

AquaPack Robotics Technical Design Report

Elizabeth Gillikin*, Abhiram Poosarla[†], Julianna White[‡], John Fetkovich[†], Achyuta Kannan[¶], Alex Ofsanik[§], Saranga Rajagopalan[†], Ashton Henderlite[†], Alexandria Epley*, Spencer Mol*

*Mechanical and Aerospace Engineering, [†]Electrical and Computer Engineering,

[‡]Nuclear Engineering, [§]Industrial and Systems Engineering, [¶]Computer Science

Abstract—AquaPack Robotics at NC State University is returning to RoboSub with SeaWolf VIII in 2025. Overall, AquaPack’s goal in the 2024-2025 design cycle has been to utilize the stability of the existing SeaWolf platform while incorporating new features and making software improvements to adapt our approach to updated competition tasks.

The lessons learned from the 2024 competition, as well as the introduction of new competition tasks for 2025, led us to add a Blue Robotics Ping360 Sonar to our vehicle and adjust our approach to tasks such as Navigate the Channel, Tagging, and Return Home. In addition, SeaWolf VIII’s torpedo launcher has been entirely redesigned following concerns of power and ease-of-reloading in 2024. SeaWolf VIII’s custom electrical system and octagonal frame have remained stable to support additional development of the acoustics and sensing systems. Overall, SeaWolf VIII has proven to be a platform suitable for continuous iteration, enabling the development of more complex subsystems.

I. COMPETITION STRATEGY

A. General strategy: reliability and refinement

Every year, AquaPack Robotics focuses on a reliability and refinement strategy for the current competition vehicle. For the past six years, that vehicle has been SeaWolf VIII. The team recognizes that a constantly changing system is unreliable, but a system that never changes is subject to repeated errors. Identifying systems in acceptable condition and those in need of improvement is necessary for continual progress to be made.

The 2024-2025 design cycle maintained that the electrical and mechanical systems of SeaWolf VIII were highly reliable. Continuous maintenance of these systems was prioritized over major alterations. We do not imply perfect electrical or mechanical systems, but these systems have proved to be a reliable platform off of which software and subsystem improvements can be made. Specific details on the current electrical and mechanical system are discussed in Sections II-B and II-A, respectively.

The software system and strategy demanded significant improvements. The chosen methods for software, discussed in Sections II-D and II-G, proved challenging to develop with the introduction of Task 2 - Navigating the Channel. To address this new task and supplement the completion of familiar tasks, an active sonar was added to SeaWolf VIII.

B. Heading Out

Before the start of the competition run, a coin flip is requested, which determines the starting orientation of SeaWolf VIII. Using computer vision (CV) and image recognition with our front-facing camera, SeaWolf VIII will locate the position of the gate.

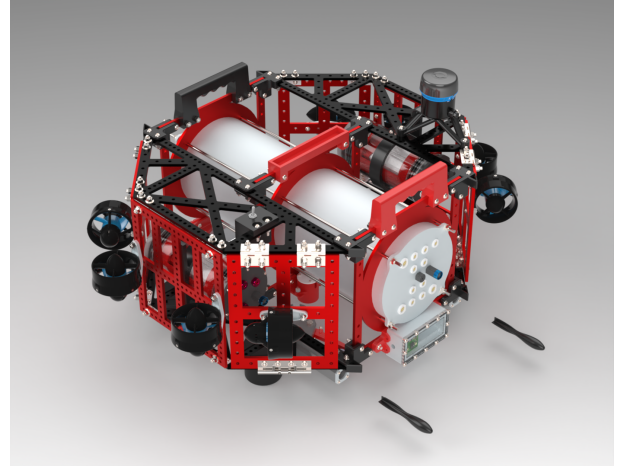


Fig. 1: SeaWolf VIII

C. Collecting Data

The competition run starts with successful navigation through the gate. Our custom-built flight controller enables six degrees of freedom locomotion, which makes navigating in a straight line through the gate and then performing a style rotation of yaw 720° achievable. Computer vision (CV) and image recognition with our front-facing camera allow us to align with the gate after a random starting orientation following the coin flip task. CV further enables us to navigate through our desired side of the gate. This task takes the greatest priority as navigating through the gate is required to qualify.

D. Navigate the Channel

After successfully qualifying through the gate, SeaWolf VIII interacts with the slalom. Utilizing our active sonar, SeaWolf VIII will detect the position of the pipes and move through each set while staying within the region without colliding with any of the PVC pipes.

E. Drop a BRUVS

The downward-facing camera on SeaWolf VIII detects the Path facing the bins. Collected images inform locomotion to center on the side of the bin corresponding to the correct destination. The downward-facing camera then detects the bin and drops a marker toward the open bin. This task takes priority in our strategy over torpedoes due to the reliability of our dropper system and simpler navigation that requires only the downward-facing camera.

