ECE 403 - Senior Project Report
Submersible Drone for Hull Inspections

Dyllon Dunton (CE/EE)
Jacob Wildes (CE)
University of Maine Orono

May 1, 2024

Abstract

The control, power, and propulsion systems for a submersible drone were designed, built, and tested in this project. The purpose of this submersible drone was to inspect the hulls of ships. The submarine streamed a live camera feed to the controller on shore, where images are stored to a USB drive. The submarine and its controller were powered with rechargeable power supplies and operated for at least one hour. In addition, the controller and submarine communicated via Transmission Control Protocol and User Datagram Protocol (TCP/UDP) over ethernet using the Robot Operating System (ROS2). Finally, the specifications were evaluated to make sure that the ROV would satisfy the minimum deployment time and display frame rate, maximum recharge time, and meet the power supply requirements.

Contents

List of Figures

List of Tables

1 Introduction

This report discusses the design and testing of a submersible drone. The submarine has the capability to traverse any body of freshwater with control over its three-dimensional position and rotation. The system is easily piloted with a handheld remote that reports data in addition to a controllable camera feed. Additionally, the submarine utilizes a sensor to monitor its angle and keep itself upright.

The entire design process of the submersible drone is discussed. In Section [2], the project's block diagrams are presented and explained at a high level. Section [3] discusses the intricacies of the design process, including the programming process, IT development, and circuit diagrams. Final results are reported in Section [4] using measurements that prove the project's specifications. Lastly, the report is concluded in Section [5].

1.1 Project Purpose

Maintaining the hull of a ship is a costly venture, and often requires the use of large equipment to either hoist the ship out of the water to dry dock, or divers are needed to go under the ships in order to assess the hull [10][11]. The currently offered solutions are either not built for the task, or are built to go on deep dives, being equipped for the harshest of marine environments. As a result of this, those Aquatic Underwater Vehicles (AUV) are prohibitively expensive. This has necessitated a remotely controlled submersible that is equipped specifically for surveillance. Since the project is built to survey, it will do well in hull inspections, but it also allows the drone to be multi functional, serving as an assessment tool for other applications, such as aquatic ecosystem exploration.

1.2 Project Description

At surface level, the project consists of two parts; the controller and the submarine. These two subsystems are interlinked with a tether that allows for bidirectional communication. The controller acts as the human interface to the system as a whole. From here the operator is presented relevant piloting information and given control over the movement of the ROV. The Submarine has the ability to take in data from the controller and decide the best course of action to meet the operator's request. Then, data is relayed back to the controller to be viewed by the operator. Together, these two subsystems make up the entire system.

The design of the submersible drone can be split into four parts. First, each subsystem had a data acquisition phase, where they need to learn about their environments. Second was the network which allows both subsystems to communicate simultaneously. Third was the power systems that allow both the controller and submarine to function. Lastly, the motor driver circuits that control all of the motion on the submarine. All of these systems together provide means for the project to meet the outline specifications.

1.3 Project Specifications

The project specifications were the guiding factors in the design phase of the submersible drone. The specifications section from the contract is provided below.

- Power Supply with output voltage of 12 V ±5% with 100 mV ripple at 1 A load current
- Recharge time of less than 4 hours
- Minimum of one-hour long deployment time
- Custom Printed Circuit Board (PCB) for the controller
- Minimum frame rate of 15 FPS

The project specifications clearly state the important action items for the design procedure and laid out guidance for the next sections.

2 Breakdown

The submersible drone can be split into two primary block diagrams, a hardware and a software diagram. Details for each block are discussed in the subsequent subsections.

2.1 Hardware

Bringing together the major constraints with the requirements from the contract, block diagrams can be created for both hardware and software. For the hardware, the two major systems have been divided into their core subsystems. The hardware functional block diagram is shown below in Figure [1]. The two processing units communicate with each other and command the smaller subsystems. The submarine receives movement instructions and actuates the motors while relaying camera and sensor information to the processing unit on the controller. The controller's processing unit takes that sensor data and displays it on a 7-inch TFT LCD screen. If toggled, the currently viewed image is saved to a controller-side USB stick. The controller also transmits movement commands to the submarine. In addition, both the submarine and controller have their own power supplies for delivering power to their subsystems.

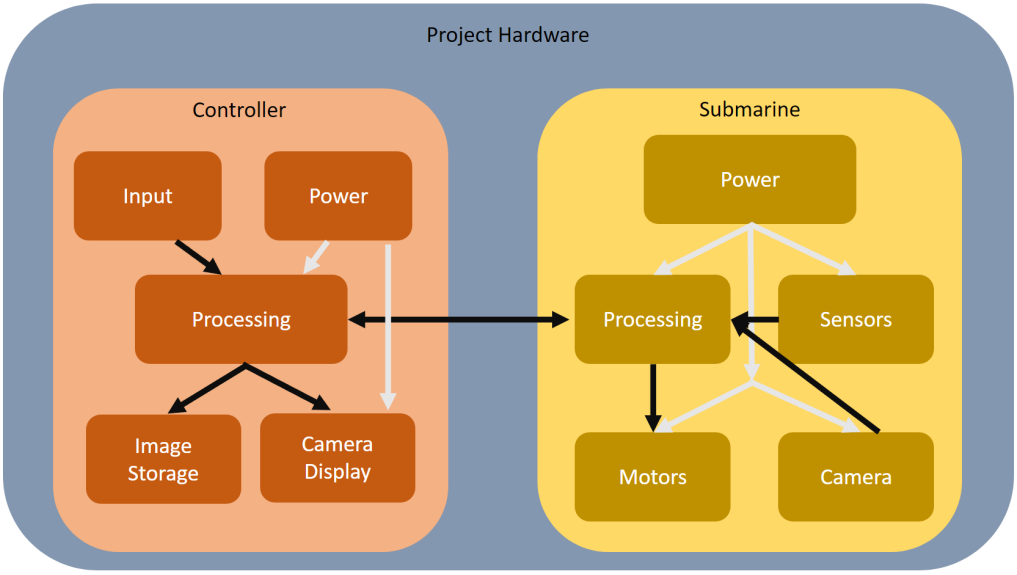


Figure 1: Hardware functional block diagram.

2.1.1 Controller Input Hardware

The Controller Input Hardware module consists of an STM32 L452RE-P which serves as a platform for all other methods of input. Additionally, there were two 2-axis joysticks, a switch, and a button. The uses of the components are discussed further in Section [2.2.2].

2.1.2 Controller Power

In order to run the controller's processing module, two ~~3.7 V~~ ICR 18650 lithium-ion (Li-ion) batteries were connected in series to provide 28.86 W of power. The power line is split two ways, one goes to processing, the other goes to the display. A variable voltage regulator steps the output voltage down from 7.4 V to 5 V. That power is then supplied to the RPi's 5 V rail. The power for the display is provided by the processing module.

2.1.3 Controller Processing, Image Storage, and Camera Display

The controller processing module consists of a RPi with 8 GB of RAM connected to the Controller Input Module via USB. All commands polled from the Controller Input module are sent over the network to the Submarine Processing module over Ethernet. The Image Storage and Camera Display modules are also combined in this module. The processing module sends 5 V via Universal Serial Bus (USB) to a 7-inch TFT LCD display. Finally, the Image Storage module is directly attached to the processing module, and its key function is to store requested images to a USB flash drive upon a button press.

2.1.4 Submarine Power

In order to power all of the other modules of the submarine, a power supply of at least 75 W was required. To satisfy this, four pairs of parallel connected 3.7 V Li-ion batteries were connected in series (4S2P) and constructed together into a battery bank with a power output of 115.44 W. The motor module required 12 V and the processing, sensors, and camera modules all required 5 V. To achieve this, two variable voltage regulators were used to step the voltage down to the required voltages for each module.

2.1.5 Submarine Processing and Camera

The submarine processing module consists of an RPi and a STM32 Nucleo L4R5ZI microcontroller that are connected via USB. This supplies both 5 V power to the microcontroller as well as serial communication which will be further discussed in Section [2.2.6]. The RPi is also connected via Ethernet to the controller processing unit [2.1.3]. The final connection to this unit is via I2C to the inertial measurement unit (IMU) [2.1.7].

2.1.6 Submarine Motors

This module encompasses all six motors, as well as their respective motor drivers. The submarine movement system is comprised of two servo motors, two DC motors, and two bipolar four-lead stepper motors. The servo motors allow for the movement of a camera gimbal to rotate both horizontally and vertically (yaw and pitch), and are controlled via Pulse Width Modulation (PWM) from the microcontroller [2.2.8]. The two DC motors are responsible for controlling forward motion and yaw. both motors are connected to propellers that can affect the translation force of their side of the submarine. These motors are actuated by a Dual H-Bridge integrated circuit (IC) which is controlled via four PWM pins from the microcontroller. Finally, the two stepper motors control the two ballast tank positions. The rotation force from the motors are converted to translation force via a lead screw which moves the ballast tank walls. This directly affects the buoyancy of either side of the submarine which controls vertical translation as well as roll. The wall position is measured [2.1.7] and fed back to the motor controller to form a closed loop control. Motor rotation is actuated by two Dual H-Bridge ICs which are directly controlled through General Purpose Input/Output (GPIO) pins from the microcontroller.

2.1.7 Submarine Sensors

The submarine sensor module consists of both the IMU as well as two linear variable resistors. First, the IMU measures acceleration on six degrees of freedom which covers all three dimensions in both translation and rotation. This sensor is one of the two feedback loops in the network that controls the motors. The two linear variable resistors measure the translation of the ballast tank walls. These form the second feed back loops for their respective stepper motors. These inputs are fed to the submarine processing unit [2.1.5] and used to plan motion [2.2.7].

2.2 Software

As shown in Figure [2], the software processes begin by initializing all the subsystems before entering their main cycles until shut down. Upon entry into this cycle, the joystick inputs from the controller are read and sent to the submarine's processing unit. Given this information, the submarine will determine its state and plan its next action to remain upright and moving the correct direction. Next, the submarine commands the ballast tanks' stepper motors, the propeller motors, and the camera swivel servo motors based on the received instructions from the controller. Then, the camera takes an image and the IMU is read. Following this, the submarine sends this information back to the controller, which is displayed to the operator. Finally, the operator has the option to save that image to a USB stick.

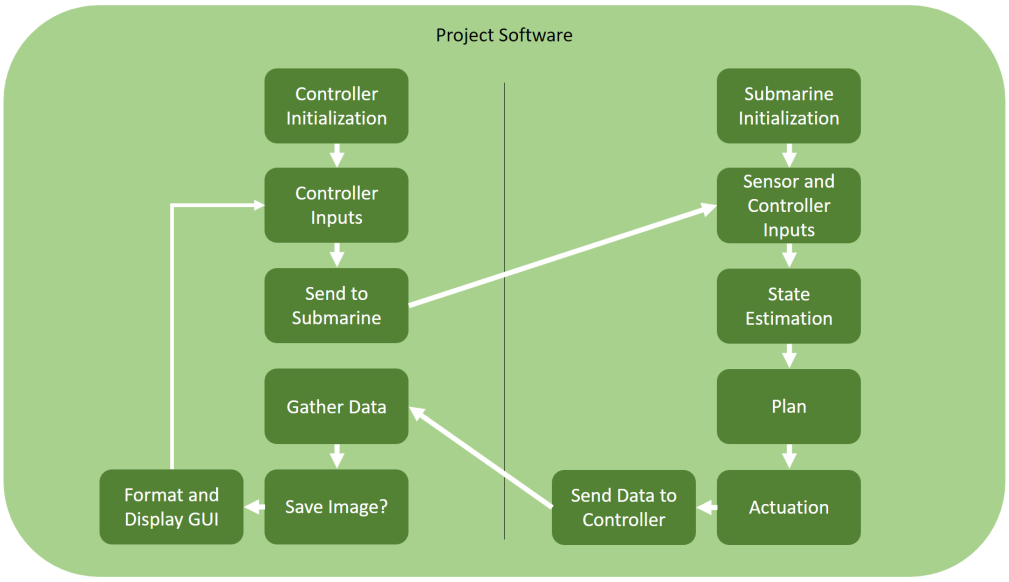


Figure 2: Software functional block diagram.

2.2.1 Controller Initialization

The controller initialization module show on the top left of the block diagram consists of all of the boot services required to launch the controller RPi. This entails a service file which runs once that runs immediately after boot, and applies a shell script that will continue to attempt to ping the submarine's RPi. If it cannot ping, it will continue to retry until the other RPi can be communicated with. This alleviates any deviance in boot times between the two systems. Once both systems are able to ping one another, a final service is called which properly sets environment variables to allow the controller graphical user interface (GUI) to appear on screen. Once these processes have successfully executed, the controller enters its main loop to communicate with the submarine.

