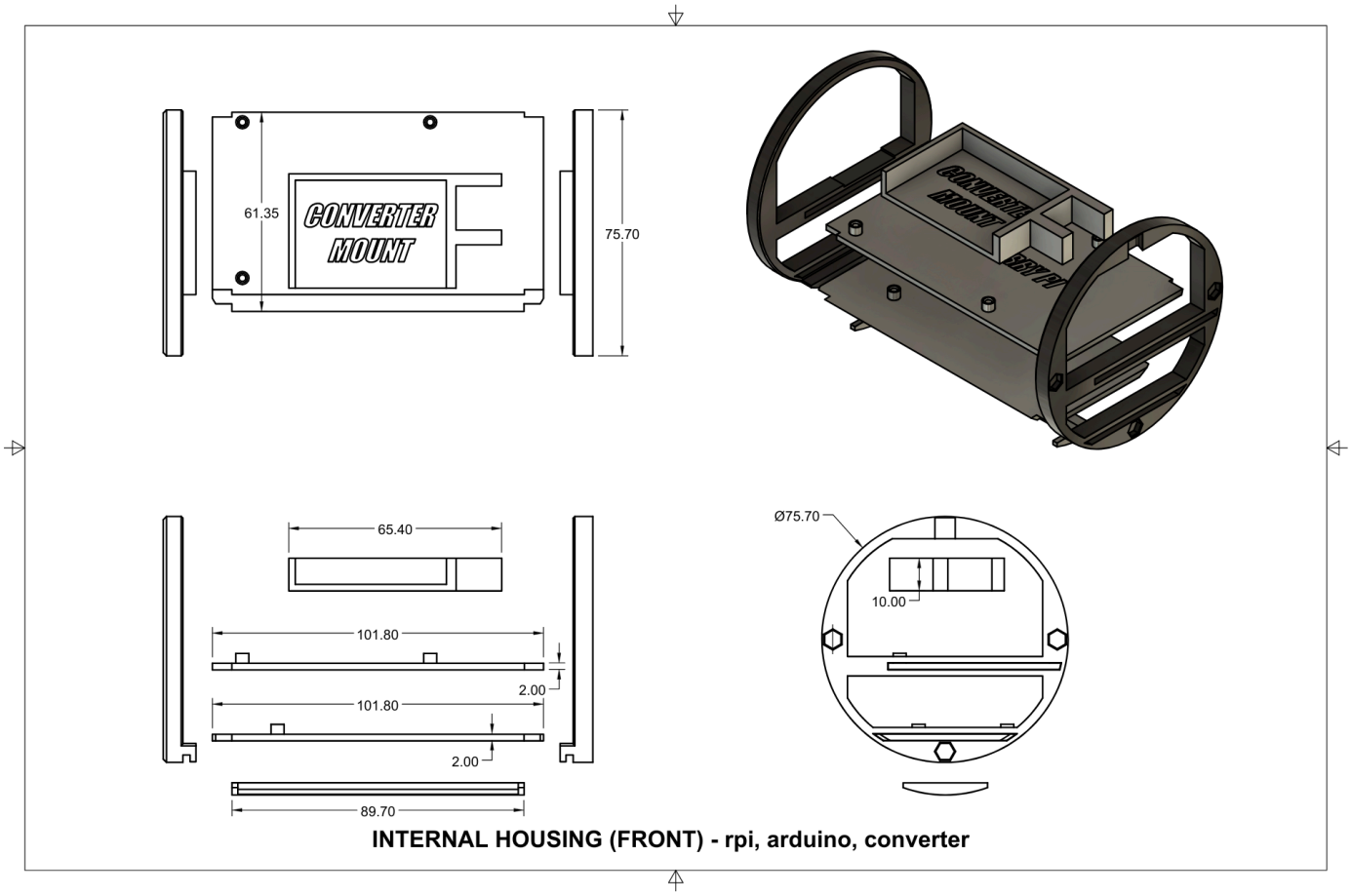

Figure 7: Front internal housing working drawing.

The middle section contains a holder for the 7.4V battery on the ceiling port to minimize contact with water in the event of flooding, and has a magnetic mount for the IMU board to sit comfortably in the exact middle position of the vehicle. Underneath the IMU mount is a kill switch housing to turn the vehicle on and off. The sides of the section feature multiple holes for wire organization, as zip ties are meant to fit through the holes to separate clusters of wires for easy labeling.

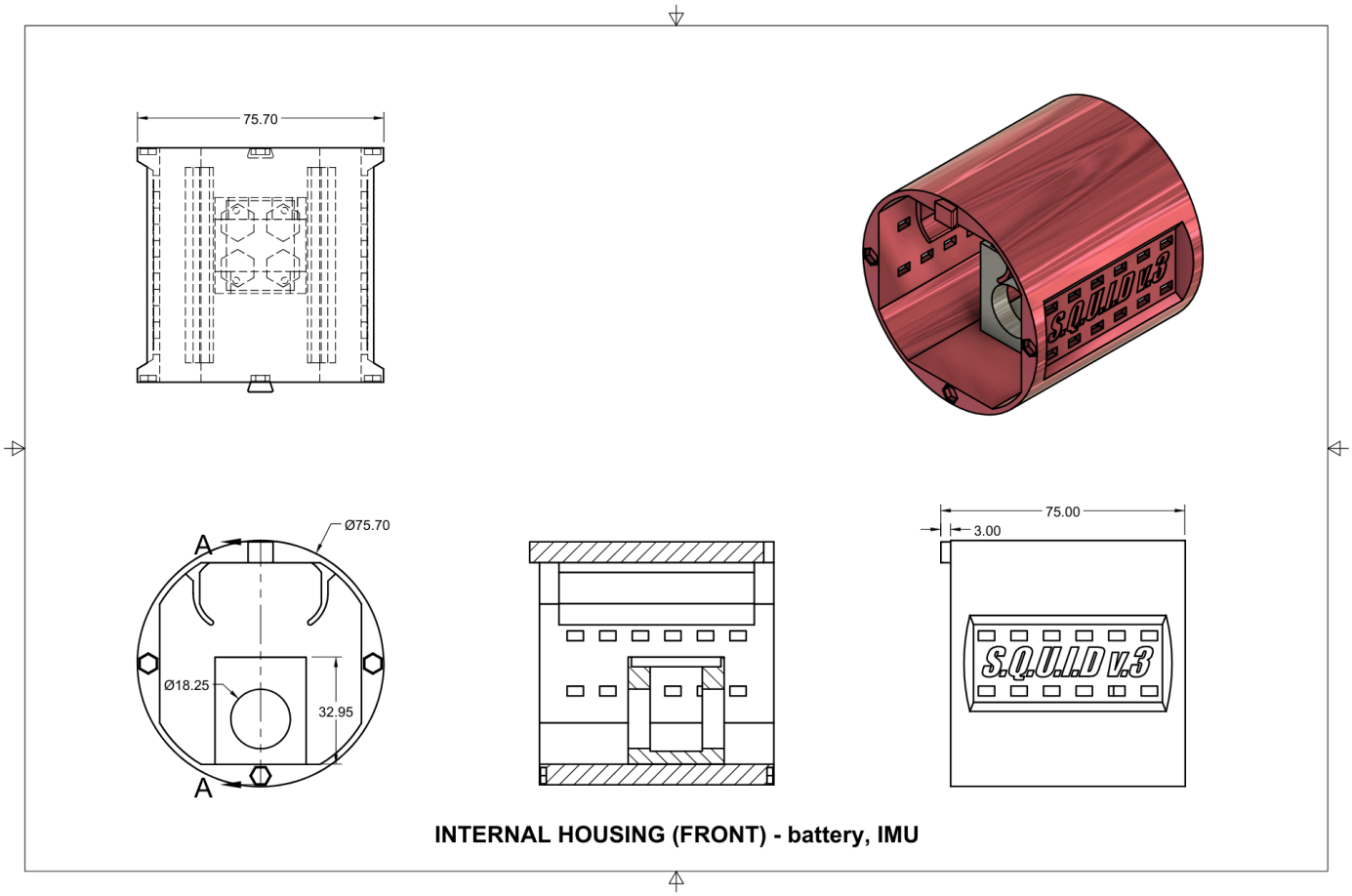

Figure 8: Middle internal housing working drawing.

The final and rear section of the housing mounts the H-bridges. The section is composed of four pieces, two of which are the modular circular connectors. There are two plates that connect in between them and feature screw-in holes to mount the H-bridges. All subsequent wiring flows through this section and into the wire glands in the back of the vehicle.