F. Tagging

Navigating to the torpedo task requires passive sonar. Acoustics' passive sonar uses a Bartlett beamformer and custom register transfer language (RTL) to estimate an acoustic ping's angle of arrival (AoA). When close enough to the target to detect, CV data takes priority for navigation and informs robot locomotion. When map detection confidence is high, SeaWolf VIII fires one of its torpedoes at the general target. Due to the difficulty in navigation, reliability of operating the launchers, and accuracy of CV to align with targets, this task is of lower priority during a competition run.

G. Collect Samples

Once SeaWolf VIII completes all other tasks, it uses the passive sonar system to locate the octagon's pinger and the downward-facing camera to detect and align to the table in the center of the octagon. Without a grabber manipulation system, SeaWolf VIII's only task is surfacing inside the octagon.

H. Return Home

SeaWolf VIII uses active sonar to locate the position of the starting gate and re-submerge to retrace its movements. SeaWolf will navigate back through the gate to end the competition run.

II. DESIGN STRATEGY

A. Mechanical system

The design of SeaWolf VIII, seen in Fig. 1, places an emphasis on control stability and modularity. The vehicle layout was selected in a way that allows fixed thrusters to achieve six degrees of freedom without relying on gimbals or hydrodynamics. A repeating hole pattern on the modular aluminum frame supports continuous mounting and modification, meeting physical needs at any time.

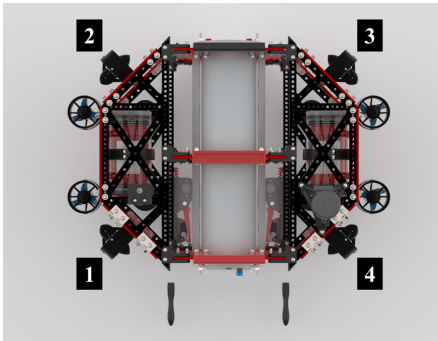


Fig. 2: Top view of SeaWolf VIII with numeric thruster labels. To either side of the central hull are bays for peripheral testing and mounting.

1) *Control stability and design shape:* The octagonal shape of SeaWolf VIII achieves control stability. Thrusters are farther from the center of mass, increasing their lever arm to counteract SeaWolf VIII's comparatively high mass moment of inertia. The shape, depicted in Fig. 2, has thrusters 1-4 placed in a strafe configuration. These thrusters are at 45° angles relative to the center of Sea Wolf VIII, enabling the four thrusters to all contribute to horizontal motion in any direction.

2) *Modularity:* A standardized hole pattern on SeaWolf VIII's frame and large bays on either side of the hull support a wide range of possible configurations. The standardized hole pattern allows for easy mounting of any peripheral component nearly anywhere on the robot without frame modification. Peripheral designs conform to the hole pattern depending on the required mounting location. The large bays on either side of the electronics hull are accessible through hinged panels, providing ample space for any new subsystem during a testing phase or final deployment. This space is also visible on either side of the central hull in Fig. 2.

B. Electrical system

1) Main Electronics Board:

The Main Electronics Board (MEB), as seen in Fig. 3, is the central interface for the communications, power, and sensor systems on SeaWolf VIII. It connects directly with the Jetson Nano via a UART connection, and with all other electrical system boards via a unified I2C bus. Thus, MEB acts as a

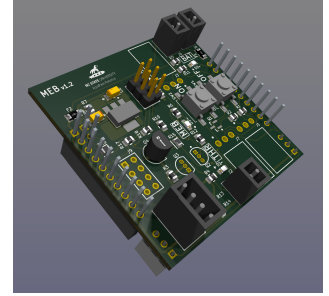


Fig. 3: Render of Main Electronics Board. Generated with KiCad.

buffer between the on-board computer and the rest of the robot. This organization enables SeaWolf VIII to effectively manage various sensors and boards situated throughout the main hull of the AUV without putting additional computational load on the on-board computer.

2) *Power System:* Safety and stability are the guiding principles of the SeaWolf VIII electrical architecture (see Fig. 11 in Appendix A). This starts at our power source; The robot is powered by two 4S Lithium-polymer (LiPo) batteries, each fused at 40A. If a short circuit develops in the system, the current will spike and destroy at least one of the fuses, mitigating potential damage. This configuration balances the need for safety with the power needs of the robot, as the combined 80A provides enough overhead to allow power stability and room for further expansion. To further ensure safe operating conditions, the two external hulls holding the batteries each contain a standalone leak sensor module to allow early detection and notification of water leakage.

Immediately downstream of the batteries is a load balancing circuit that leverages an ideal diode controller to safely connect the batteries in parallel. It also provides reverse polarity protection, which protects the electronics in the event a battery is plugged in incorrectly.