2.2.2 Controller Inputs

The controller input module consists of an STM32 microcontroller which reads user inputs from two directional toggles. One controls the submarine's forward, backward, left, and right movements. The other controls the onboard camera's up, down, left, and right movements. Additionally, there is a switch which allows the user to control the depth. Finally, there is a button that allows the user to take a screenshot. The STM board continually sends updates to the RPi over USB.

2.2.3 Send Input to Submarine

This module consists of the RPi connected to the STM32 reading in user inputs. The RPi reads from the STM32 board at a rate of 20 Hz. It then publishes that data at the same rate over the ROS2 network to the submarine. Transmission Control Protocol (TCP) is used to facilitate unidirectional communication from controller to submarine. This ensures that the submarine is always in communication with the controller, and that no command is missed.

2.2.4 Format and Display GUI

This module is a combination of three smaller modules - gathering data and potentially saving an image. This module gathers all data published by the submarine over the ROS2 network and formats the submarine's heading, any error messages related to the submarine's functionality (such as a ballast failure), time running, and the level of water in the ballast tanks. It places this information in a GUI over the image from the sub-side camera. It is in this module where the input from the user dictates whether an image is saved. The loop returns to the controller inputs [2.2.2] module from here.

2.2.5 Submarine Initialization

The submarine initialization module consists of service and shell scripts which run at the submarine's startup that allow it to immediately begin querying data from the attached STM32 board and send data over the ROS2 network to the controller. One service file launches a shell file that will attempt to ping the controller RPi. If it cannot, it will continue to try until it can. Once communication has been established between the controller and submarine, the final service file is called, and the RPi begins to query data from the STM32 board and publish it over the network to the controller.

2.2.6 Sensor and Controller Inputs and State Estimation

This module of code is written in C and Python and serves to receive the IMU data and controller inputs from the RPi via serial over USB using information it received from the controller using ROS2. In addition the microcontroller uses Analog to Digital Converters (ADCs) to measure the position of the ballast tank walls as well as the battery voltage. This information was then compiled into a well organized data structure and sent to the planning module [2.2.7].

2.2.7 Plan

The planning module is written in C and receives information from the previous module and parses the inputs from the controller to understand exactly what the operator wants from the submarine. Next, the sensor input is used to form a two feedback loops that form a closed loop proportional control for the stepper motors in order to control depth and roll. Next, data structures are set up for the actuation module [2.2.8] that instruct the microcontroller on what its next step is and how to do it.

2.2.8 Actuation

The actuation module is written in C and receives instructions from the planning module and changes GPIO and PWM pins in order to interface with the motors drivers that actuate the six motors.

2.2.9 Send Data to Controller

The final module in the submarine code is for data transmission and is written in C and python. This sends the battery voltage, ballast tank levels, error messages, the upwards and forward acceleration, as well as the compass reading from the IMU. This is sent back to the RPi over serial, and then the RPi sends the entire data frame to the controller via UDP over ethernet using ROS2.

3 Details

This section provides a holistic discussion of the submersible drone project. First, the component selections will be justified with respect to the project specifications and project goals. Finally, each module of the project will be discussed in further detailed, accompanied by justification for each decision made.

3.1 Microcontroller Selection

The main processing units for the controller and the submarine are two RPis and two STM32 Nucleo Microcontrollers. These were picked to satisfy two main requirements: networking [3.1.1] and extensive hardware capabilities [3.1.2].

3.1.1 Networking

In order to establish full-duplex communication between the controller and the submarine, two Raspberry Pi 4Bs with 8 GB of RAM were used. Each one serves as the communicator for its

subsystem. In addition, a main goal for this project was to use ROS2 in order to facilitate this communication over Ethernet using TCP and UDP. TCP was chosen for commands travelling from controller to submarine to ensure that command packets were not lost. UDP was chosen because receiving frames late is more detrimental than simply dropping them. These requirements necessitated a microcontroller with the capabilities of running an Ubuntu 20.04 Operating System (OS). In addition, the RPi can be powered via 5 V over power pins and can communicate via I2C with the IMU. Lastly the 8GB variants were chosen because the controller needed to be able to communicate in addition to constructing and displaying a complex GUI. Lastly, it was important to have at least two USB 3.0 ports so that the camera and serial communications could be facilitated quickly. For these reasons, the Raspberry Pi 4B with 8 GB of RAM was chosen.

3.1.2 Hardware

In order to interface with the inputs and outputs of the controller and submarine, it was important to pick microcontrollers that had extensive hardware capabilities such as a large quantity of GPIO, PWM, and ADC pins, in addition to having a USB port that could be used to deliver power as well as communication over serial. In addition it was important that these microcontrollers were able to run without consuming considerable amounts of power. For these reasons, the STM32-L452RE-P was chosen to run the controller and the STM32-L4R5ZI was chosen for the submarine. The controller microcontroller is a mid-size low power Nucleo-64 variant. This was important because the controller did not need as many inputs and outputs (I/O) as the submarine and power waste was avoided. The submarine microcontroller was a Nucleo-144 larger low power variant capable of handling much more I/O.

3.2 Power Supply

This following sections provide a comprehensive analysis of the power supplies that were designed and built for the controller and the submarine.

3.2.1 Submarine Power Supply

The power supply introduced in Section [2.1.4] is designed to provide a power solution for the motors and processing equipment aboard the submarine. The systems both require different voltages to operate, necessitating two variable voltage regulators. Table [1] shows the power requirements of each system.

Table 1: Submarine Power Table

Device	Voltage (V)	Amps (A)	Power (W)	Quantity	Extended Power (W)
Raspberry Pi 4B	5	3	15	1	15
12 V Motor	12	.960	11.52	2	23.04
Stepper Motor	12	.350	4.2	2	8.4
Servo Motor	6	.530	3.18	2	6.36
Camera	5	.250	1.25	1	1.25
STM32 L4R5ZI	3.6	.550	2	1	2
IMU	3.3	.151	.5	1	.5
LED	12	.125	1.5	6	9

The extended power totals to 65.55 W. To ensure that the one hour minimum deployment time requirement was satisfied, extra padding of roughly 15% was added for a goal of 75 W. As discussed in Section [2.1.4], a 4S2P battery bank was built from 3.7 V battery cells. The four batteries in series satisfied the requirement of having 12 V for the motors. The 18650 Li-ion cells have a maximum discharge current of 3.9 A. Using the equation for wattage

$$P = I * V \tag{1}$$

The 4S configuration of the power bank only supplies 57.72 W, significantly lower than the target power. To increase the power output of the battery bank, a 4S2P configuration was adapted as described in Section [2.1.4]. This effectively doubled the power output, yielding 115.44 W of power - substantially more than the target power requirement. This ensured that any unforeseen usage pattern does not hinder the submarine's ability to operate for one hour. Figure [24] depicts the discharge time of the submarine.

3.2.2 Controller Power Supply

The power supply discussed in Section [2.1.2] is designed to provide a power solution for the systems on the controller. Table [2] depicts the power requirements for each system on the controller.

Table 2: Controller Power Table

Device	Voltage (V)	Current (A)	Power (W)	Quantity	Extended Power (W)
Raspberry Pi 4B	5	3	15	1	15
7" LCD TFT Display	5	.62	3.1	1	3.1
STM32-L452RE-P	3.6	.550	2	1	2

The total power requirement for the controller was 20.1 W. Like the submarine power supply, a padding of approximately 15% was applied, thereby making a target power of 23.115 W. To achieve the goal, two 18650 Li-ion cells were placed in series, yielding 7.4 V and 3.9 A. Those components gave the battery pack a power rating of 28.86 W - above the target power as well. Figure [24] depicts the discharge time of the controller.

3.2.3 Recharge Requirement

In addition to deployment time, there was also a specification for a maximum charge time of four hours for both the controller and submarine. To ensure that the time was met, a goal of recharging the submarine in three hours was set. Each individual 18650 cell has a maximum recharge rate of 2.6 A. The following equation was used to verify recharge rates:

$$I_{charge} = \frac{Capacity}{ChargeTime} \tag{2}$$

In the case of the submarine power supply there were two cells in parallel, giving each pair a charge capacity of 5.2 Ah. This yielded a required charge current of 1.73 A, far below the maximum charge current of 2.6 A.

In the case of the controller power supply the cells were only in series, so each cell had a charge capacity of 2.6 Ah, yielding a charge current of 867 mA, significantly below the maximum charge current of each cell. Figure [23] depicts the charge time. Before the battery packs were complete, a battery management system (BMS) had to be chosen for each pack, as discussed in Section [3.2.4].

3.2.4 Battery Management System Calculation

In order to properly balance the voltage of each of the 18650 cells, a BMS had to be chosen for both the controller and the submarine. For the submarine power pack, the target power consumption for each pack was used. This ensured that the charge draw would never exceed the capabilities of the BMS. For the submarine battery pack, the target power rating of 75 W was used. The 18650 cells are considered depleted at 3 V. This necessitated a BMS that was capable of charging at a minimum 116 W and at least 25 A. To make sure the BMS was not pushed to the extreme limits of its capabilities, a BMS capable of 120 W and 30 A was selected.

The controller needed a much smaller BMS - requiring only 7.7 A charge capability. As a result, a 7.4 V, 8 A BMS was selected.

3.3 Motor Driver Circuit

In order to properly control all three motor types in this build, it was important to come up with a method of control that would use the same IC multiple times and be controllable from the STM32-L4R5ZI. This is to reduce the complexity of the circuit overall. The individual modules controlled by the motor driver circuit are detailed in the following subsections.

3.3.1 Servo Camera Motors

The servo motor camera gimbal are comprised of two servo motors. The first rotates its shaft around pitch axis, allowing for looking up and down. The second is attached to the first and rotates its shaft

along the yaw axis so that the camera can pan left and right. The camera is attached to the shaft of this camera. Controlling servo motors is simple when compared to DC and stepper motors. The SMRAZA S51 servo motor was chosen for this application because it is fairly small and consumes low power. These motors have three leads each as shown below in Table [3].

Table 3: SMRAZA S51 Servo Motor Pins

Lead	Color	Pin
1	Red	5 V
2	Orange	Signal
3	Brown	GND

The two motors shared 5 V and GND. The signal pins were controlled via PWM pins on the STM32-L4R5ZI with a frequency of 50 Hz. Varying the Duty Cycle (DC) from 2.5% to 12.5% allowed for 180 degree travel around the shaft. Pin 2 of each servo motor was then attached to its respective PWM pin on the microcontroller. This is shown below in Figure [3].

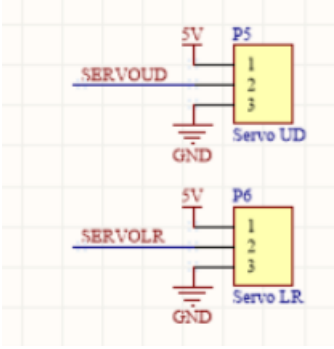


Figure 3: Servo Motor Connections.

3.3.2 DC Propeller Motors

To control the Propellers on the submarine, MSSN-1885-R/51 12 V DC Motors were chosen. The reason for this was due to the fact that the motors had excellent documentation, and had a torque of 53.5 g-cm which is more than enough for slow and steady motion needed for underwater image processing, and only consumed just under 9 W. In order to control these motors, the DRV8841PWP Dual H-Bridge IC was chosen. This IC was chosen because it could comfortably control the motors at 12 V, in addition to having safety systems that self-guard against over-current, overheating, etc. In addition, one of these ICs can control two DC motors, or a single four-lead stepper motor, which was important for the next section [3.3.3]. The circuit for controlling the propeller motors was constructed as shown in Figure [4].

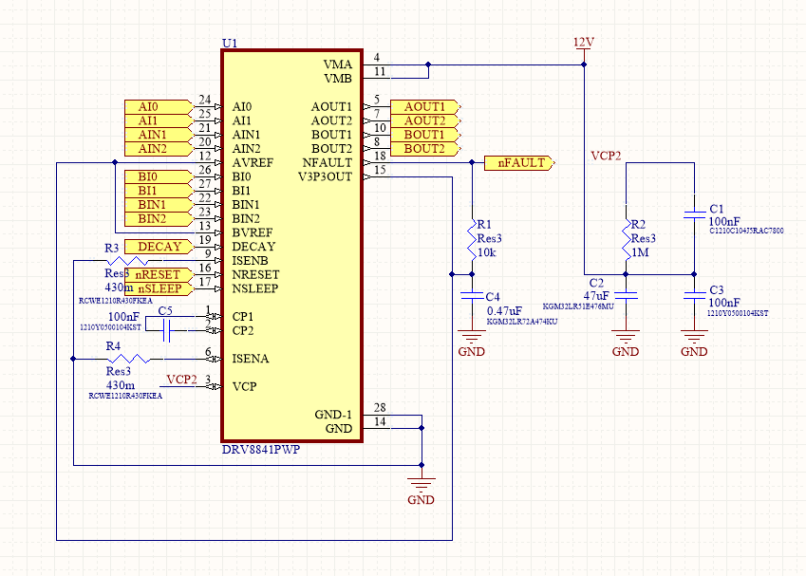


Figure 4: DC Propeller Motor Driver.