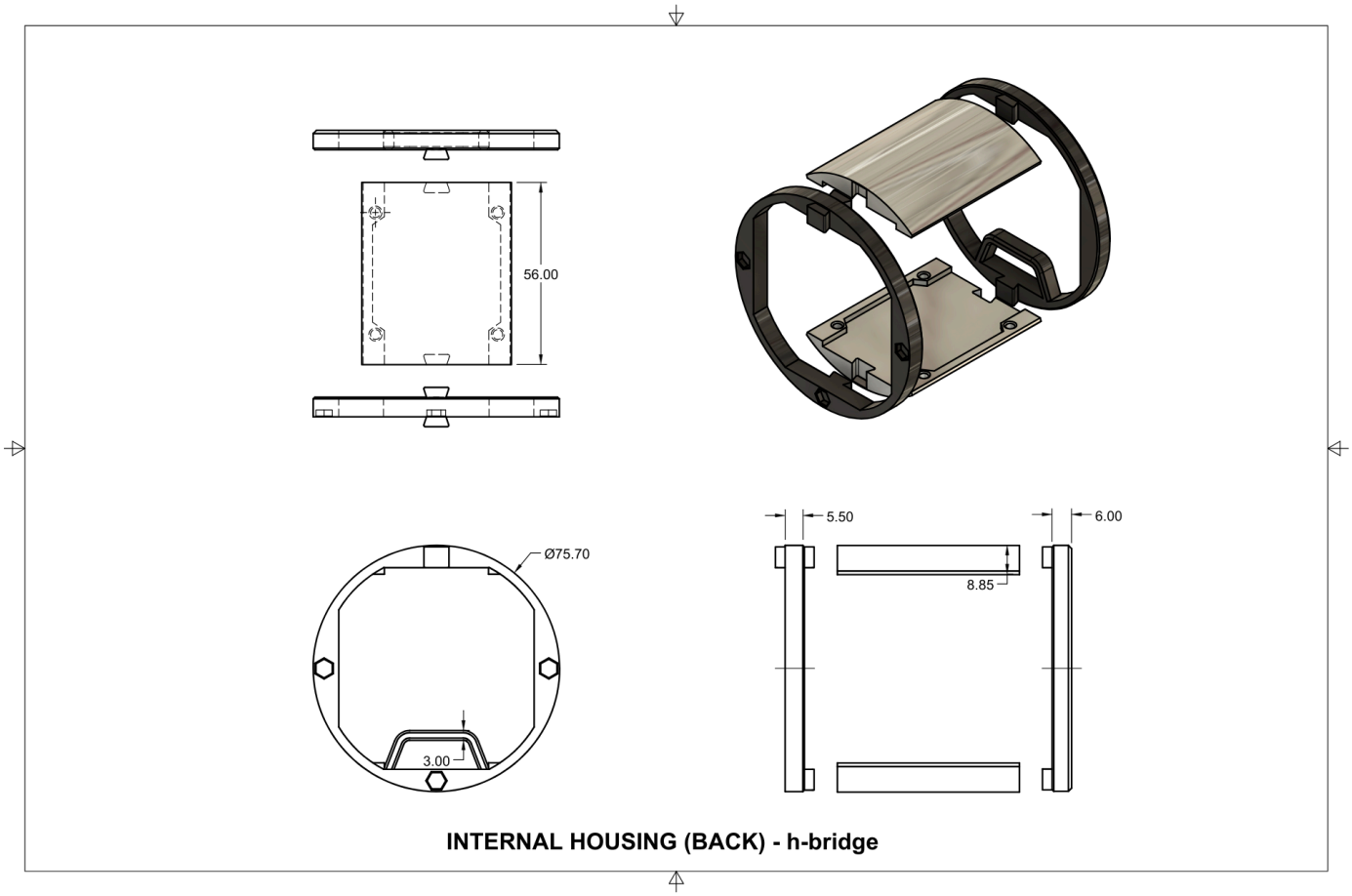

Figure 9: Back internal housing working drawing.

The overall wiring diagram is as follows, and explains how each subsystem works in conjunction with one another:

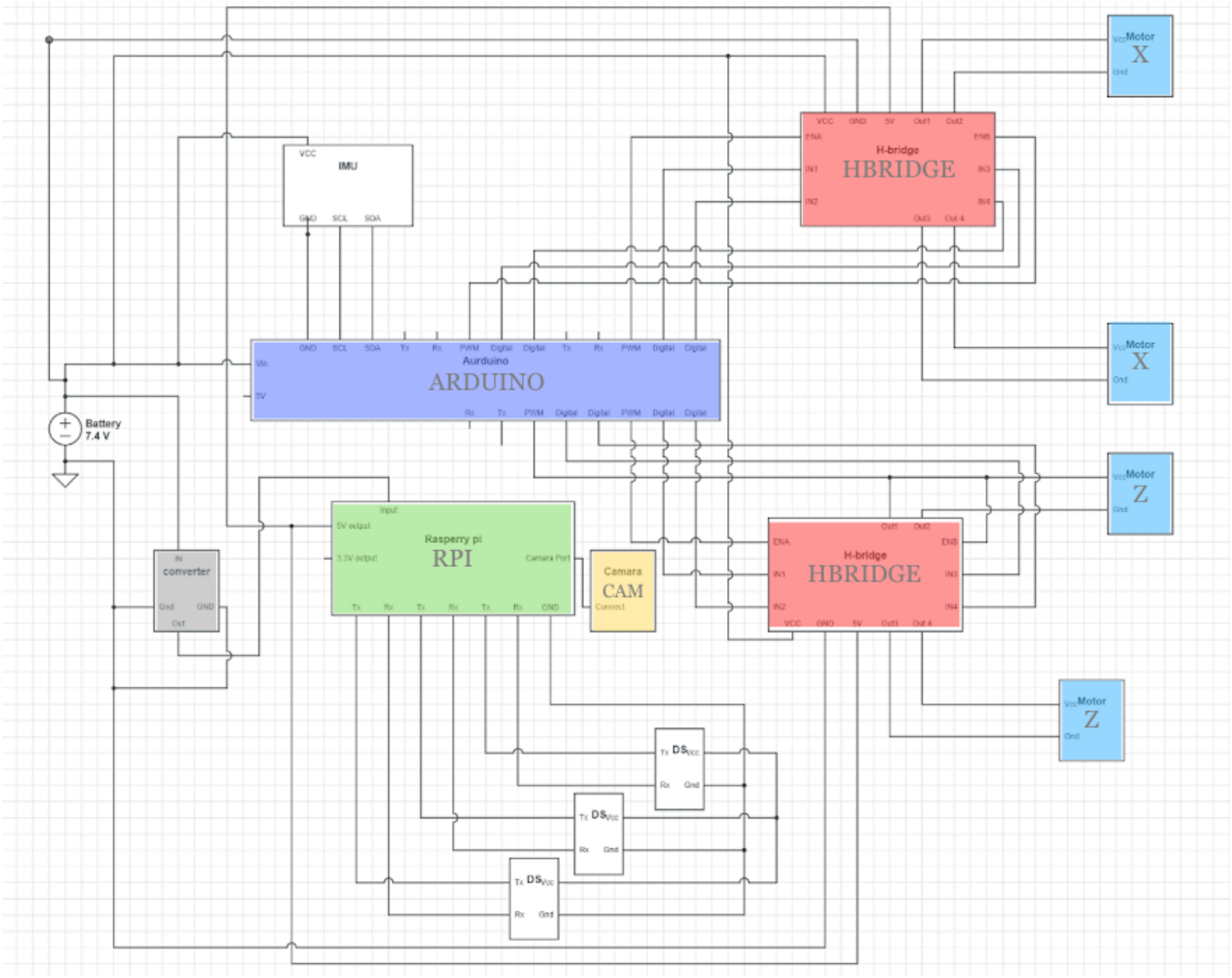

Figure 10: Complete wiring diagram.

Each wire gland contains six wires, and the chart below illustrates which external electrical components belong to each gland:

Glands	1 (6 wires)	2 (6 wires)	3 (4 wires)	4 (4 wires)
	Motor X Right (power)	Motor X Left (power)	Sonar Y (power)	Sonar Z (power)
	Motor X Right (GND)	Motor X Left (GND)	Sonar Y (transmit)	Sonar Z (transmit)
	Sonar X (power)	Motor Z Right	Sonar Y (receive)	Sonar Z (receive)

		(power)		
	Sonar X (transmit)	Motor Z Right (GND)	Sonar Y (GND)	Sonar Z (GND)
	Sonar X (receive)	Motor Z Left (power)		
	Sonar X (GND)	Motor Z Left (GND)		

To account for weight distributions and neutral buoyancy, the team introduced a weight holder "torpedo" separate from the main submarine, but attached to the underside of the hull. It features two magnetic interlocking mounting rings that hook onto the main vehicle body, with raised edges for alignment so that the main carriage compartment of the torpedo sits nicely underneath the external motor mounts. There are vents on the underside for convenient water drainage from the weight compartments. If there is a need for weight distribution, the torpedo also features a sliding door for easy access to any weights held within the main compartment.

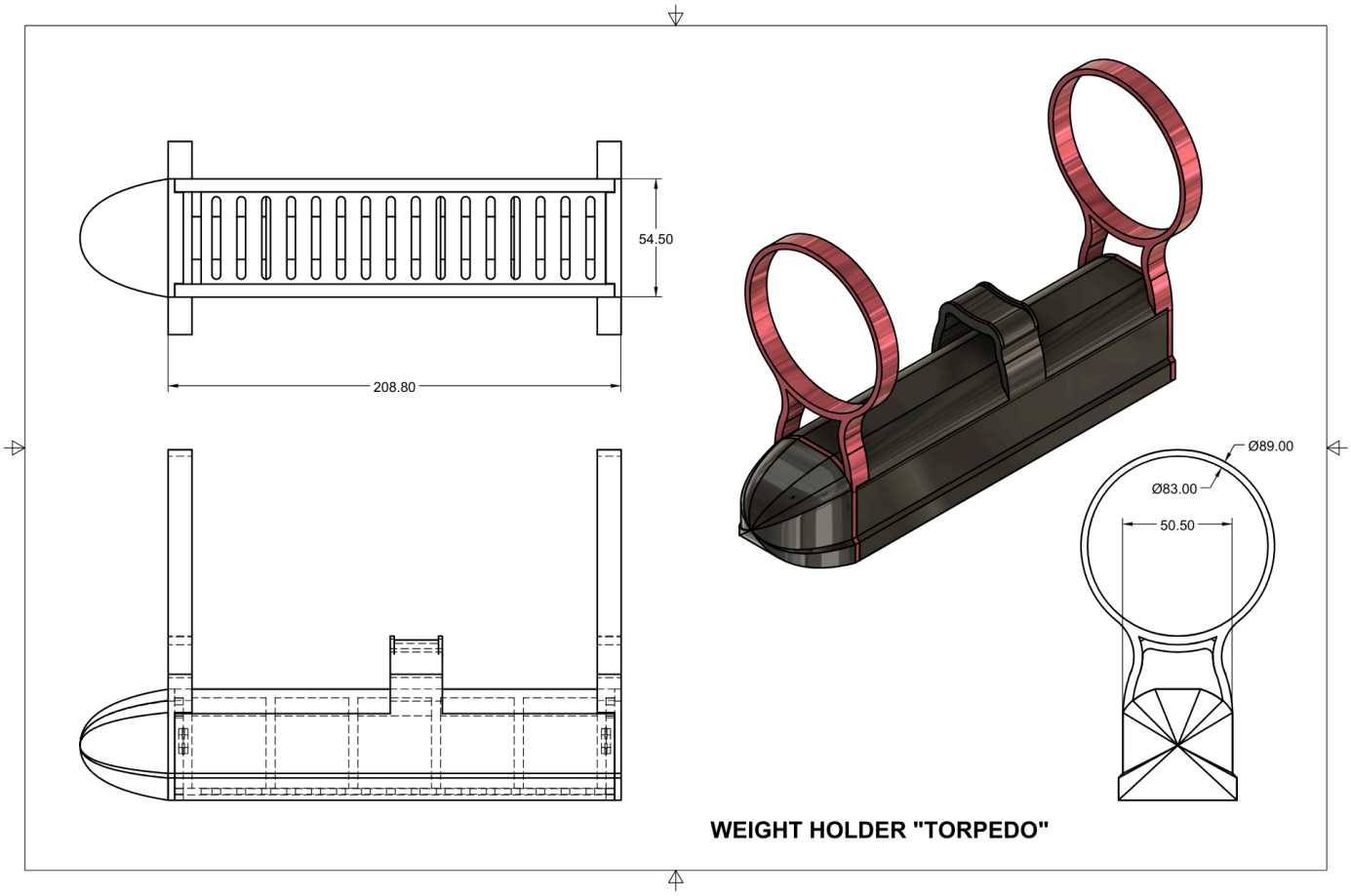

Figure 11: External weight compartment working drawing.

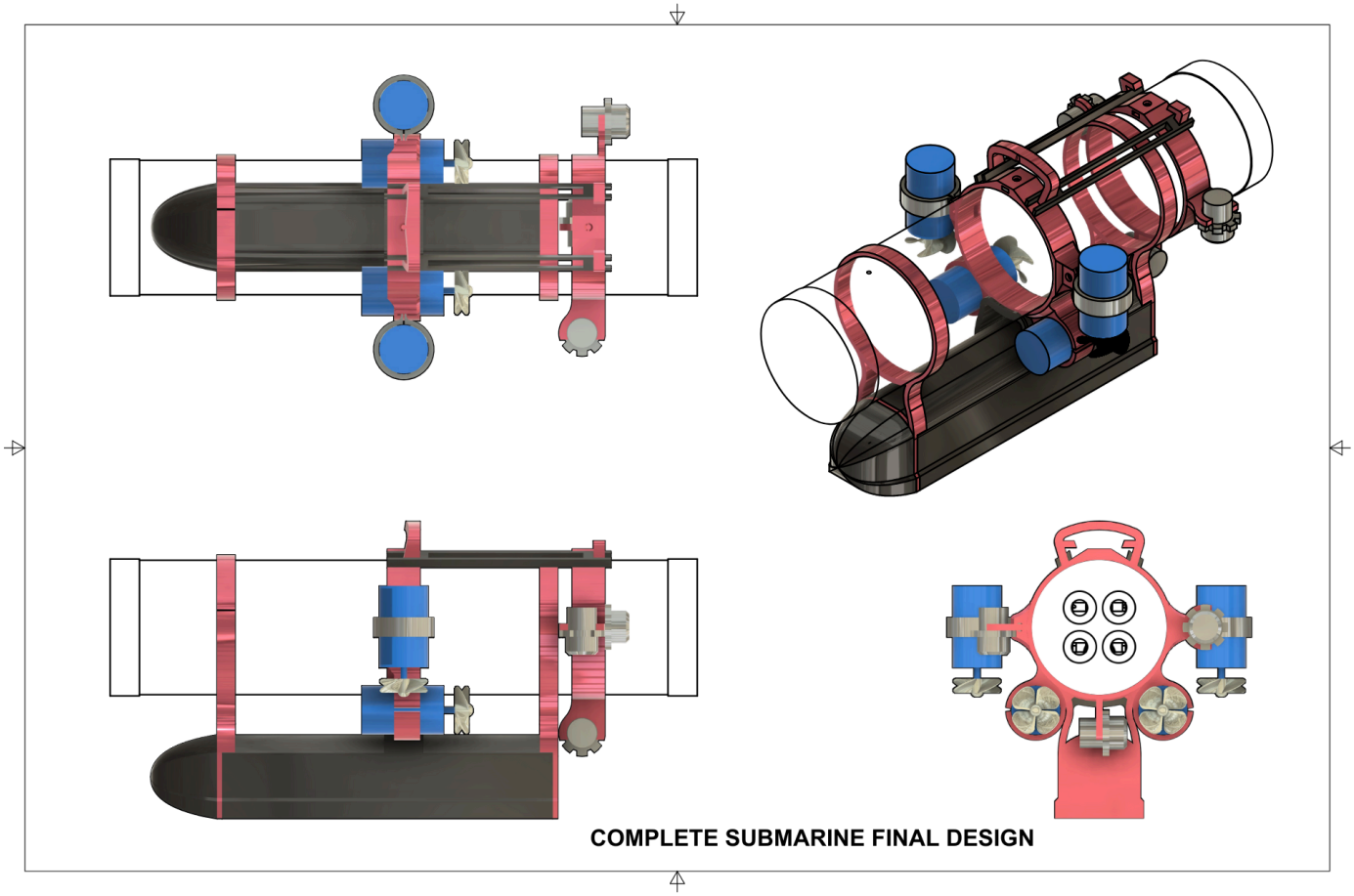

Figure 12: Complete submarine final design.

Clear Plastic Tube	450 g
Arduino Uno	25 g
Wires	100 g
PLA plastic	374 g
Distance sensors (3)	3*35 = 105 g
Battery	155 g
Raspberry Pi 4B	47 g
IMU sensor	9 g
H-Bridge (2)	2*32 = 64 g
Motors (4)	4*50 = 200 g
Lead weights	836 g
Total	2360 g (2.36 kg)

Engineering Standards

IP (Ingress Protection)

The nature of this project revolves around temporarily submerging a vehicle with electronics up to ~50cm. This requires IP67 protection (IP standards can be found here), which involves sealing all openings with rubber/silicone gaskets to protect circuits from water damage. Such gaskets were used for the caps, and additional layers of epoxy and silicone were applied on the inside of the glands to further prevent leaking. While moisture-repelling coating is typically applied on PCBs for this use case, the team taped sponges to the insides of the end caps and wrapped the electronics housing with a thin plastic wrap to prevent contact with water.

Toxicity of Materials

In order to achieve neutral buoyancy, the submersible vehicle needed to incorporate additional weights. The ones available in the lab were a variety of lead fishing weights between 0.5 - 8 oz each. When incorporating this into the design process, it was important to refer to the Occupational Safety and Health Administration (OSHA) standard for potential exposure. While the fishing weights did not hurt the overall air quality to justify strict lab-time limits, it was important to set reminders to wash hands before and after being in the lab due to physical contact with the weights.

Design Choices & Iterations

Design Progression

There have been six main iterations of the submersible vehicle before the team reached the final design as showcased in this report. The first initial render looked the most visually like a squid, with components placed without much regard for propulsion nor wire organization. The render includes a prediction of what was to be installed within the submarine, as components were not yet finalized.

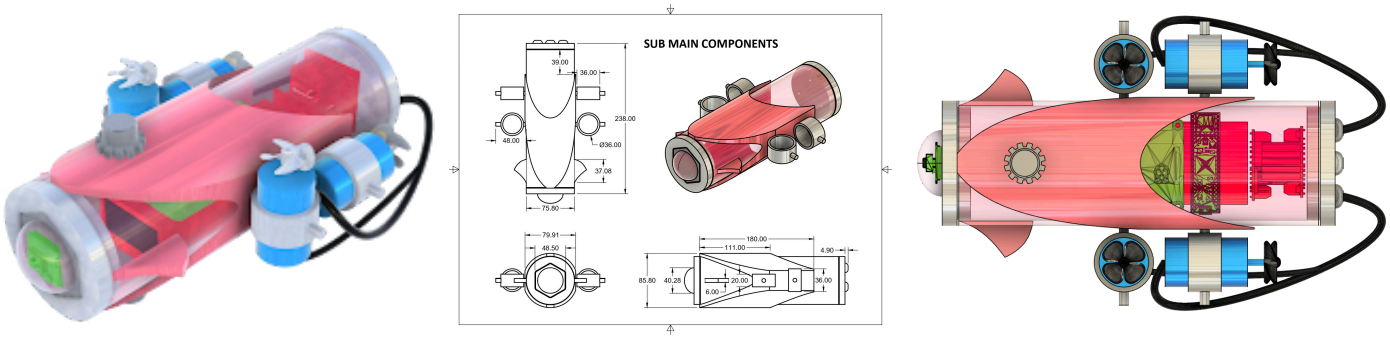

Figure 13A-C: Initial submarine design technical render, working drawing.

The team then developed a central motor mount system that would attach to the tube via screw with adjustable tolerance. This was known as the “test version” because we was under the impression that all teams were designing a remote-controlled submarine for milestone 1 instead of pursuing autonomous control immediately. At this moment, the submarine looked more like a shark than a squid. Early on in the semester, the team decided to increase the diameter size of the submarine central tube. We noted that it would be impossible to fit the central processor (RaspberryPi board) within the current tube’s diameter (3”). Note of advice: Use a 3.25” diameter **minimum** tube, with 3.5” diameter being comfortable for sizing. By the time of milestone 1, Team SQUID had a vehicle capable of propelling through the water while on a tether.

Figure 14: Milestone 1 presentation with both initial submarine prototypes.

Team SQUID continued with the central motor mount system and then introduced an auxiliary mount for the sonar sensors. The team needed to measure coordinates / distance from all three axes (x, y, z) which required the sonars to be mounted at these cardinal directions. To ensure that the sensors would be at a static placement for every assembly, aligner rods were developed in order to keep the sensor mount a set distance away from the motor mount.