As seen in the figure above, the circuit makes use of five capacitors and four resistors as well as the DRV8841PWP IC. The component descriptions are given in Table [4] below.

Table 4: DRV8841PWP Dual H-Bridge Component List

Component	Value	Purpose
C1	100 nF	Bypass
C2	47 μF	Bulk
C3	100 nF	Bypass
C4	470 nF	Bypass
C5	100 nF	Charge Pump
R1	10 kΩ	Pullup
R2	1 MΩ	Filtering
R3	430 mΩ	Current Limiting
R4	430 mΩ	Current Limiting

All Bypass capacitors clean the DC signal at a particular node by stripping out the AC component of the signal. The bulk capacitor, C2, slows the rate that generated power from motor torque affects the 12 V supply voltage. The charge pump capacitor, C5, is an external capacitor to the ICs charge pump circuit. This circuit is used to generate a control signal that, when applied to the gate of the internal switch mode transistors, dictates whether or not current may flow between the drain and source. This control voltage must be greater than the sum of the supply voltage and the threshold voltage of the transistor.

The resistor R1 is a external pullup resistor for the fault sensor in the IC. Next, R1 functions in collaboration with R2 to create a low-pass filter to strip the charge pump voltage of any high frequency components. Lastly, R3 and R4 are current limiting resistors which were decided upon using

Equation [3] which was given by the datasheet for the DRV8841PWP IC.

$$R_{Sense} = \frac{V_{REF}}{5 * I_{MAX}} \tag{3}$$

Plugging in the a value of 3.3 V for the reference voltage and 1.5 A for the maximum motor current, a sensing resistance of 440 mΩ was calculated and a 430 mΩ resistor was used.

The four current setting pins, AI0, AI1, BI0, and BI1, are used to set the current limits in software for both H-bridges. Table [5] shows how to set up the motor driver for a specific current limit.

Table 5: DRV8841PWP current limiting pins

xI1	xI0	Current Limit (%)
1	1	0%
1	0	38%
0	1	71%
0	0	100%

Since the current had already been limited with the the sensing resistors, all four pins were set low for 100% current.

The decay pin sets the dynamic mode at which the motors will be allowed to move. This pin was set low to allow active braking when the motion signal is halted. Setting the reset pin low resets all internal logic, meaning this pin was set high during operation. Setting the sleep pin low means that the motors will go into a low power sleep state. Because of this, the sleep pin was set high during operation as well. The final discrete pin is the fault sensor for the IC. If the pin is low, this means that there is an electrical error in the IC, such as overheating or overcurrent.

The final four control pins are AIN1, AIN2, BIN1, and BIN2. These four pins are used by the internal gate driver in the IC to set the corresponding output pins, AOUT1, AOUT2, BOUT1, and BOUT2 to be tied to the supply voltage or ground. When the input is high, the voltage will be set to 12 V, and when the voltage is low, it is shorted to ground. Using a PWM input signal allows the system to set the average voltage at any lead of the two motors. Using this behavior, the average voltage can be seen to be the average differential voltage across the two output pins on either side of the motor. Controlling the DC motors is discussed in Section [3.7.3].

3.3.3 Stepper Ballast Tank Motors

The stepper motor circuit uses the same circuit as the DC motors. However, even though the DRV8841PWP can control two DC motors, it can only control a single stepper motor. Because of this, the circuit in Figure [12] is replicated for each motor.

The only component difference is the sensing resistors. Using Equation [3], 3.3V was used for the reference voltage and 1 A was used for the maximum motor coil current. A sensing resistance of

660 mΩ was calculated and a 680 mΩ resistor was used. The final difference between this circuit and that of the DC motor, is that the four output pins are tied to the the four motor leads and the input pins are controlled with GPIO pins to set the step value, rather than using a PWM signal to set the motor speed. The software for controlling this system is discussed in Section [3.7.4].

3.4 PCB Design

The goal of making a PCB for a circuit is to give it better mechanical strength and electrical connections. This was done for the circuit in the Controller and the Submarine for a total of two PCBs. In both cases, the PCBs interface the STM32 microcontrollers with their respective circuit's I/O. Both PCBs and all prototypes were designed on Altium Designer 15.

3.4.1 Controller PCB

The PCB shown in Figure [16] was built to cleanly assemble all the controller components used to take operator input and robustly connect to the STM board. Those components consisted of buttons, two axis joysticks, and a button. For feedback outside of the controller screen, a red light emitting diode (LED) was used.

To ensure that the LED was protected from too much current, a current limiting resistor was calculated and used. Since this project was already quite expensive, some components such as the switch and the LED were salvaged from other devices. As such, the exact specifications were unknown. Given its physical composition, it was assumed that it was a standard 5 mm red LED with a maximum current of 20 mA and a typical forward voltage of 1.8 V. Using the following equation, a 100 Ω resistor was calculated.

$$R = \frac{V_s - V_F}{i} \tag{4}$$

To ensure that the LED would be well within its operating parameters, a 1 kΩ resistor was used instead of a 100 Ω resistor.

Additionally, the switch needed a pull-up resistor to ensure that the GPIO pin it was connected to would not float. To calculate, the following equation was used.

$$R = \frac{V_{cc}}{i_R} \tag{5}$$

An input voltage of 3.3 V was used, as well as a 3.3 mA current. This resulted in a 1 kΩ resistor.

3.4.2 Submarine PCB

The Submarine PCB, shown below in Fig [17], is more complex, connecting all parts of the system together in one comprehensive circuit. First, the power supply delivers both 12 V and 5 V to the PCB, which is then distributed to power STM32-L4R5ZI, the RPi, headlights, linear variable resistors, and all motor drivers. The IMU is the sole component to be powered not by the PCB, but by the 3V3 pin off of the RPi. In addition, many prototypes made it apparent that having the ability to swap STM32 microcontroller was important. That led to the addition of female Dupont headers placed on the STM Hat of the PCB. Lastly, all of the components are connected to the PCB through headers that can be connected and disconnected easily, while maintaining mechanical and electrical strength. All components were surface mount and were attached using solder paste and an oven. After that, the headers were attached using a soldering iron. This PCB is home to three DRV8841PWP ICs and their respective external components, in order to drive both stepper motors in addition to the two DC propeller motors. All STM32 connections are routed to the proper pin locations for I/O.

3.5 Controller and Submarine Initialization Sequence

The following subsections will detail the initialization sequence of the controller and the submarine.

3.5.1 Controller Initialization Sequence

As discussed in Section [2.2.1], there are multiple custom service files which set the environment up and allow the system to automatically run the GUI and command publishing programs. Just after boot, the controller runs a service file which launches a shell script that will perpetually attempt to ping the submarine RPi every five seconds until the communication is successful. Once the communication is established, the shell file exits, another service file is called, and it sets the necessary environment variables to allow ROS2 to access the connection between both RPis. This service spawns a final shell script which launches the ROS2 nodes and displays the GUI and publishes commands over the ROS2 network down to the submarine.

The same process is true for the submarine initialization. The only difference is the final shell script that is spawned launches two nodes. One begins receiving data and sending commands to the attached STM board, and the other takes sensor readouts and publishes the data over the ROS2 network to the controller.

3.6 ROS2 and Serial Network

The RPis are responsible for establishing communication between the circuits of the controller and submarine. Each RPi communicates to it's respective STM32 microcontroller via serial over USB, while communication between the two RPis is accomplished through the use of ROS2. This is a Python/C++ library that allows for TCP and/or UDP communication between clients and servers. This Library is common to the point that it is expected in robotics applications due to its extensive

support for all types of robotics and visualization of systems. In addition, it allows for many programs to run simultaneously while managing a communication network of data structures. Since this project is being used in future robotics research, the use of this library was essential. This section will describe the purpose behind each ROS2 package and how it works.

3.6.1 ROS2 Network

There are multiple versions in the ROS2 ecosystem. The chosen version was ROS2 Foxy for a couple of reasons. Firstly, it has been released since Ubuntu 20.04 released. This has allowed the open source foundation to release bug fixes and for the community to make a massive assortment of useful packages. Moving up to a newer version could pose hazardous, as there is a risk of changes between ROS2 versions that break previously functional packages. Additionally, there is the existence of ROS1. Although it is still supported, a large issue is that it only supports TCP connections. Being forced to have every connection be TCP would put a strain on the system and increase latency as the GUI program would have to ensure it received every single camera frame.

3.6.2 Controller Publisher

This node had two responsibilities. The first is to serially interface with the attached STM board. As discussed in Section [2.2.3], the publisher queries data at a rate of 20 Hz. In order to deduce the rate, a set of empirical tests were run. Those tests were subjective to the operators - too slow a response time, the ROV becomes sluggish and difficult to operate. Querying data too quickly simply wastes the RPi's resources, as it must spend more time querying data from the STM board than doing other processing. There are five different datapoints that needed to be gathered. The ascend and descend commands, screenshot, movement, and camera movement. Once a frame of data has been received, it placed the data in a message structure and then published it using ROS2 TCP protocol to the submarine.

Finally, the imperfections of the 2-Axis toggles needed to be dealt with. If the full resolution of the STM board's ADCs were utilized, the submarine would appear to jitter, as the values would fluctuate slightly. To account for this, a "dead zone" was established in the software. The toggles had a center value of 128, so a range of 100 to 150 was used. This ensured that the submarine operated much more smoothly.

3.6.3 Controller GUI

The controller GUI node has three UDP connections that it makes. The first connects to the submarine-side camera. The second connection receives information from the submarine [3.6.4].The third connects to the controller publishing node.

In order to draw the GUI over the images from the submarine, an image augmentation library had to be selected. To satisfy this need, OpenCV was chosen as it has been written in highly optimized C and C++. Not only are these faster languages than high level languages like Python, but it also allows for multi-core processing. In order to display the images, Tkinter was chosen because it is

native to Python, and like OpenCV, much less computationally intensive than alternatives. To allow for easy measurement of the frames per second (FPS), a pin on the RPi is toggled on once the image is obtained, and off once the final image is displayed. A custom built module is called to draw the heading, FPS, ballast tank level, run time, battery level, and any error messages from the submarine.

In order to get the data drawn on the received image, the submarine connection simply collects all data being published from the submarine and stores it locally for the image routine to interpret and draw. The controller publishing connection is solely responsible for ascertaining if a screenshot is desired. If it is, the image routine will save the image after the image augmentations. The controller GUI is displayed in Figure [5].

Figure 5: Controller GUI under Operating Conditions

As shown in the above figure, the module draws the pieces of the GUI by using OpenCV's built in rectangle, polyline, and text placement functions. In order to draw the ballast tank icon, the boxes are custom defined, and a rectangle fills from top to bottom of the ballast tanks to depict the fill level of each individual tank. Similarly, the battery bar is drawn depleting from right to left, and as the battery discharges, the bar appears increasingly red. In order to draw the cardinal headings, the headings are stored in a list and iterated through. The current heading is taken, and the cardinal position multiplied by 45 is added to it. This allows the calculation of the actual degree in a clockwise manner. Additionally, only the bottom 120° are shown, so instead of counting from 0 to 360, the measurement is done from -180 to 180. Finally, if the degree is between -60 and 60, the headings are displayed. The image is then passed back to the main loop and the error messages, uptime counter, and any error messages are drawn over the image. The OpenCV image is then converted into a Pillow image format and displayed using Tkinter.

3.6.4 Submarine Communication

The submarine communication node functions to take the input commands from the controller publisher [3.6.2], deliver the information to the STM32-L4R5ZI, receive information back, and send relevant information back to the controller. This node is written in python with an object oriented style. The program consists of three objects: IMU, StmSend, and SubData. The IMU node initializes by averaging fifty measurements of the linear acceleration to measure the steady state bias and calibrate the system. Then, each time that information comes in from the controller, the IMU is measured to obtain the roll, forward acceleration, upward acceleration, and cardinal directions. This is done by fusing all degrees of freedom together using the MadgwickAHRS algorithm [REFER TO THAT HERE], after receiving the base movement information from the supplied python library. The roll is then stored into the StmSend node to be delivered to the STM32-L4R5ZI, while the remaining information is stored in the SubData node for later. Next, the StmSend node performs a handshake with the STM32-L4R5ZI by sending the roll from the IMU along with the controller inputs over serial, and then receiving sensor data over the same serial connection. Finally, this information is stored in the SubData node. The SubData node then packages the information into a ROS2 message which is then sent to the controller GUI node.