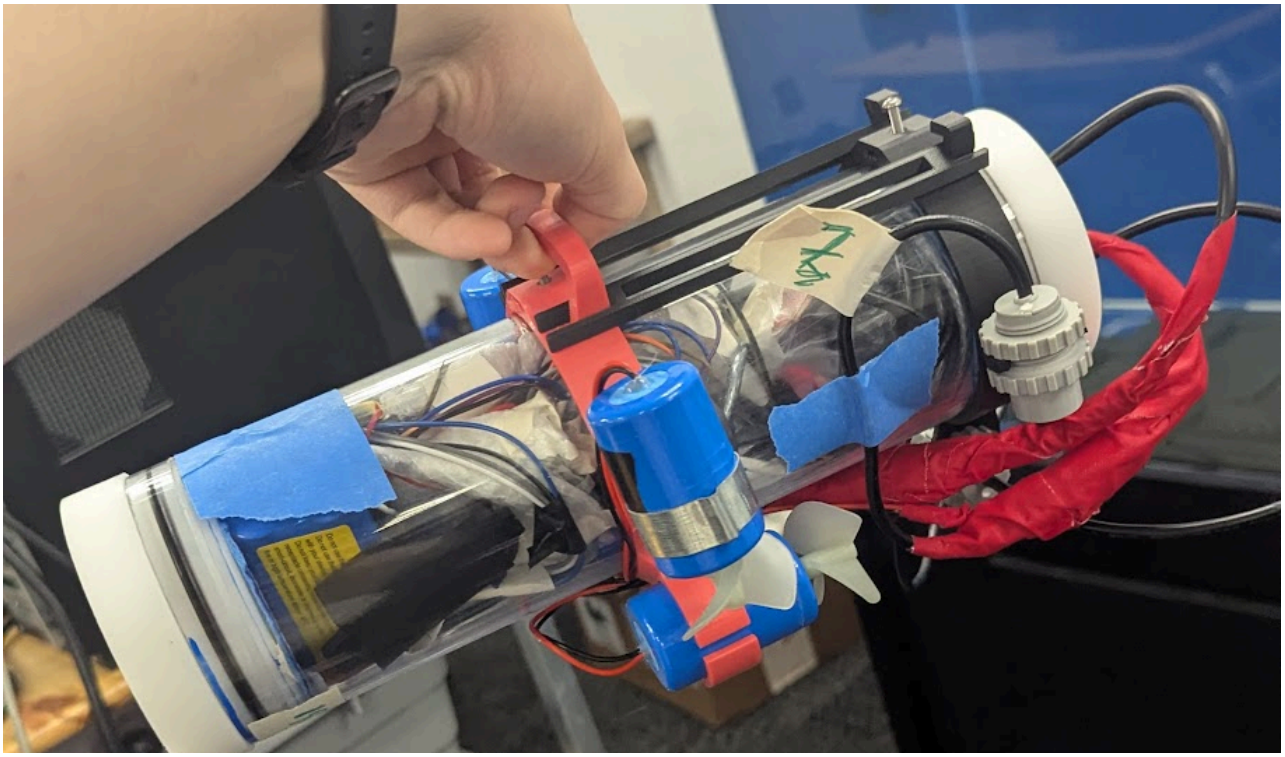

Figure 15: Prototype with sensor mount and aligner rods.

The internal housing required calculations to ensure a proper fit of all materials, understanding weight, and ensuring balanced torque and force. This went through a couple iterations due to a constantly changing design as the needs and parameters changed from the software design.

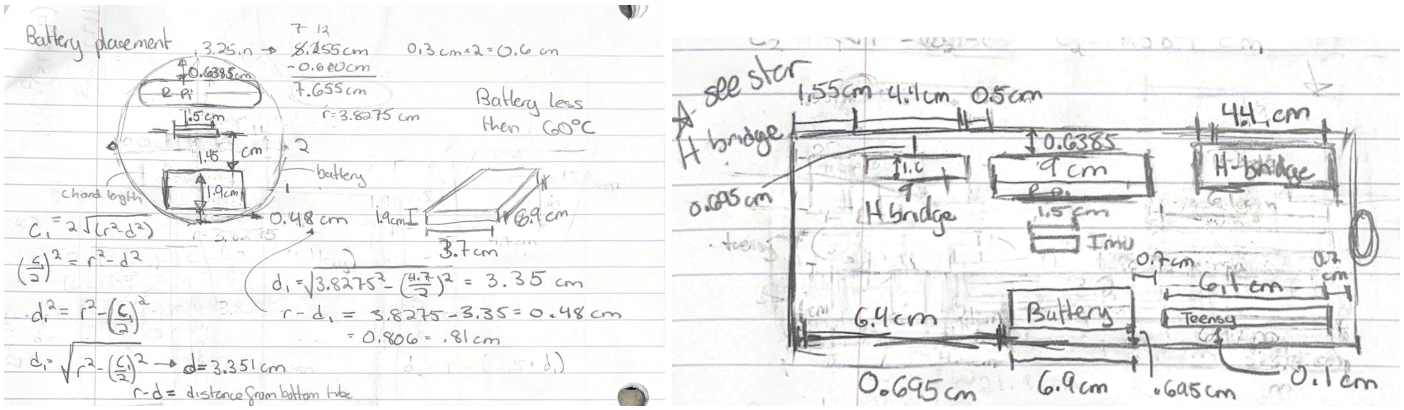

Figure 16A-B: First internal design calculations

The first iteration of the internal housing featured entirely magnetic attachments for a modular, 6-part unit. However, we found that the magnets were not strong enough to pull out the rest of the electronics if pulling from one end, as the internals were also friction fit to the inner diameter of the tube to ensure there is no stray movement while the submarine is in operation. It also did not really account for wire organization, and was subsequently scrapped in favor of an updated design.

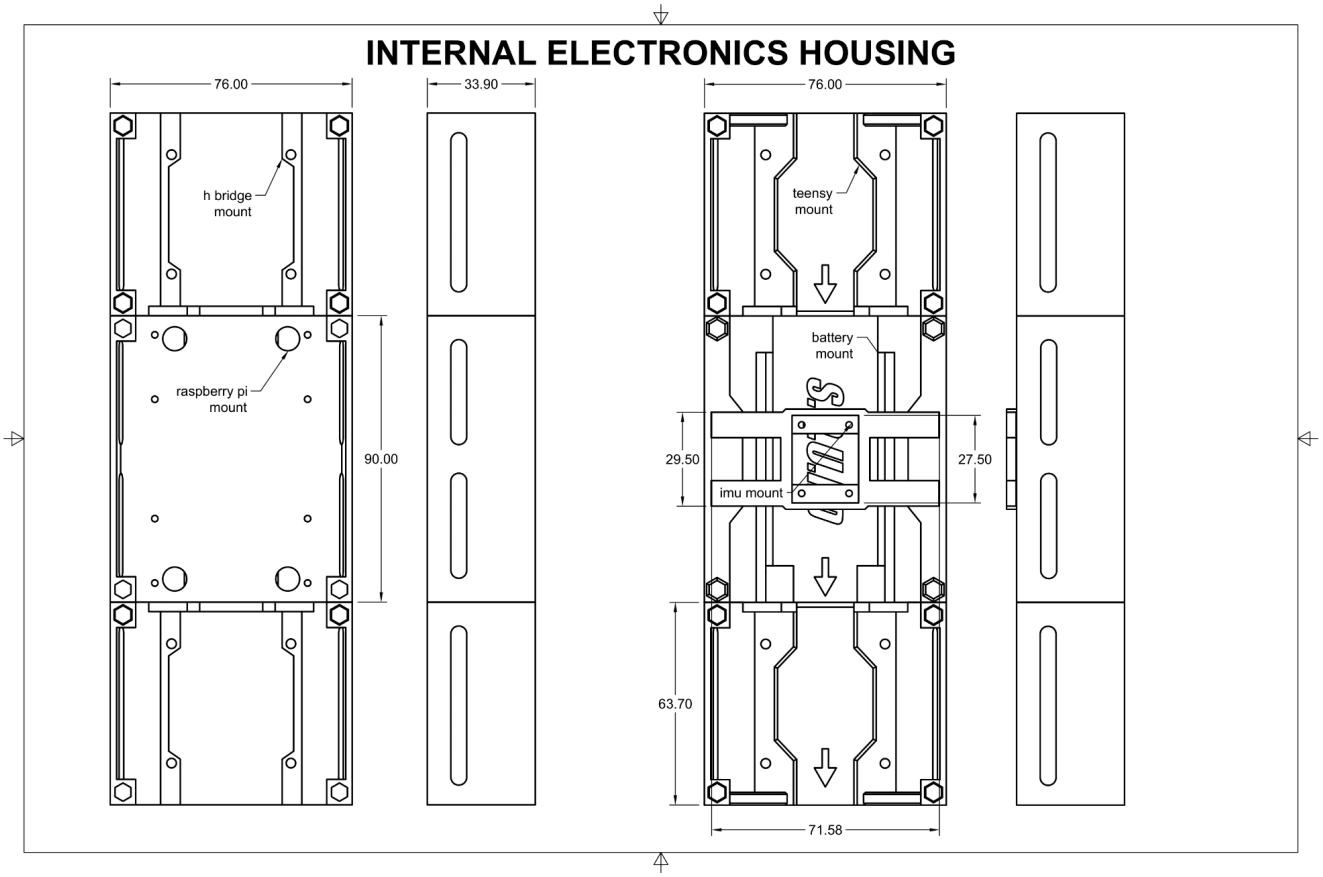

Figure 17: Internal electronics housing version 1.0 working drawing.