3.6.5 Serial Communication

To send information back and forth between the STM32 microcontrollers and the RPis, the serial data transmission protocol was used. For sending and receiving information on the RPi, the python3 library "pySerial" was used. This allows for bidirectional communication with only a few function calls. On the STM32 side, the information is sent using the HAL library after setting up the transmission pins in the development environment. For receiving data, and global interrupt was set up that triggers the main loop after storing the data in a string. The Controller STM32 microcontroller is set up to only send controller inputs, while the submarine is set up to receive information, and the send back sensor data.

3.7 Motor Control

All of the computation that drives the six motors on the submarine is done on the STM32-L4R5ZI microcontroller. Each of the three different types of motors required a different control signal. This section will discuss the process of actuating all six motors in regards to the input data from the controller.

3.7.1 Parsing Inputs

The first step in controlling the motors on the submarine was to interpret the commands coming from the controller. On its way from the analog input devices all the way to the submarine, the command data transforms to the format that makes the most sense for the communication protocol. This data is sent over serial to the controller RPi and then sent to the submarine RPi using ROS2. From there, the data is assembled into a 42 byte binary string. This includes one byte for commanding ascent

and descent for each, in addition to eight bytes each for the camera's vertical and horizontal position, the propellers forward and rotational speed, and the current roll. Once 42 bytes were received over serial, an interrupt is triggered which stored the information and set a flag that lets the main loop know that new data is ready to be parsed.

Once the main loop has noticed the new data, the first step is to read the six sensors on the submarine that are measured by the STM32-L4R5ZI. These are the three fault sensors for the propellers and ballast tank motors, the two linear variable resistor positions, and the battery voltage. The three fault pins are active high discrete input pins that signal an electrical issue in the motor drivers such as a short circuit or overheating. To measure these sensors, an input pin is read and the opposite of the reading is stored. To read the ballast tank wall positions, the linear variable resistors are tied to the 3.3 V regulator from the STM32 microcontroller, as well as ground. Then, the wyper pin is connected to an 8-bit ADC pin. This is replicated for both ballast tank motors. To measure the position from this sensor, the ADC pin will translate the analog signal to a digital value between 0 and 255. The digital value can be calculated using Equation [6] below.

$$DigitalValue = \frac{255 * WyperVoltage}{3.3V} \tag{6}$$

This value can then be used as a position. A value of 128 means that the wall is centered, while a value less than that means it has pulled in water and is decreasing buoyancy. The opposite can be said for the other direction. The final sensor is the battery voltage detector. To bring the voltage down linearly to a readable voltage for an ADC pin on the STM32-L4R5ZI, a buck converter was used to reduce the unregulated battery voltage from a maximum of 14.8 V to just 3.3 V. This is a reduction of 4.5 $\frac{V}{V}$. This analog value is plugged in to the Digital Value of Equation [6] to calculate the wyper voltage. When this is multiplied by 4.5 $\frac{V}{V}$, The actual voltage can be determined. Once all six of these sensors are measured, the data is organized into a structure to be used later. Second, the different fields of the input data are converted from binary to decimal to determine the control variables that will govern the next state of movement for the submarine motors. With all of this done, the motor control can begin.

3.7.2 Servo Control

This first type of motor control is for the two servo motors. The hardware configuration for this motor driver was discussed in Section [3.3.1]. The SMRAZA S51 Servo motors require a PWM control signal with a frequency of 50 Hz. Controlling a PWM signal on an STM32 microcontroller requires first calculating the Auto-Reload Register Value (ARR). This is how many clock cycles will go by for every period in the PWM signal. This value can be calculated using Equation [7].

$$ARR = \frac{ClockFrequency}{PWMFrequency} - 1 \tag{7}$$

Plugging in 120 MHz for the clock frequency and 50 Hz for the PWM frequency, the ARR was calculated to be 2,399,999. Controlling the 180 degree position requires a DC between 2.5% and 12.5%. The command given from the remote control is an 8-bit value where 128 is centered, 0 is

the furthest to the left, and 255 is furthest to the right. Mapping this command to a DC value is done using Equation [8].

$$DC = \frac{DC_{MAX} - DC_{MIN}}{2^8 - 1} * (8bitCommand) + DC_{MIN} \tag{8}$$

Once the DC has been calculated for its respective command, the next value to set is the Capture/Compare Register (CCR). When the PWM signal counter is counting from 0 to the ARR for every period, the PWM signal will be high when the counter is below the CCR, and low when it is higher. This is the value that is given to the PWM object to directly control the DC, thus controlling the motor speed. This value is calculated using Equation 9.

$$CCR = ARR * DC \tag{9}$$

The DC and CCR are calculated for each command, while the ARR is only calculated once at the beginning. Since there are two servo motors, this process is done twice each time the controller inputs are updated.

3.7.3 DC Control

Controlling the DC motors for the propellers requires controlling the average voltage across the two motor leads. This is done by controlling the average voltage of both leads. To set the voltage across the motor, one side was always set to be 0 V, while the other side was set to be a PWM signal that could set the average voltage using Equation [10].

$$V_{avg} = V_{supply} * DC \tag{10}$$

Changing the lead that is the PWM signal at any point allows the motor to change direction since the voltage across the motor is being multiplied by -1. Using the Jameco 232022 12 V DC motor, the maximum DC was experimentally determined to be only 33% since it was the maximum desired propeller angular velocity. Using Equation [7], and plugging in 120 MHz for the clock frequency and the required 10 kHz for the PWM frequency, the ARR was calculated to be 19,999.

The command coming from the controller is a set of two 8-bit values. The first is the turning thrust, where 128 corresponds to 0 $\frac{radians}{second}$, 0 corresponds to maximum counter-clockwise thrust, and 255 corresponds to maximum clockwise thrust. The second command is forward thrust, where 128 corresponds to 0 $\frac{meters}{second}$, 0 corresponds to maximum reverse thrust, and 255 corresponds to maximum forward thrust. Mixing these two commands together into a thrust value for each motor requires first mapping the turn thrust into its corresponding left and right thrust values, and the mapping the forward thrust into its corresponding forward and backward thrust values. All four of these values vary from 0% to 50%. For example, if the turn command is 64, the left thrust value is 25% and the right thrust is set to 0. When the forward command is 192, the forward thrust is set to 25% and the backward thrust is set to 0%. The final step is to combine these four values into the left motor thrust using Equation [11] and the right motor thrust using Equation [12].

$$DC_{LeftMotor} = Thrust_{right} - Thrust_{left} + Thrust_{forward} - Thrust_{backward} \tag{11}$$

$$DC_{RightMotor} = Thrust_{left} - Thrust_{right} + Thrust_{forward} - Thrust_{backward} \tag{12}$$

This process converts the two controller input commands from a single joystick into the individual thrust values for each motor which directly relate to the DC. Multiplying both of these values by the maximum DC of 33.3% will give the DC of the two motors. The sign of the DC corresponds to the lead of the motor that should be connected to the PWM signal, and which one should be connected to ground, while the magnitude of the DC is used in Equation [9] to find the CCR. Connecting a motor lead to ground is done by setting the leads PWM CCR to 0.

3.7.4 Stepper Control

Controlling Stepper motors does not require a PWM signal. Instead, the four-lead PM25L-075-035 stepper motor is controlled using four GPIO pins. This motor is controlled by setting the pin voltages in a particular order to step forward or backward. The advantage of a stepper motor is its unique precision control. DC motors control speed, while stepper motors are more related to controlling their position. In order to step forward and backward, the pins must be set in the order shown in Table [6].

Table 6: Stepper Control Order

Pin	Color	Step 1	Step 2	Step 3	Step 4
1	BLACK	1	0	0	1
2	BROWN	0	1	1	0
3	ORANGE	1	1	0	0
4	YELLOW	0	0	1	1

Starting from step 1, moving toward step 4 will move clockwise, while the opposite direction moves counter-clockwise. The step value simply wraps to the other end of the table once it reaches past the final value, meaning step 5 corresponds to step 1, and step 0 corresponds to step 4. Since the motor was turning a lead screw with a threading of 2 $\frac{mm}{revolution}$ and the motor resolution is 24 $\frac{steps}{revolution}$ the ballast tank wall position will move 83 μm per step.

The stepper control system is a proportional control feedback system that uses a linear variable resistor as a position sensor. This system is replicated for both sides of the submarine, which allows for control of the depth and roll. Getting the position from the linear variable resistors is described in Section [3.7.1]. The roll is controlled passively with the proportional control system while the depth is controlled by adding a linear offset to the two stepper motors to increase or decrease buoyancy. The feedback system for controlling the roll is shown in Figure [6].

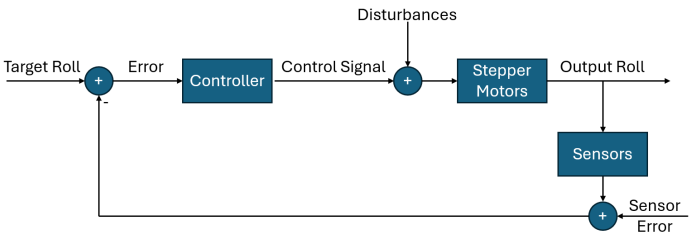


Figure 6: Roll proportional control.

The first step of this system is calculating the error by subtracting the roll, found by measuring the position using the sensor, from the target roll of 0. This value is then sent to the controller. This error value is equal to the opposite of the roll of the submarine. This error is then used to set a target position for the two ballast tank walls that is 1 step per degree of roll in the direction that applies buoyancy to the sinking side and removes buoyancy to the rising side, up to 30 steps in either direction. Next if the submarine needs to ascend, 15 steps are taken to let in air and raise the buoyancy of each side equally. The inverse is done to descend. Once this target position is calculated, the stepper motor needs to move a step toward that target. This involves setting the four GPIO pins for each motor to move in the correct direction. This process is repeated for every incoming command from the controller, which results in a stable control system.

4 Results

The Submersible Drone for Hull Inspections was successfully designed to allow a user to navigate an aquatic environment. The ROV is capable of streaming images and display at GUI at a minimum of 15 FPS. The system operates for over an hour and can recharge in less than four hours. Additionally, the controller was built on a custom PCB. All contract specifications were at a minimum met, and in some cases exceeded.

To ensure that all of the ROV systems were operating as expected, a number of testbenches and measurements were taken. Those methodologies are discussed in the subsequent sections.

4.1 Power Supply Measurement

To meet the power supply specification, the submarine needed a 12 V $\pm$ 5% voltage supply with less than 100 mV of ripple at a 1 A load current. To prove this specification, a PCB was designed to set a load current of 1 A using a resistor. The voltage could then be measured across the resistor to determine the battery voltage and current. This load is designed to be purely resistive, using a single resistor. To calculate the resistance for the resistor, the supply voltage of 12 V was divided by the target load current of 1 A. This resulted in a resistance of 12 Ω. The power dissipated by this resistor was calculated by multiplying voltage across the component, 12 V, by the load current

of 1 A. This resulted in a dissipated 12 W. To ensure the proper functioning of the resistor, a 20 W resistor was chosen. The measured power supply voltage is shown in Figure 20 and the measured load current is shown in Figure 22. These results show an average voltage of 11.7 V, which is well within specification. In addition, the average load current was measured to be 960 mA, which is only 40 mA away from the planned load current. The offset is due to an actual measured resistance value of 12.15 Ω rather than the ordered 12 Ω resistance. The final measurement was for the voltage ripple, which is shown in Figure 21. The measured value was a ripple of 97 mV peak to peak, which is well within specification.

4.2 Recharge and Deploy Time

The recharge and deploy time of both the submarine and controller were empirically tested to ensure that both components met the requirement of hour long deployment and sub four hour recharge time. In order to test the deployment requirement, both battery packs were completely charged. Both systems were then powered on, and a timer set while both systems ran. Both systems were verified to operate for over an hour. To verify the recharge requirement, both systems began from the completely depleted state and recharged. The submarine was verified to recharge in two hours and 45 minutes. The controller was verified to recharge in approximately one hour and 45 minutes. Both systems far exceeded the deploy time and recharge time requirements.

4.3 Framerate Measurement

In order to ensure compliance with the framerate requirement, a variety of tests were conducted. To conduct the first, the time was marked when the frame was received, and then the final time was marked after all of the augmentations had been completed. The difference was then drawn on to the image just before being displayed. To ensure that the measurement was accurate, a pin was toggled on when a frame was received, and off when a frame was displayed. The frequency was measured on an oscilloscope and was found to vary between 15 and 20 FPS. The corresponding oscilloscope output is available in Appendix D Figure [18].

4.4 Submarine Test bench

In order to test the different hardware and software components of the submarine, several test-benches were set up. To test the software, several test files were programmed in python and C to test the ROS network, the serial communication between the RPi and the STM32-L4R5ZI, and the IMU. Knowing the inputs and outputs of each component, fake yet reasonable inputs were provided to the different programs. The individual components were only committed to the main project code once the outputs were correct for the given inputs.