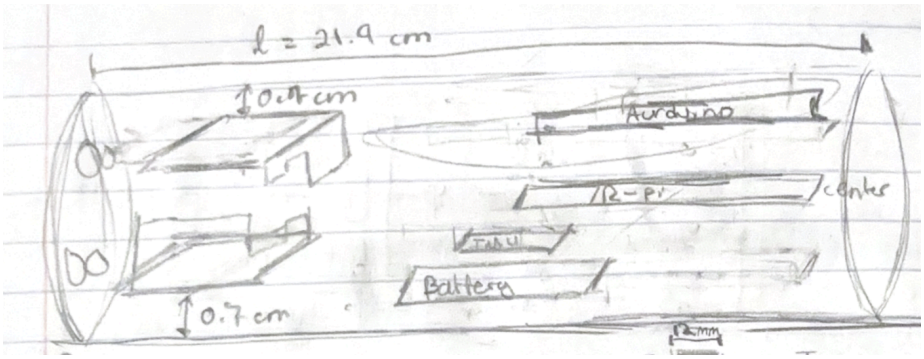

Figure 18: Final calculated design for the placement of housing, followed albeit upside down.

The following design, outlined above and in the Design Overview, was designed with connectivity in mind. The team realized that the current tube length was too short to fit all of the electronics needed, so an expanded tube was manufactured , bringing the total length of the tube up from 10” to 13”. The design was split into three primary modules, featuring dovetail connectors for a stronger interlocking system that could easily combat the friction from the inner tube during disassembly. However, we noticed that the wiring had to be threaded through the inside of each mount, as they did not come apart in two degrees of freedom like the prior model. Instead, they were enclosed hollow cylinders which made initial installation a bit more difficult, but was ultimately more convenient and tidy during subsequent assemblies.

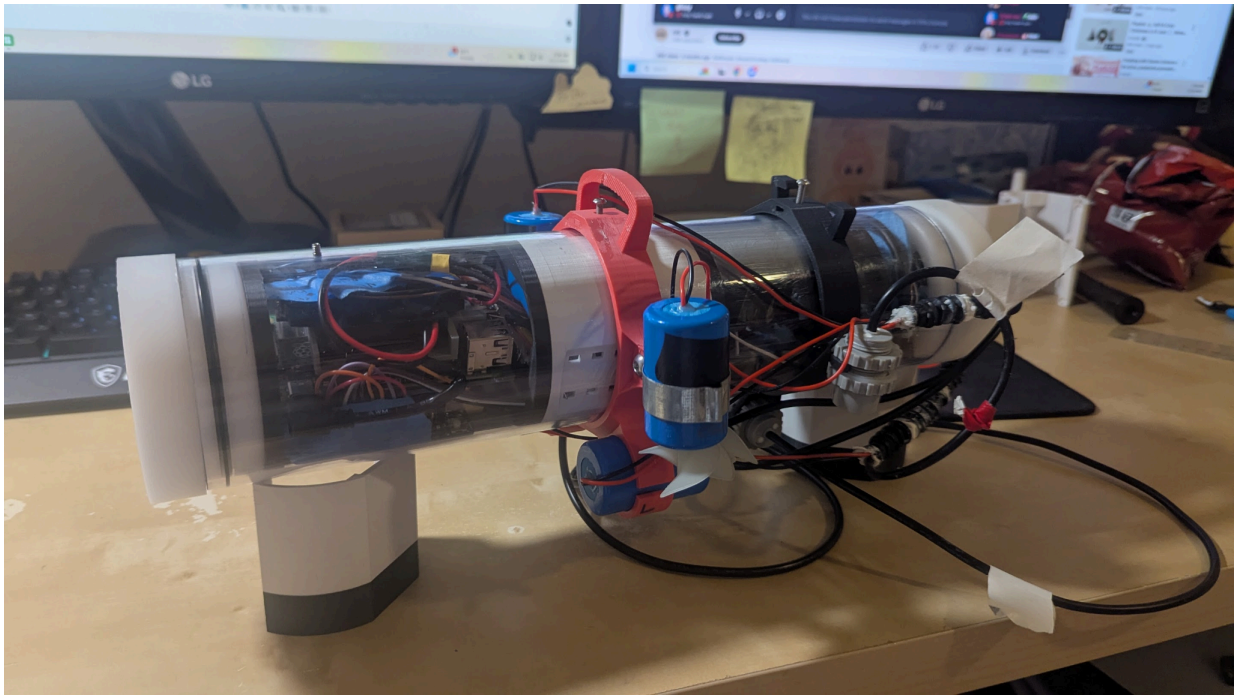

Figure 19: Expanded tube with installed electronics.

After initial tests of this latest iteration, the team found that the submarine was still too light. To combat this, we designed a makeshift weight holder attached to the hull of the submarine, and calculated the total weight necessary to achieve neutral buoyancy. It was found this weight to be 20-22oz (570g). A newer, more efficient design would subsequently be designed and 3D printed later for the final iteration.

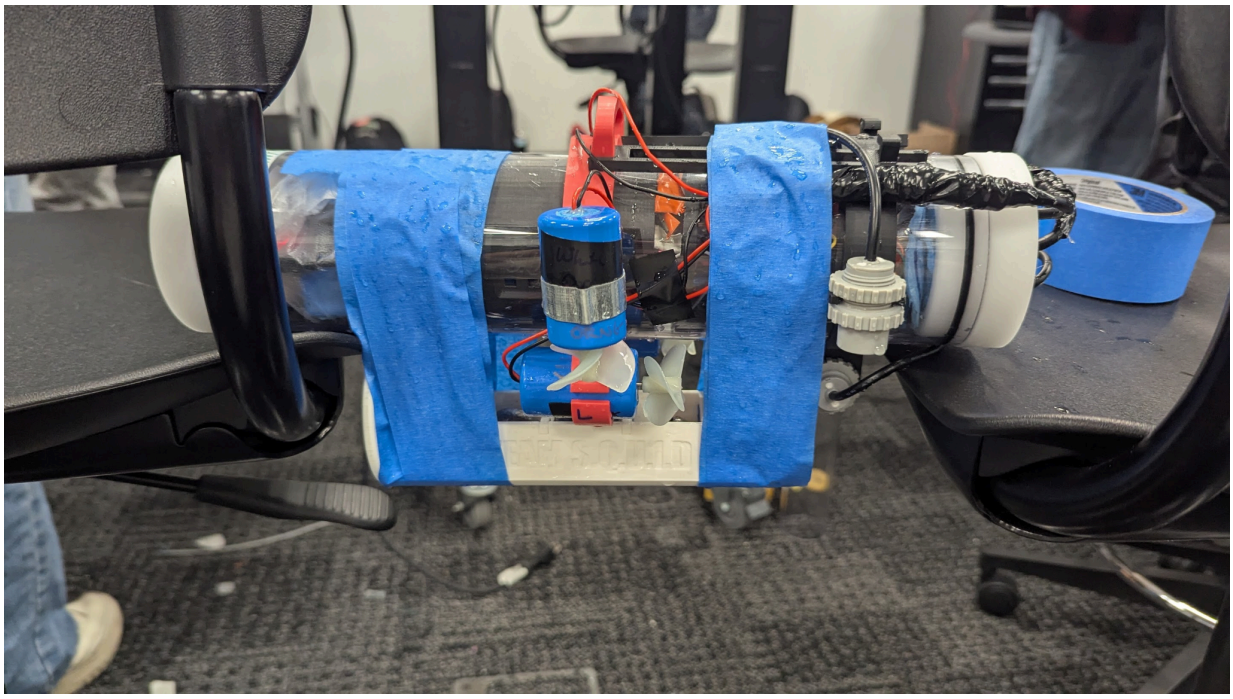

Figure 20: Submarine with makeshift weight holder attached to hull.

By now there were *still* issues with water ingress, so epoxy was then used for the wire glands so that all of the gaps would be filled as well as no longer needing to rely on hot glue. This did not work. It actually introduced *more* flooding into the submarine compartment. To mitigate this, 1:1 part silicone was poured into the inner section of the back gland cap. This effectively neutralized any water ingress. Later on, this ended up impacting the weight calculations of the vehicle, so in order to achieve neutral buoyancy, the extra weights were recalculated to be roughly ~30oz (836g).

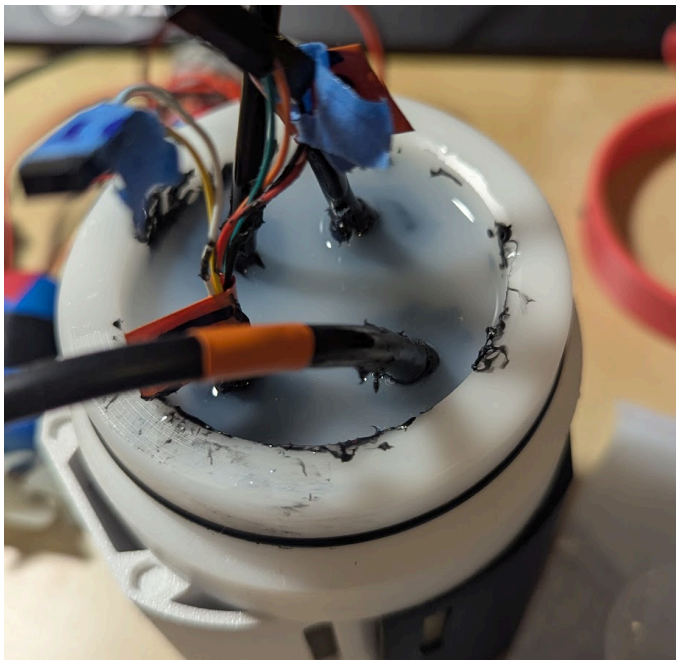

Figure 21: Silicone poured into the gland cap to prevent water ingress.

Showcased below is every saved iteration of the submarine's 3D-printed parts.

Figure 22: All iterations of the submarine's printed parts.

Software Overview

Team SQUID's code base can be found at https://github.com/omadams721/Team_SQUID_Submarine_Codebase.