To test the hardware, the first step was to test the three types of motors supply their respective motor drivers with the required input signals using a wave generator that could be easily set to supply the correct signal. Once it was determined that the motors were spinning correctly with changes in their input signals, the next step was to make sure that the STM32 code was producing the

correct signals that could be delivered to the motor drivers. An oscilloscope was used to measure the control signals and verify that they were working correctly. The last step was to connect all of the hardware together and measure the voltage across the motors as they changed angular velocity. Figure [19] shows the differential voltage across one of the DC motors found by connecting the high sides of two oscilloscope channels to either lead of the motor, and connecting both low sides to AC ground. Running the oscilloscope in differential mode allows for a custom channel that is the result of subtracting the voltage at channel one by the voltage at channel two.

This shows that the voltage across the motor is following the PWM control signal from the STM32-L4R5ZI and proves that the hardware and software systems are working as intended.

4.5 Controller Test bench

Before running the GUI on the ROS2 network between both RPis, it was necessary to test the GUI capabilities. This entailed making individual files in which each component was built. The first test file centered around constructing the battery icon and placing it in the upper right hand corner of the GUI. The battery needed to properly display the battery level in percent, as well as a gradient bar that would change from green to red as the battery depleted. To test this, battery voltages were iterated through, and response from the icon was observed to ensure that the battery functioned as intended. The next test involved placement and response of the compass. To test functionality degrees varying from 0 to 360 were passed to the function, and each possibility was passed through to ensure that the compass operated as expected in each possible scenario. The console was tested next, ensuring that the longest sent message from the submarine would fit within the confines of the console boundaries. The program timer and frame counter were tested in a similar fashion. Finally, the ballast tank icon was tested to ensure that the ballasts properly display fill level. This test was run in a similar fashion - values that would be sent from the submarine were passed to the function. As shown in in Figure [5], all aspects of the GUI functioned as needed.

5 Conclusion

This document described the process of designing and testing the underwater ROV for hull inspections. The underwater ROV was successfully designed to operate for one hour, recharge in under four hours, a minimum framerate of 15 FPS, and a custom controller PCB. Through the custom controller interface, an operator is able to get active updates on the entire system's status as well as save any desired images to a local disk. Section [2] gave an overview of the project and gave a high-level view of each module. Section [3] discussed each module in greater detail and justified each decision made. Section [4] discussed the results of the underwater ROV project and the measurements taken to ensure that it met or exceeded each designated requirement.

6 References

[1] DIY Perks, “Building a DIY submarine,” www.youtube.com, Dec. 03, 2021. https://www.youtube.com/watch?v=pUba126uzvU&t=449s (accessed Feb. 25, 2024).

[2] Drix’ Works, “DIY: How to make 4s2p 12Ah Battery Pack | 32650 LiFePo4 Battery | 100A BMS | DIY,” www.youtube.com, Aug. 24, 2021. https://www.youtube.com/watch?v=ZrEvtiP-XM8 (accessed Feb. 25, 2024).

[3] Arduino, “Arduino and Stepper Motor Configurations | Arduino Documentation,” docs.arduino.cc. https://docs.arduino.cc/learn/electronics/stepper-motors

[4] Arduino, “Servo Motor Basics with Arduino | Arduino Documentation,” docs.arduino.cc, Feb. 20, 2023. https://docs.arduino.cc/learn/electronics/servo-motors

[5] K. Hunter, “H-Bridge Circuit Design | MicroType Engineering,” MicroType Engineering, 2019. https://www.microtype.io/h-bridge-circuit-design/

[6] B. Siepert, “Adafruit TDK InvenSense ICM-20948 9-DoF IMU,” Adafruit Learning System, Aug. 05, 2020. https://learn.adafruit.com/adafruit-tdk-invensense-icm-20948-9-dof-imu (accessed Feb. 25, 2024).

[7] Great Power, “Product Specification Model: 18650,” Jul. 2012.

[8] Texas Instruments, “DRV8841 Dual H-Bridge Motor Driver IC,” www.ti.com, May 2010. https://www.ti.com/lit/ds/symlink/drv8841.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1708841924169

[9] Parallax, “2-Axis Joystick,” Parallax, Aug. 13, 2020. https://www.parallax.com/package/2-axis-joystick-product-documentation/

[10] Shipnerd. "Underwater cleaning of ship hell - Explained, Nov. 4, 2022 https://shipnerdnews.com/underwater-cleaning-of-ship-hull-explained/

[11] Song, C. Cui, "Review of Underwater Ship Hell Cleaning Technologies", J. Marine Sci. Appli., 2020. https://doi.org/10.1007/s11804-020-00157-z

Appendix A Project Contract

Submersible Drone for Hull Inspections

Dyllon Dunton (CE)

Jacob Wildes (CE)

April 12, 2023

Description

The control, power, and propulsion systems for a submersible drone will be designed, built, and tested in this project. The purpose of this submersible drone is to inspect the hulls of ships. The submarine will stream a live camera feed to the controller on shore, where images and videos will also be stored to a USB drive. The submarine and its controller will be powered with a rechargeable power supply and operate for at least one hour. Finally, the controller will communicate with the submarine via Transmission Control Protocol and User Datagram Protocol (TCP/UDP) over ethernet.

Inputs

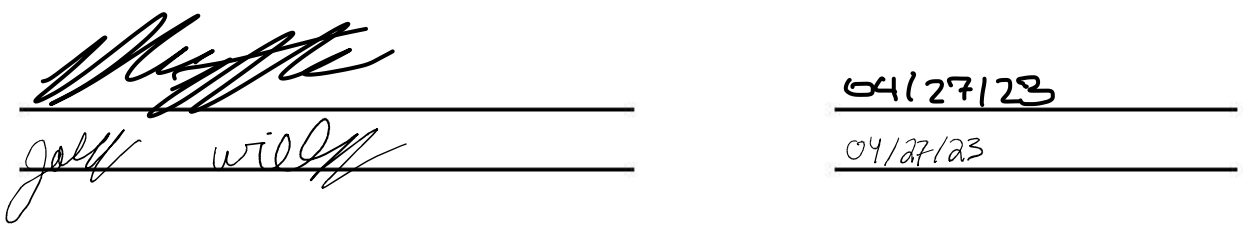
- Controller with joysticks and buttons for input
- Submarine controls built on microcontroller reading input from controller over serial

Outputs

- Controller displays battery voltage and submarine depth and camera feed
- Images and videos will be stored locally on the controller

Specifications

- Power Supply with output voltage of 12 V ±5% with 100 mV ripple at 1 A load current
- Recharge time of less than 4 hours
- Minimum of one-hour long deployment time
- Custom PCB for the controller
- Minimum frame rate of 15 FPS



Appendix B Bill of Materials

Line	Vendor	Quantity	Part Number	Description	Price	Extended
1	JLCPCB	5	-	Submarine H-Bridge Motor Driver PCB	$46.3080	$231.54
2	OSH Park	3	-	Controller PCB	$15.30	$45.90
3	OSH Park	3	8ktWRktq	Battery Bank test PCB	$1.23	$3.70
4	Sparkfun	10	PRT-138189	18650 Lithium-Ion cells	$6.95	$69.50
5	Jameco	2	232022	12 V DC Motor	$4.041	$8.08
6	Lowes	1	AC3603-BE-V1	Cat-6 Ethernet Cable	$6.98	$6.98
7	Adafruit	1	4754	9-DOF IMU	$24.95	$24.95
8	Adafruit	2	245	2-axis Joystick	$9.95	$19.90
9	Amazon	1	6453681	ELECROW Display for Raspberry Pi	$55.99	$55.99
10	Amazon	1	B09XRG6LXG	18 Gauge twisted pair cable 16.4’	$9.99	$9.99
11	Amazon	1	B075D3BMKF	4S2P Battery Management System	$8.99	$8.99
12	Amazon	1	B07KSQY67X	2S Battery Management System	$9.49	$9.49
13	Amazon	1	B07D5RRXTS	Kapton Tape	$10.99	$10.99
14	Amazon	2	B07B7G1KRX	IP57 Charger Connector	$9.99	$19.98
15	Amazon	3	B00C4QVTNU	Variable Voltage Regulator	$10.45	$31.35
16	Amazon	1	B015Y6JOUG	Jumper Wire connectors	$10.99	$10.99
17	Amazon	2	B0BFHZ9XYX	12V/24V Anti-spark power switch	$9.99	$19.98
18	Amazon	1	B09NX39NWR	10k Ohm Linear Resistors	$11.99	$11.99

19	DigiKey	2	PM25L-075-035	Stepper Motor	$21.83	$43.66
20	DigiKey	2	497-17688-ND	STM32 Nucleo-144 STM32L4R5ZI	$20.42	$40.84
21	DigiKey	6	296-47909-ND	Bi-Polar Motor Driver	$6.26	$37.56
22	DigiKey	10	399-14719-1-ND	0.1uF 50V Ceramic Capacitor	$0.439	$4.39
23	DigiKey	10	478-KGM32LR51E476MUCT-ND	47uF 25V Ceramic Capacitor	$2.153	$21.53
24	DigiKey	10	1608-1210Y050010KSTCT	1210 50V 100nF Capacitor	$0.469	$4.69
25	DigiKey	10	478-KGM32LR72A474KUCT-ND	0.47uF 100V Ceramic Capacitor	$0.341	$3.41
26	DigiKey	10	478-KGM32ECG1H103FUCT-ND	10000pF 50V Ceramic Capacitor	$2.047	$20.47
27	DigiKey	10	13-RC1210JR-0710KLCT-ND	10k Ohm Resistor	$0.065	$0.65
28	DigiKey	10	A130635CT-ND	1M Ohm Resistor	$0.146	$1.46
29	DigiKey	10	RCWE.68FCT-ND	0.68 Ohm Resistor	$0.376	$3.76
30	DigiKey	10	RCWE.43FCT-ND	0.43 Ohm Resistor	$0.461	$4.61
31	DigiKey	3	AP72512RF-ND	12 Ohm 20 W Resistor	$6.19	$18.57
32	DigiKey	2	2648-SC0195(9)-ND	Raspberry Pi 4B 8GB	$75.00	$0.00
33	DigiKey	1	497-17328-ND	Nucleo-L452RE-P for Controller	$15.96	$0.00
34	Walmart	1	960-000694	Webcam	$23.99	$23.99
TOTAL						**$829.88**

Appendix C Schematic

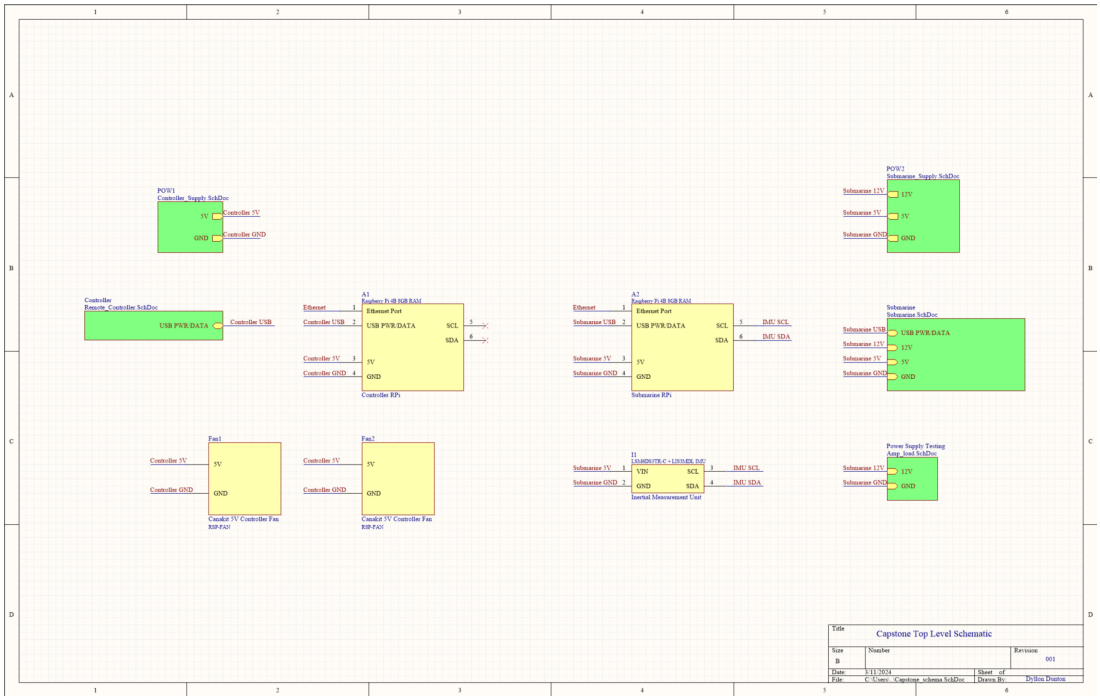


Figure 7: Top level schematic.