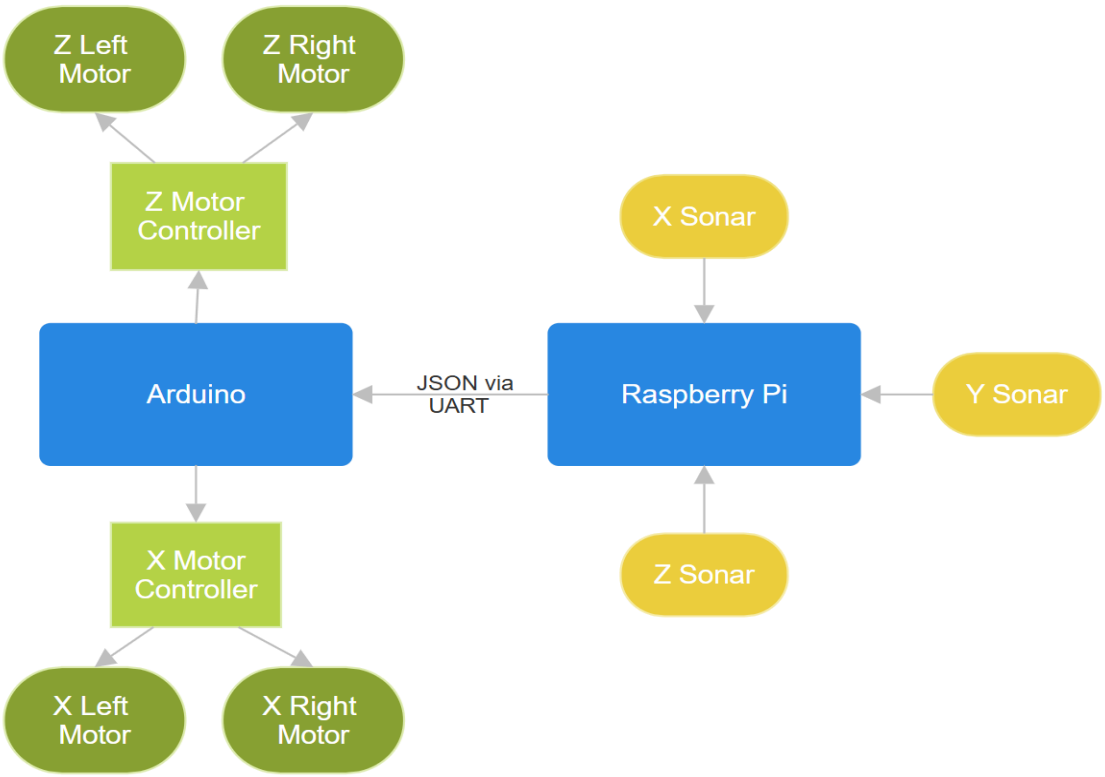

Figure 23: Data flow diagram

The software architecture of the submersible vehicle is divided between a Raspberry Pi and an Arduino, with each platform running code tailored to its strengths. This separation allows high-level sensing and control logic to be implemented on the Raspberry Pi, while time-critical actuation and hardware interfacing are handled by the Arduino. Together, these components form a modular and robust control system.

On the Raspberry Pi, three primary Python programs are used. The **keyboard teleoperation program** provides a manual control interface that captures single-key inputs from the terminal and packages them into JSON-formatted commands. These commands are sent over a serial connection to the Arduino and are used primarily during testing and debugging to directly control the vehicle's motion. This program enables rapid verification of drivetrain behavior and communication reliability without relying on autonomous control logic.

The **autonomous depth control program** implements closed-loop depth regulation using feedback from underwater ultrasonic sensors. This script interfaces directly with multiple sonar

modules over hardware UART, decodes the raw binary distance measurements, and designates one sonar as the primary depth reference. Based on the measured depth, the program determines whether the vehicle should dive, surface, or maintain its current depth and send the corresponding command to the Arduino. A hysteresis band is used to prevent rapid command switching caused by sensor noise, resulting in more stable and predictable depth behavior.

The **sonar data logging program** is dedicated to sensor validation and calibration. It continuously reads distance measurements from all connected sonar sensors, timestamps each reading, and writes the data to a log file. This program allows the team to analyze sonar performance offline, identify noise or dropout conditions, and tune control parameters independently of vehicle motion. Separating data logging from control logic reduces risk during testing and simplifies debugging.

Complementing the Raspberry Pi software, the **Arduino firmware** is responsible for low-level motor control and command execution. The Arduino listens continuously for JSON-formatted commands sent by the Raspberry Pi over serial communication. Incoming data is buffered until a complete message is received, parsed using the ArduinoJson library, and then translated into motor actions. The firmware controls two independent drivetrain subsystems corresponding to different axes of motion, enabling both horizontal movement and vertical depth adjustments. By handling deterministic tasks such as PWM generation and motor direction control on the Arduino, the system achieves reliable and predictable actuation even if the Raspberry Pi experiences timing delays.

Software Choices

One intrinsic and somewhat unconventional element of the design is the use of both a Raspberry Pi and an Arduino. This decision was motivated by a combination of technical considerations and team-level logistical factors.

A dual-processor architecture was selected to balance computational capability with real-time reliability. The Raspberry Pi serves as the high-level controller, responsible for sensor data processing, decision-making, data logging, and user interaction, while the Arduino is dedicated to low-level, time-critical motor control and direct hardware interfacing. This separation is important because the Raspberry Pi operates on a non–real-time operating system, which can introduce unpredictable timing delays that are undesirable for precise motor control and safety-critical actuation, particularly in an underwater environment. By offloading deterministic tasks such as PWM generation and immediate actuator response to the Arduino, the system achieves more predictable and stable control behavior. At the same time, the Raspberry Pi provides the flexibility and processing power required for autonomous behaviors and multi-sensor integration.

In addition to the technical benefits, this architecture also supported effective team collaboration. The team consists of both computer engineering and electrical engineering students, and it was important that the codebase remain accessible so that all members could contribute meaningfully and learn from the project. Since all team members had prior experience with Arduino development and C/C++, the Arduino provided a familiar and approachable platform for implementing and testing low-level functionality. This allowed individual components such as motors and sonar sensors to be developed and validated independently, without requiring the full Raspberry Pi system to be configured. Given that the initial Raspberry Pi setup required multiple class periods, this parallel development significantly improved productivity.

Overall, this distributed architecture improves system robustness, simplifies debugging and future expansion, and reflects design practices commonly used in real-world robotic systems.

Lessons Learned

Leaking

The most important thing to consider first and foremost is that all of the materials supplied by Professor Gomez are a **suggestion**. He does not require teams to do anything and actually recommends each team to find their own innovative way to solve the problems at hand. It is recommended that future teams do this too because 90% of the parts that Gomez provides are **not meant for underwater applications**. The machined gland caps with O-ring fittings work fine, but the wire glands that are meant to be installed onto a gland cap to minimize water ingress do not work well. For replacement parts, it would be best to (very early on in the semester) seek the counsel of Jay or Bryan Quinn from the Electronics Shop in A.V. Williams. In order to minimize water ingress, the team wanted to seal up any spaces water could seep through between the gland cap holes and the wire glands. The original idea was to just cover it in hot glue, and while this serves as a quick and easy solution, it is only temporary. The submarine could last around 5~ dunks before water would get past the cracks and flood the chamber. By putting sponges on either end of the tube, it slowed down the flooding, but a pattern quickly emerged to indicate water mainly flowed in from the wire glands side (if the design has glands on both caps, good luck). Hot glue is a good temporary solution since it allows some time before the submarine needs to dry out the sponges, but later learned that it does not flex well enough like other polymers would (such as silicone). If any thick wires coming out of the glands were to shift, the hot glue would give way and allow water to pass through. So we chose to move to a different polymer: epoxy. Do not do this. It doesn't work. It only works in theory if it is executed correctly. Epoxy requires a steady hand to apply, so if it gets messy a complete seal is not formed and air bubbles can be found which, unsurprisingly, would allow water to seep through. If future teams decide to go through with epoxy, use epoxy resin, not epoxy glue. Epoxy takes a long time to cure which is not the best idea especially while on a time crunch. To mitigate the drawbacks of the epoxy installation, we then poured 2-part silicone into the gland cap to fill in any holes the epoxy left. This worked like a charm and allowed for minimal water ingress. However, the silicone introduced unforeseen weight factors that required us to recalculate the weights to maintain neutral buoyancy and level weight distribution. To avoid all of this headache, future teams may want to use RTV sealant or aquarium silicone off the bat (buy it immediately, the Electronics Shop does not have it). Overall, it is paramount that this is one of the problems future teams should solve first before electronics installation, as that will save future teams many headaches (and fried electronics) in the future.

Sonars

We decided to use three underwater ultrasonic obstacle avoidance sensors (sonars) (SKU:sen0598). The initial proof of concept was simple, one sonar was connected to the arduino using a guide from DFRobots. It was proven that the arduino is able to read serial output from the sonar and that the sonar is able to properly detect distance. The problem arose when the sonars needed to be integrated into the existing system. While the arduino has multiple serial ports, it was not able to synchronously read from three different serial ports. This meant that only one of the three sonars would properly transmit data, though all were connected and properly wired. The sonars ended up needing to be placed onto the Raspberry pi. On the Pi, the sonars were connected to UART compatible pins (AMA3, AMA4, AMA5) using the Raspberry Pi Pin Chart (https://pinout.xyz/). Once transferred to the Pi, the system was able to reliably synchronously receive data from all sonars.

Through intensive testing and experimentation, it was discovered that the sonars would only be able to properly collect distance measurements if they were pointed perfectly perpendicular to the object they are measuring. This means that **if they are skewed or measured from an angle the reading will be None**. This created two distinct problems. First, the sonars must be mounted precisely on the submarine in a secure manner so that it is parallel to the axis it's measuring. There is very little room for error so it was essential the mounts were designed specifically for the sonars. The second and larger problem occurs when the submarine is in the tank. If the submarine is not perfectly orthogonal to all walls of the tank, the location is not transmitted. This means that if the vessel is not neutrally buoyant and balanced resulting in a forward or backward tilt, the depth can no longer be read. If the submarine is turning or unintentionally veers from a straight path, both its forward and side distances will be lost. Thus, it is essential that the submarine is designed perfectly balanced to prevent any pitching or veering when moving forward/backward. If Team SQUID had more time, the next step would be to implement a control system that kept the submarine moving straight and adjusted for any veering from human operation error or imbalances in the vessel. This could be done by using and IMU and monitoring its gyroscope data

It's also helpful to note that they are only able to measure distances underwater, if tried above water the reading is 0.0 – this made testing a bit of an annoyance. After trial and error, the most efficient way to test was to take an unused tube meant for designing the submarine, plugging one end with the caps machined with an O-ring and filling it with water. This meant that the sonars were able to be tested from the work space and there was no need to transport all of the electronics to the tank and risk water damage.