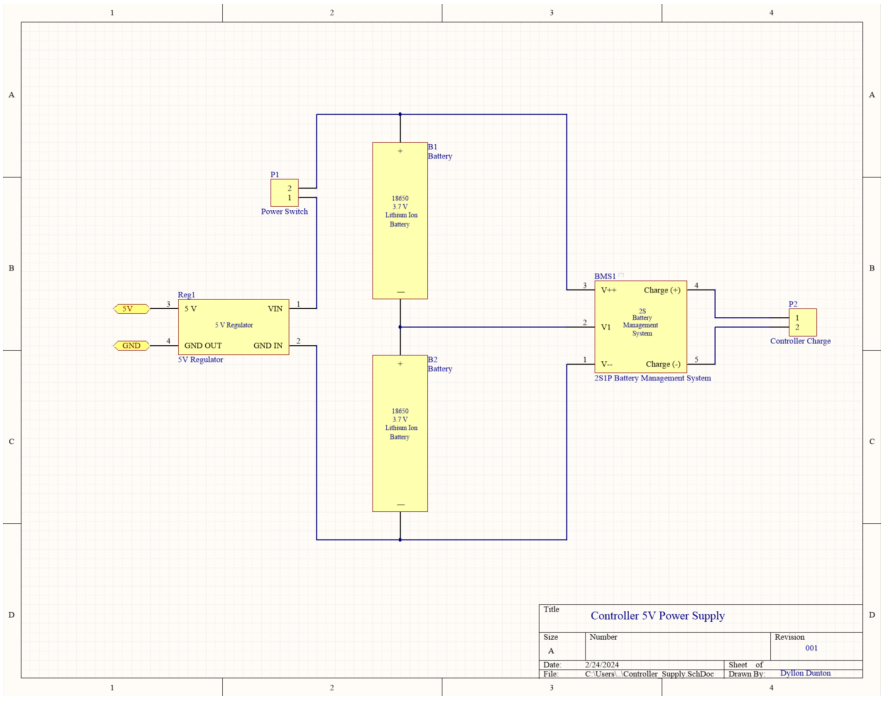


Figure 8: Controller power supply schematic.

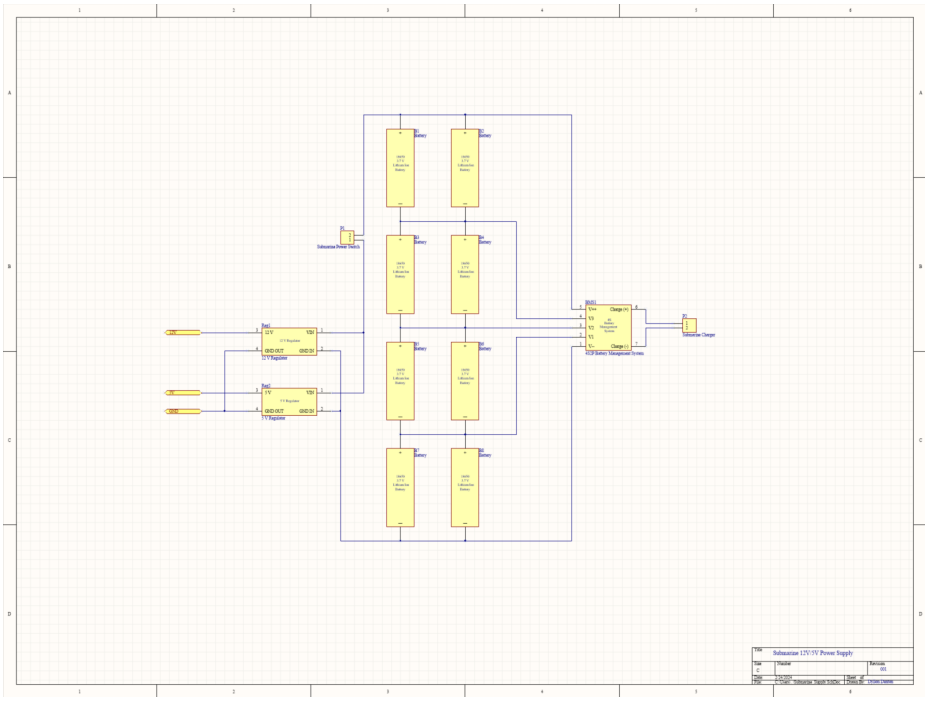


Figure 9: Submarine power supply schematic.

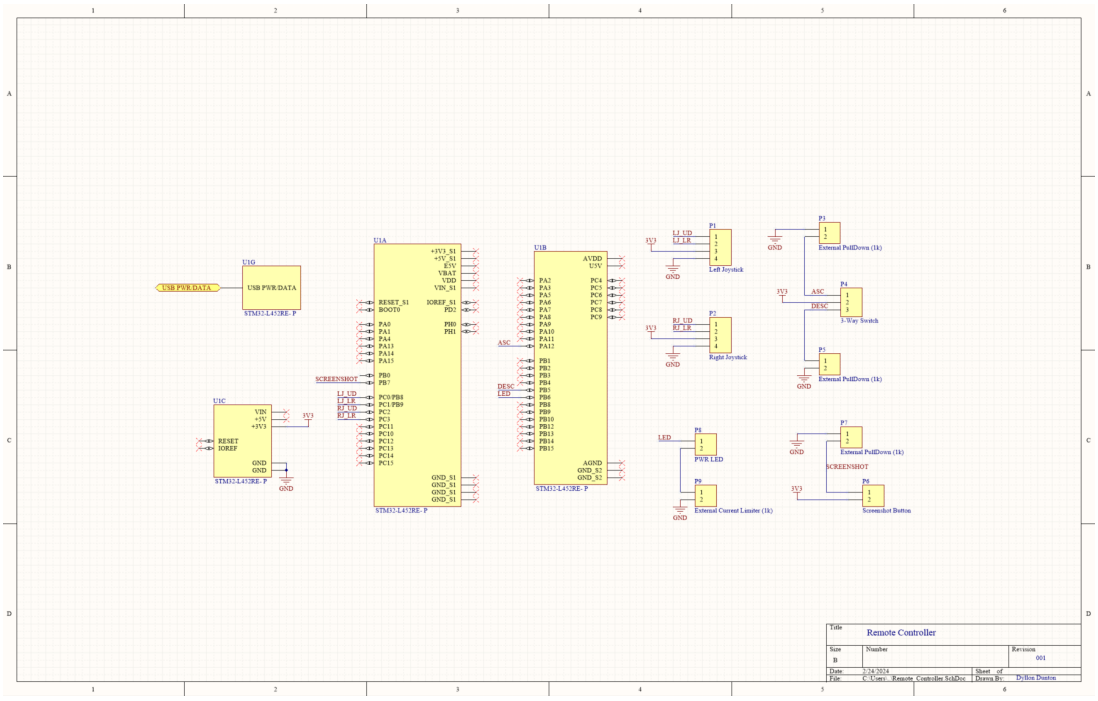


Figure 10: Remote controller schematic.

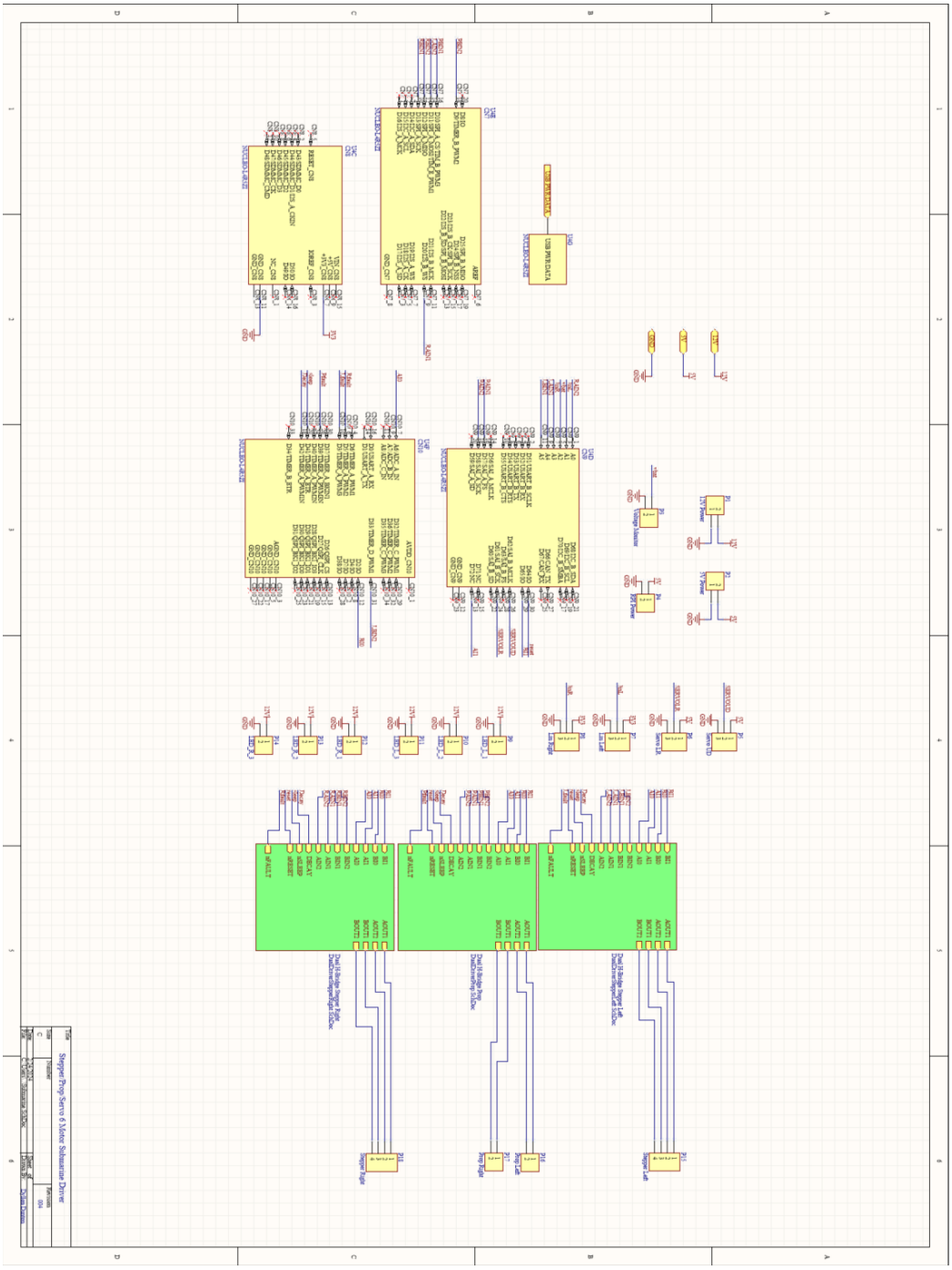


Figure 11: Submarine motor driver schematic.

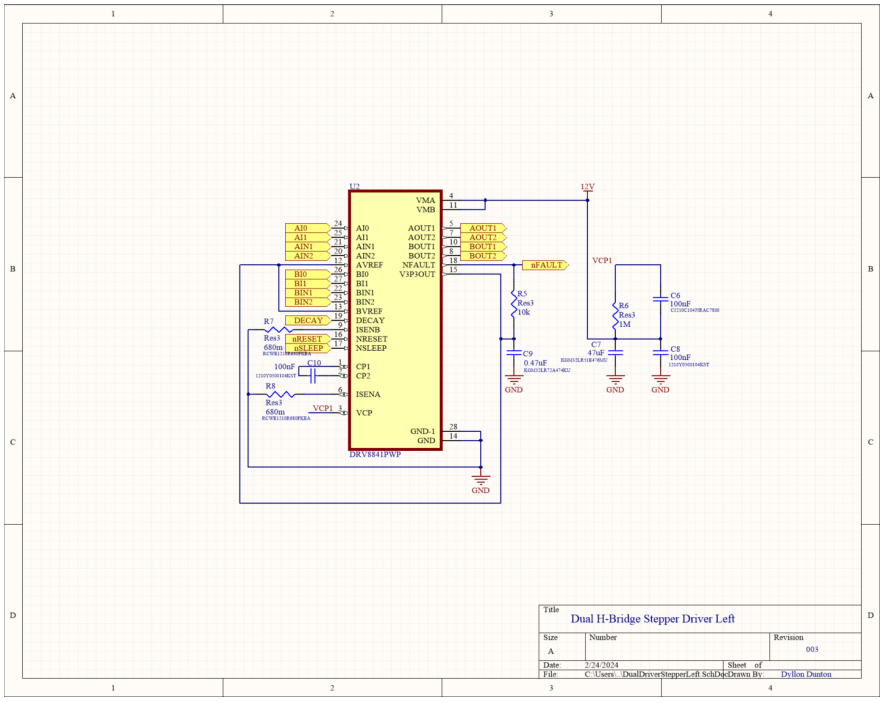


Figure 12: Left stepper motor driver schematic.