SSH

In order to wireless communicate with the submarine, an SSH connection had to be set up to the raspberry pi. The goal was to have a SSH client running on a personal laptop that could connect to the server on the Pi. From there any program could be executed through terminal commands and make minor edits to any code. After many failed attempts to set up the system, it was found that **Eduroam blocks any sort of SSH connections**. Instead a classmate, Wyatt Geckle, on Team Tankgate had set up his own router that all teams were able to use without any difficulties. Another solution that other teams employed was to set up hotspots on their phones, but it seemed that the strength of the hotspot varied by phone.

During tele-op testing, there seemed to be intermittent loss of connection. In a test, motor commands like “w” or “s” would be sent, but there was no response from the vehicle. This was also noticed during times where the system would stop receiving distance reports from the sonars. After talking with the other teams, the consensus is that **SSH connections are unreliable when the vehicle is in water**, the deeper the water, the worse the connection.

Port Calculations and Innard Design

Measure twice, cut once are words to live by. When designing the housing and placement of each electronic part, the ports for connection cannot be forgotten. Calculations were done to optimize the space, using the equation of the chord of a circle to find the lowest and highest possible placement of the vehicle. Unfortunately, **the port to power the raspberry pi is on the side.** Since the diameter of the tube was only 3.25 inches, this meant a USB that connected at a 90 degree angle was needed. The port connection protruded about half an inch. Adding the width of the raspberry pi to the protrusion of the port meant it could only barely fit in the exact middle of the tube. **Additionally, wires, especially messy wires, take up a lot of space.** These wires are also delicate, so cramming them into the sub will result in loose connections or possible complete disconnections.

Technical Analysis

Manufactured system

The vehicle utilizes a variety of provided electronic components that are assembled with a proprietary mechanical system. The majority of these components are 3D-printed, and designed so that all parts fit together without much need for additional adhesive material. The internal housing slots in nicely into the tube through which the wires travel through the wire glands and each subsequent component

attaches to its respective external mount. There are a variety of jumper cables that allow for modular connection between each subsystem and easy assembly.

Propulsion system

We have a total of four thrusters. Each thruster has an input voltage of 7.4 V. The rpm rating is 16800 rpm or 580*pi rad/s. The propeller size is 1.7 cm. The forward velocity of the thrusters is 0.201 m/s. The max input current is about 2.4 amps, but with the connection to the H-bridge, the max current drops to 2 A. The minimum current at full power for the thrusters is 1.3 A. The total torque of the motor is 0.00503 N*m. The mass flow rate of the thrusters is estimated to be 0.2815 kg/s. Additionally The thrust force is about 0.08725 N*m. A typical brushless motor, attached to the propeller, has a motor efficiency of about 87.5%. The drag force of the vehicle, which opposes the motors, is 0.296 N. The total force, thrust force minus drag force, then equals 0.0822 N*m. Dividing the system by its mass then gives us the acceleration. The acceleration is 0.104 m/s^2

Localization system

We have three Underwater Ultrasonic Obstacle Avoidance Sensors (referred to as sonars from now on) mounted on the exterior of the sub, for each X, Y, Z axis. Since each sensor requires their own dedicated hardware serial port, the team decided to connect them directly to the Raspberry Pi since it had enough UART ports to accommodate. The Pi is able to read serial information from them using AMA3, AMA4, AMA5 ports. The python program in the appendix (sonar.py), is executed while the submarine is in the water and allows us to read the submarine's distance from each respected wall. The distances are printed to the terminal as a list [x_distance, y_distance, z_distance].

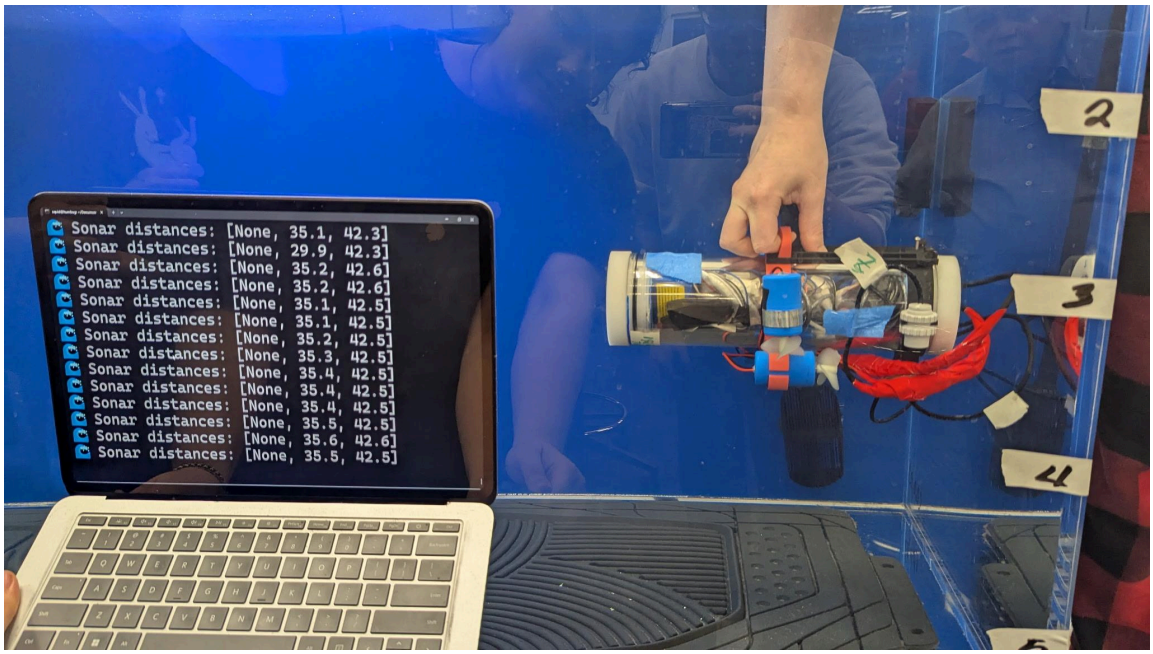

Figure 24: Demonstration of depth collection method.

Power system

The battery supplies power to the entire sub system. It is a 7.4 V battery. Its capacity is 2 Ah, and has a discharge rate of 3C. This means that the max current drawn on the system is 6 A. The battery would last for 20 minutes on the max current. It can supply enough current to power the raspberry pi, two thrusters, and all other components. This means the four thrusters cannot run at the same time. There must be a delay put in place so the battery is not overdrawn. It connects directly to the arduino, but a DC-DC converter must be used for the Raspberry Pi, as only a 5 V input is allowed.

Design Validation

Propulsion system

To ensure that the propulsion system could overcome the drag and its weight to move effectively, especially with the added weights slowing down the vehicle, a test was conducted to find the speed and the acceleration over time. The resulting graph is shown below. As the graph shows, it takes about 4 seconds to reach the max velocity of the vehicle. It also shows velocity is non-linear, but it can in fact overcome the total force (calculated to be Fd = 0.041 N, and measured force is 0.031 = 0.082 N) on the vehicle.

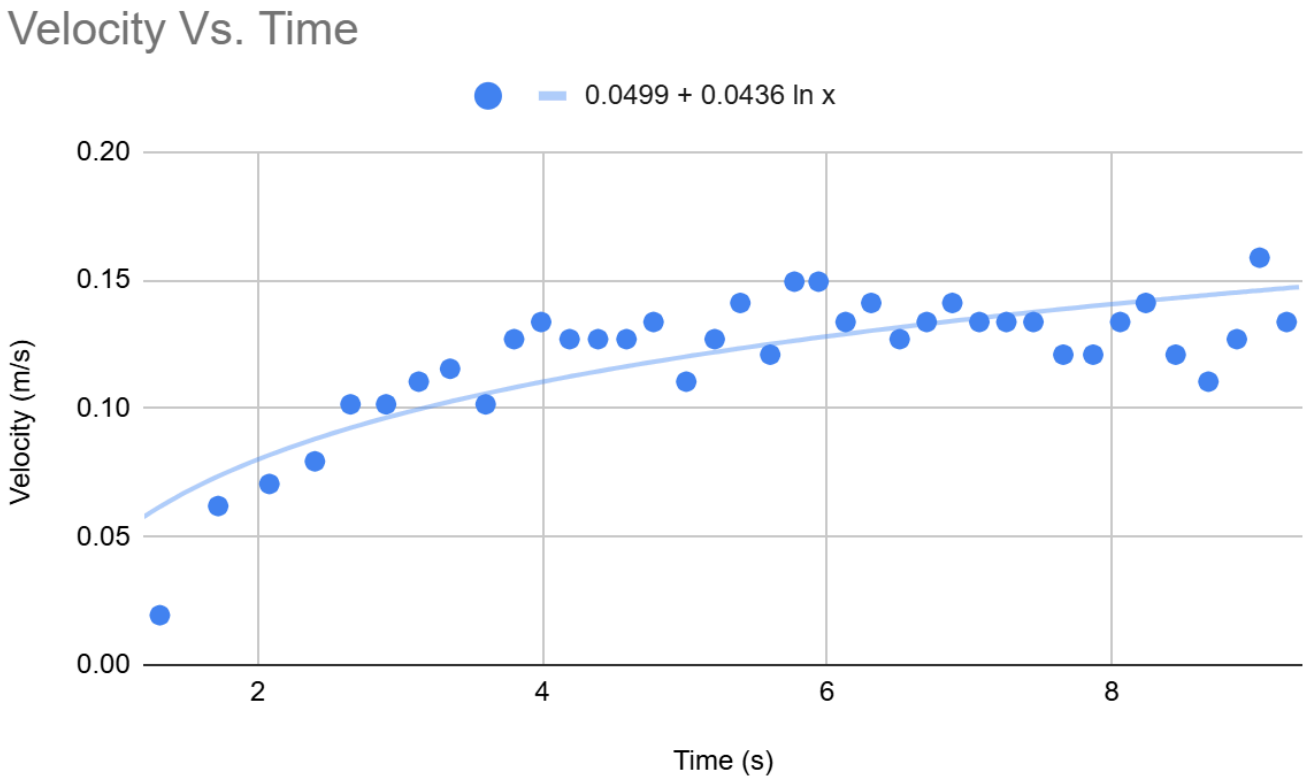

Figure 25: Velocity vs. time graph.