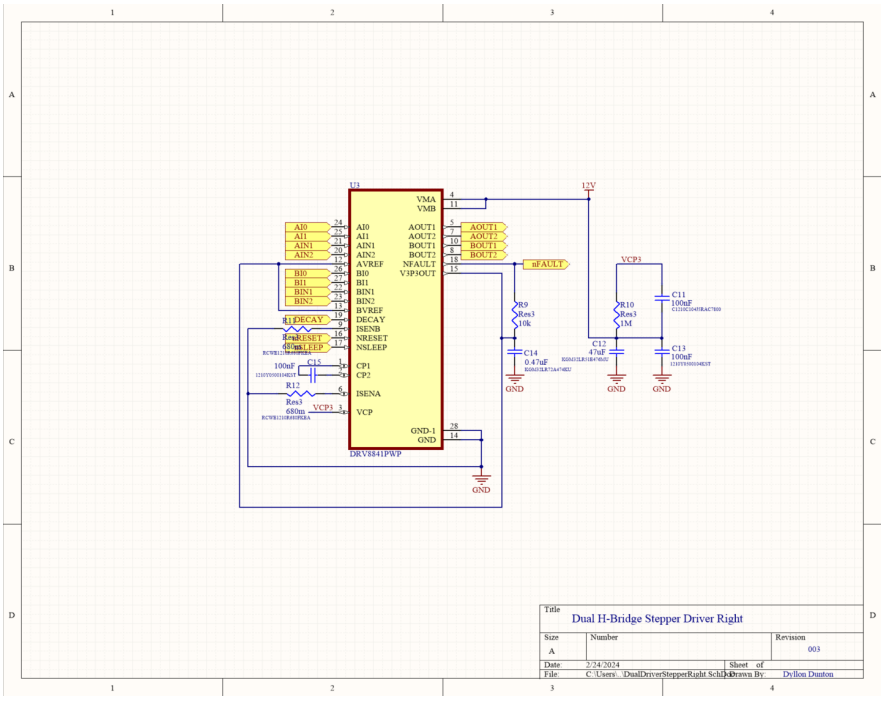


Figure 13: Right stepper motor driver schematic.

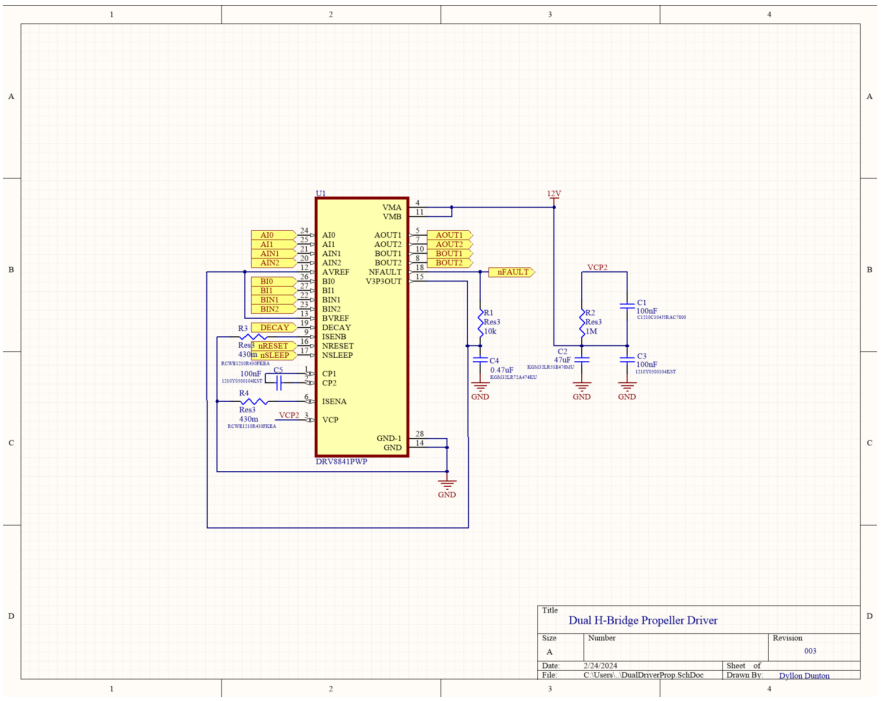


Figure 14: Propeller motor drivers schematic.

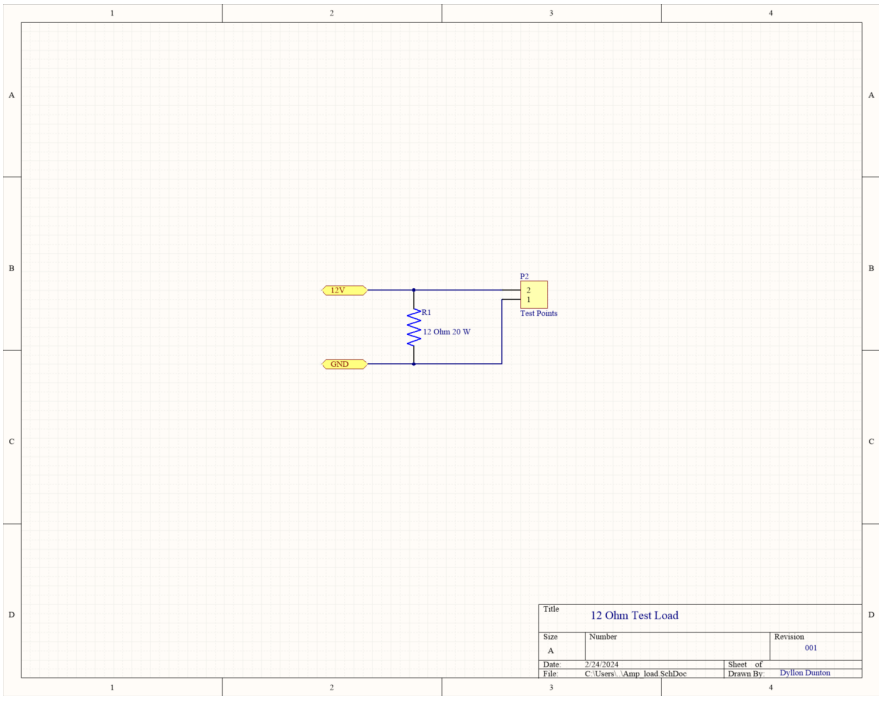


Figure 15: One amp test load schematic.

Appendix D Additional Figures

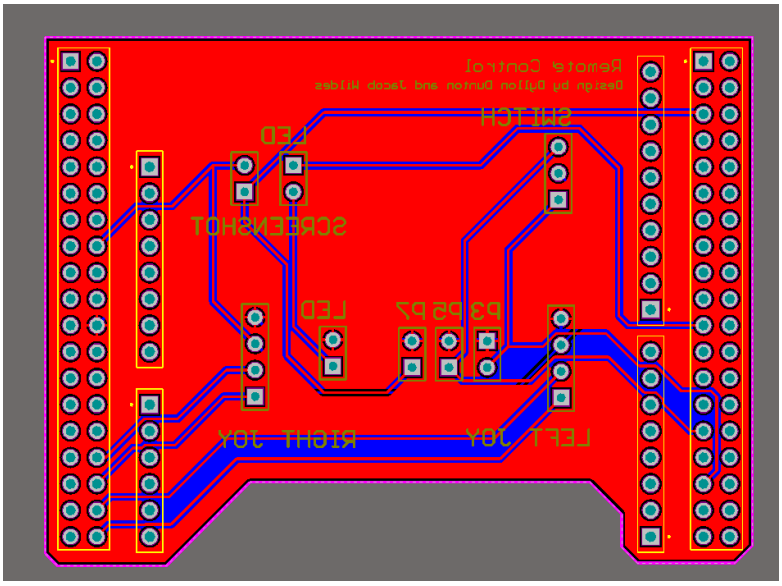


Figure 16: Controller PCB.

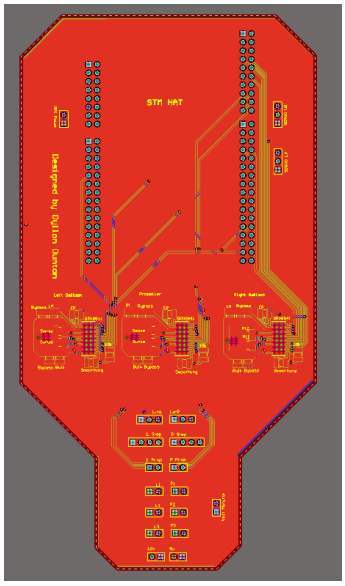


Figure 17: Submarine PCB.

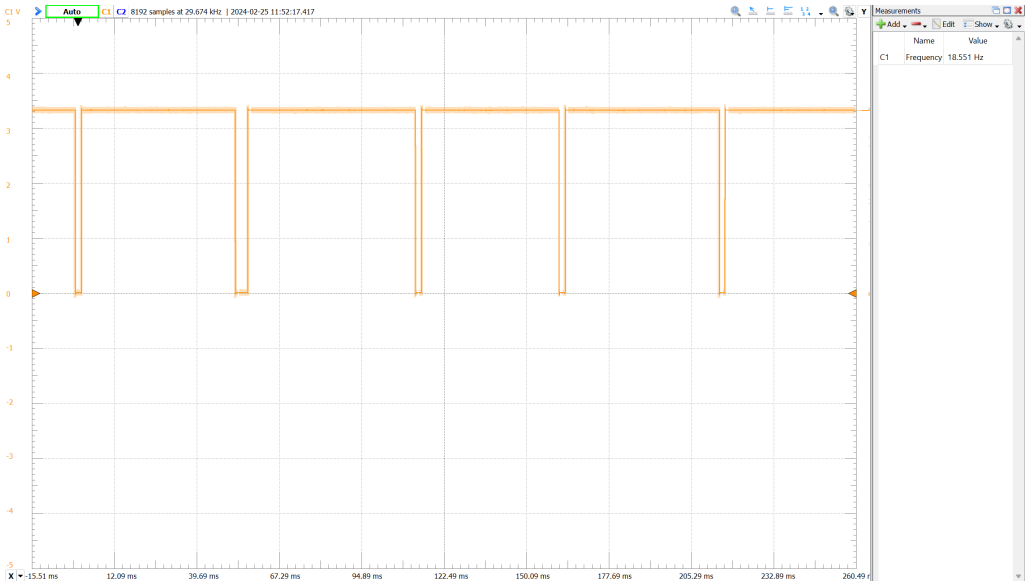


Figure 18: Frame Rate Measured with an Oscilloscope

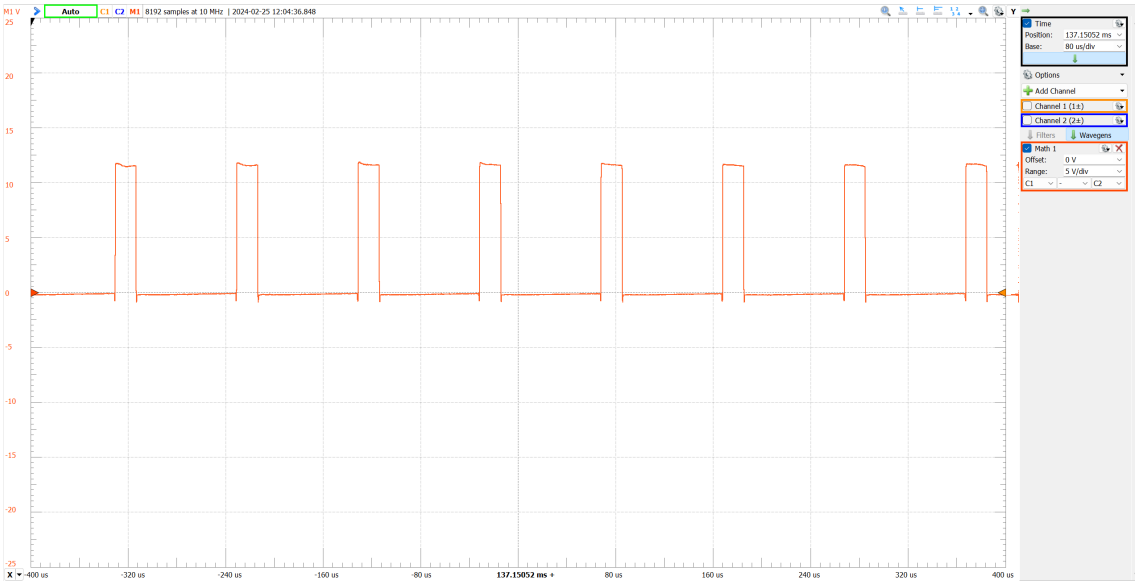


Figure 19: Differential voltage across DC motor.