Control system

A proportional controller was implemented in order to achieve depth regulation. The Raspberry Pi would receive sonar data corresponding to the vehicle depth, determine if the depth was below, in between, or above the range of 35-45 cm, and send commands to the Arduino-Motor subsystem to surface, stay put, or dive, respectively. Using a +/- hysteresis to reduce noise and irregular behavior, this control system proved to be robust enough to maintain depth and complete the milestone. While relatively simple enough to implement, further precision could have been achieved with PID control.

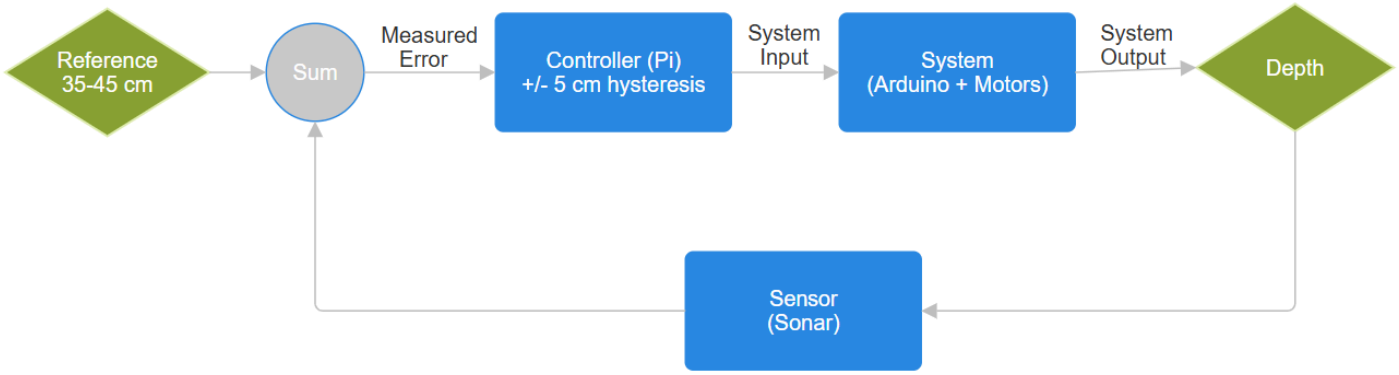

Figure 26: Control feedback diagram.

Localization system

In order to verify the accuracy of the sonars, the system was first placed in the tank, held the submarine 7 distinct depths and collected sonar readings of the Z axis at each stage. Note that each depth distance refers to the number of centimeters above the floor of the tank. The results are plotted below.

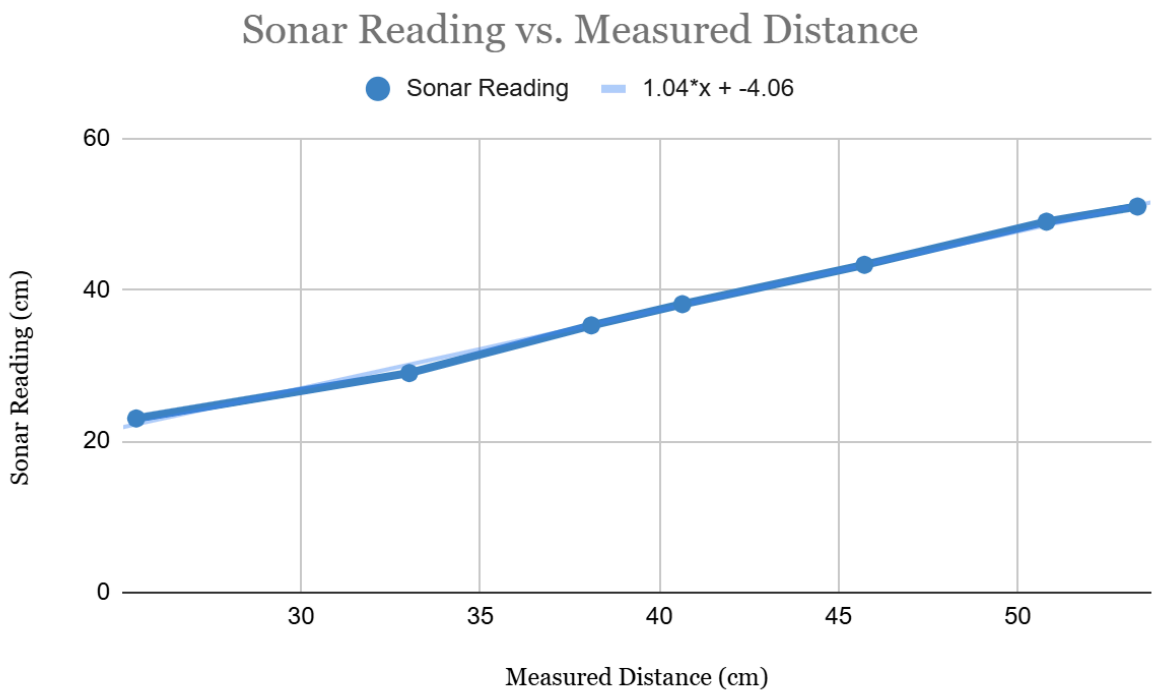

Figure 27: Graph of sonar distance reading at different depths.

From the testing, the conclusion is that the sonars can provide us accurate and reliable readings. The graph shows a linear relationship between the depth of the submarine and the sonar value; this indicates that the sensor is giving consistent readings regardless of the distance. There are no weird spikes/dips for any distances. There is a consistent 4 cm downward shift from the measure height to the sonar reading. This can be attributed to the fact that the sonar is mounted on the bottom of the vessel while the measured height is in reference to the middle. In the future this will be accounted for in the software by adding a 4 cm offset to the raw data before delivering to the terminal and the control system.

System-level validation

Solely verifying that components work does not prove that the end goals of the project were achieved. Testing the various functionality required for each milestone involved individually troubleshooting motors and sensors to do exactly what they were supposed to do, especially before installing these components on their designated mechanical mounts.

In order to reach milestone 1, all 4 motors needed a controller (L298N H-bridge) to change the motor direction. In order to test forward and backward motion, the Arduino code first run used the hotkeys (W, A, S, D) to test (go forward, turn left, go back, turn right, respectively). Then, to ensure Raspberry Pi-Arduino connection, the same test was conducted by running the Raspberry Pi equivalent code.

This would allow us to send the motion commands via SSH to the Raspberry Pi. After this verification, an experiment then tested the SSH capacity of connecting from the laptop to the submerged robot and, by extension, the full functionality for milestone 1 - Team SQUID submerged the vehicle with the motors mounted, ran the Raspberry Pi code for forward and backward motion.

Reaching milestone 2 involved a similar troubleshooting process. After connecting the Sonars to the GPIO pins on the Raspberry Pi, test code was run to verify that the Raspberry Pi could read data from the corresponding UART ports and communicate an [X, Y, Z] coordinate ([0.0, 0.0, 0.0] when the sonars were not in water). Then, by dipping the sonars in a tube of water, it was verified that the sonars could read different depth levels. Finally, a test was conducted for full milestone 2 capacity by submerging the vehicle in water at different depths and reporting a correct depth level (see Localization system).

II. Conclusion

Throughout the semester, Team SQUID has reached important milestones such as autonomous depth control and a consistently working propulsion and sensor system. Unfortunately, full autonomy was not reached. Additionally, no camera module was ever introduced to the sub system. The iterations of the waterproofing proved challenging, as the previous teams noted. Still, the submarine design process allowed all team members to learn about an authentic design experience.
Hopefully, the lessons learned, code, calculations, and cad drawings can help future students design an even better submarine that will reach full autonomy and navigation. It's honestly the coolest part of the design process, that the successes of this iteration may live on to help other students.

III. Appendix

Code base: https://github.com/omadams721/Team_SQUID_Submarine_Codebase
Youtube videos: https://youtube.com/

IV. Team Member Contributions

Olivia Adams

- Developed all software
- Conducted testing for motor actuation, distance detection, wireless communication, wireless control, and depth regulation
- Assembled components and electrical wiring, including soldering, crimping, and wire management

Marco Albano

- Verified wireless communication via SSH
- Helped conducted testing for motors, distance detection, depth regulation
- Electronics assembly, including soldering, crimping, and wire management

Marg Gouker

- Gathered all components for the submarine as the team needed, including safety equipment
- Did all calculations to ensure battery could support electrical system, force and torque stability, buoyancy, and calculated optimal internal sub placement
- Designed and drew electronic schematics
- Soldered, crimped, and epoxied wires

Brian Wu

- 3D modeled and printed all of vehicle internal and external components
    - 88 hours, 31 minutes spent printing iterations
    - 6+ hours per week outside of lab dedicated to designing, fabricating, and prototyping the submarine
    - 2836 grams worth of filament (not including failures)
- Operated machine lathes, drill press, milling machine to create machined plastic pieces and tubing
- Created pressure valve hole for easier disassembly
- Soldered numerous wires and connections to boards
- Designed software simulation for IMU and sonar system collaboration
- Created milestone presentations and other graphics