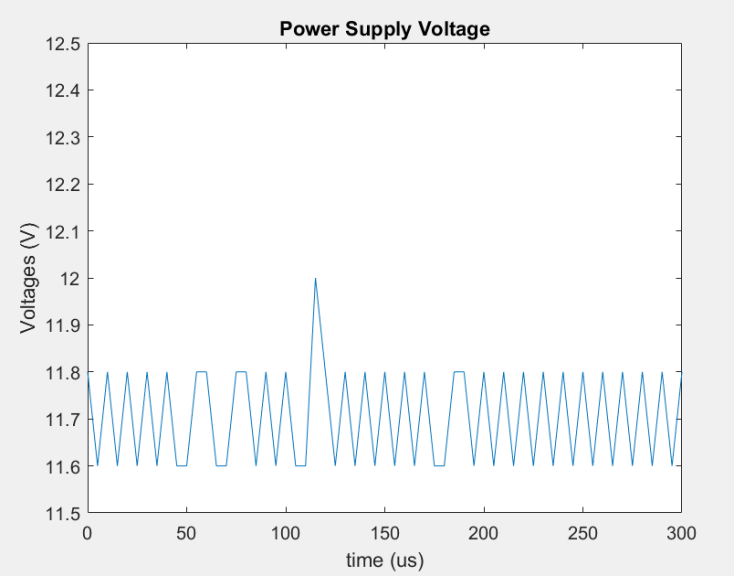


Figure 20: Power supply voltage.

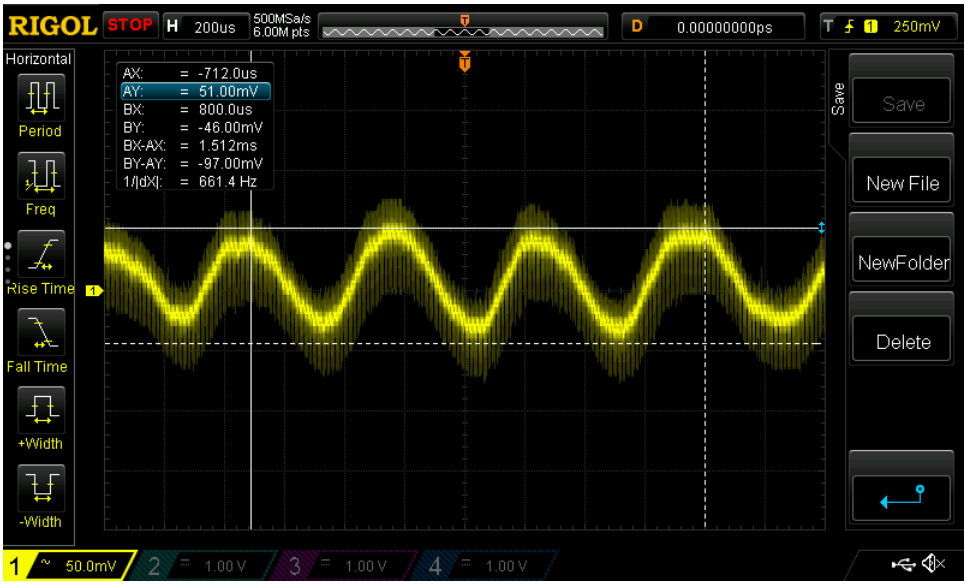


Figure 21: Power supply ripple voltage.

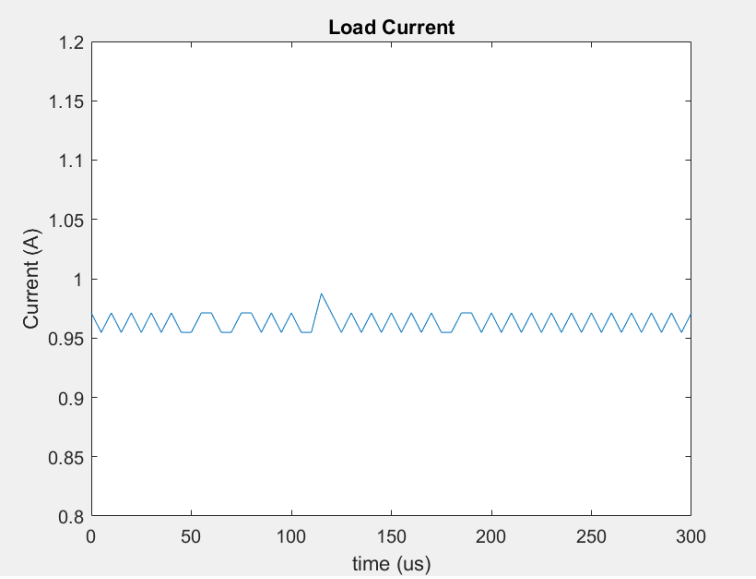


Figure 22: Resistive load current.

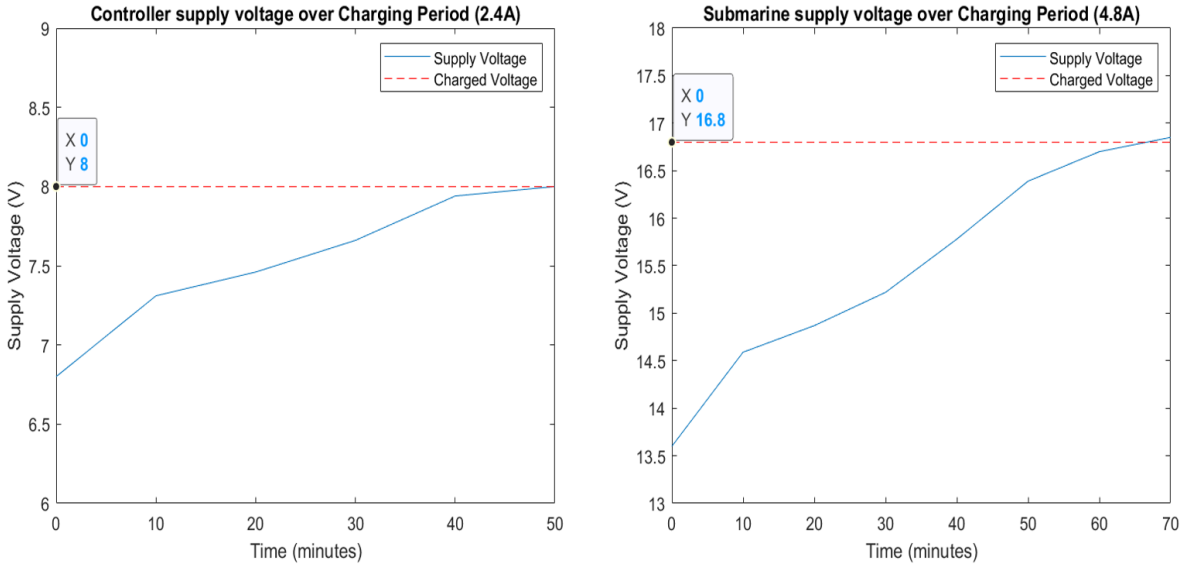


Figure 23: Graph of Submarine and Controller Recharge Time.

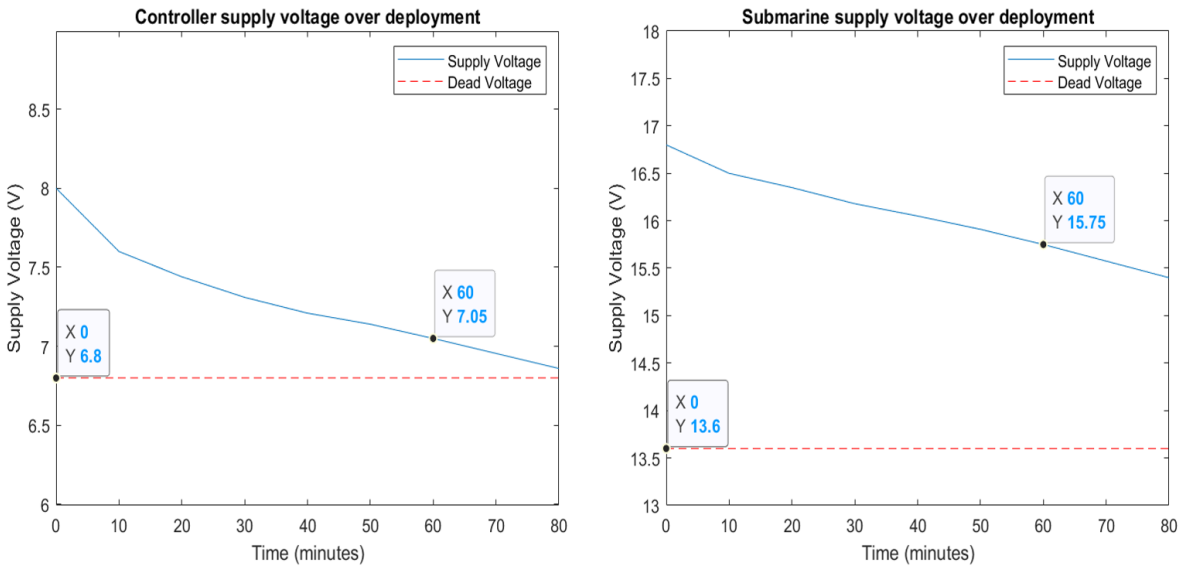


Figure 24: Graph of Submarine and Controller Discharge Time.

Appendix E Photos

Figure 25: Controller overhead.

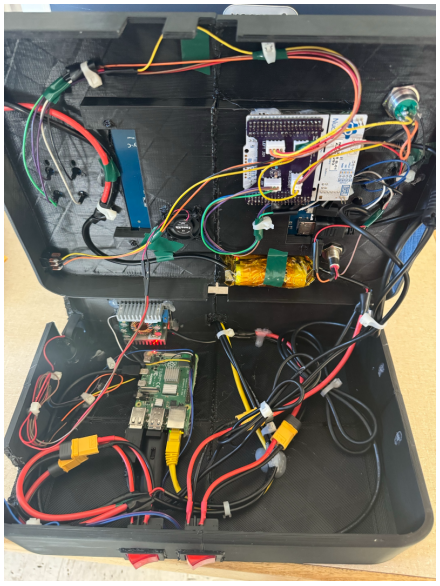


Figure 26: controller hardware.

Figure 27: Charging port and FPS measuring leads.

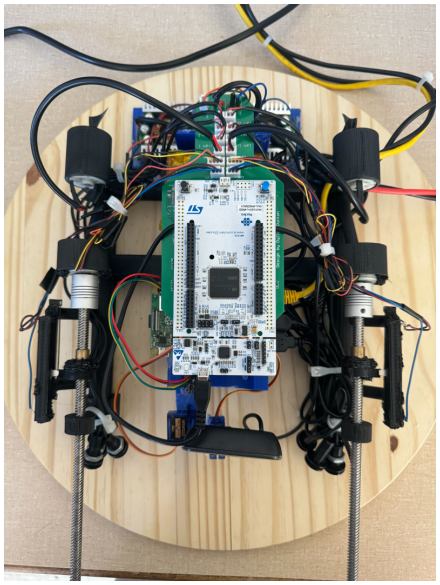


Figure 28: Submarine overhead.

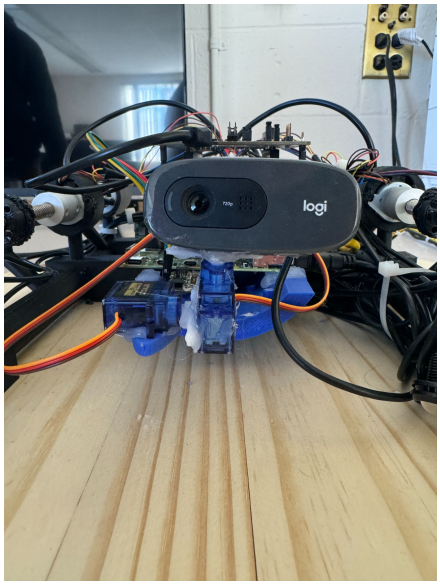


Figure 29: Camera gimbal.

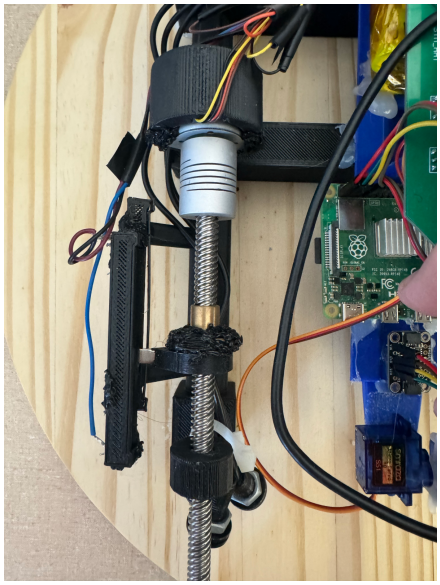


Figure 30: Ballast tank hardware.