After the load balancing circuit, power is provided to the system using two Solid State Relays (SSRs) – the system SSR and thruster SSR. The system SSR is responsible for switching battery power to the entire system. When the vehicle is powered on via an external switch, MEB is the first component to receive power. MEB then drives system SSR to supply

power to other systems, including the Jetson Nano, acoustics system, thruster SSR, and other peripherals. Thruster SSR is enabled only if the vehicle’s computer generates a software arm signal and the physical kill-switch is in the “armed” position. Importantly, the kill-switch is connected directly in series with the software arm circuitry, ensuring that electrical component failures cannot prevent the ability to kill the vehicle with the hardware switch. Finally, various components on SeaWolf VIII require power regulation. SeaWolf VIII uses a power distribution architecture in which each board has a dedicated regulator, which allows rapid development with minimal disruption to the rest of the system. The system provides several identical 5V buck regulators for use by the acoustics system, auxiliary boards, USB hub and any future additions that may require them. The active sonar accepts a wide range of voltage inputs, but is powered by a 12V buck-boost regulator to isolate the device from dips in battery voltage. The Jetson Nano uses a more complex regulator architecture due to its high current requirements and sensitivity to voltage drops. First, power is regulated to 12V using a SEPIC converter, which can maintain a steady output when the input voltage suddenly drops. This is necessary because the battery voltage occasionally sags due to thruster in-rush current. The 12V is then regulated down to 5V using an additional buck converter. This additional converter is required to provide the lower voltage and higher current the Jetson Nano requires.

C. Locomotion controls

Our custom control board handles vehicle locomotion. It is a custom motion controller using an Arm Cortex M4F micro-controller. It acts as a motion co-processor, allowing mission code running on the vehicle’s computer to describe motion in various high-level schemes. This co-processor design ensures the computer spends minimal processing time on motion and ensures control loop stability due to the deterministic nature of timings on the control board.

The control board uses a Quaternion-based approach similar to what can be used with a quadcopter [1]. A Quaternion-based approach allows numerically stable control of orientation in 3D space [2] without the potential for losing degrees of freedom that accompany Euler angles [3]. Added to this is a PID controller to maintain depth and tilt (pitch and roll) compensation to allow the description of motion in a partially world-relative 2D plane parallel to the surface. This approach ensures that slight pitch or roll errors do not result in unexpected motions.

This motion description also abstracts the vehicle’s nature to mission code, allowing the code to be easily used on different vehicles or in a simulator.

D. Software architecture

1) *Action Tree and Technologies:* The higher-level mission code that interfaces with the MEB and control board to execute tasks is written in Rust, using action trees. As depicted in Fig. 4, actions are defined at compile time in a tree from the starting to the ending action, with various operations

to reach the end state. Actions can be nested, chained, run in parallel, conditionally execute branches, and modify the execution of the action itself. Actions can perform more functions than those listed, but we primarily use those listed. We demonstrated the success of this architecture at RoboSub 2024 where we crafted and tuned missions which executed reliably and with minimal failures.

Writing our action trees in Rust provides robust tooling to optimize code, a borrow checker that prevents segmentation faults during runtime, and comparable runtime speeds to C/C++. Action trees clearly define our actions and their transitions at compile time, with the ability to show various kinds of execution such as parallel, concurrent, and conditional execution. We generate tree visualizations by converting the tree structure into a Graphviz DOT file rendered as an SVG. Fig. 4 shows how this is rendered, with the “AlwaysTrue” conditional taking the place of a conditional action that may change at runtime. These graphs are currently auto generated with each build and published to our internal documentation site.

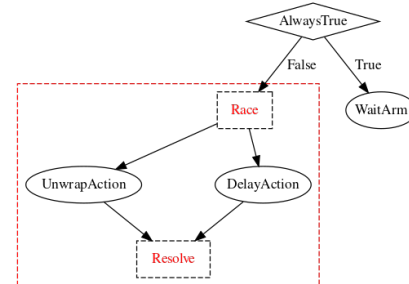


Fig. 4: Example action tree which shows a race conditional under the case of “AlwaysTrue”.

2) *Communication:* A discrete communications manager handles all communication with external systems. To prevent system stalls from explosive thread growth, it is allocated to a static number of threads to send and receive messages from the control board. All serial messages generate an asynchronous task that returns true when an acknowledgment is received, allowing a wait for success. Specific messages are sent to the control board to take specific actions. The control board treats other messages as requests that return information about the robot’s current state. For example, we can command the robot to submerge to a certain depth and subsequently request its depth. This interface is abstract enough that a change to the control board would not require changes to the high-level state machine. Currently, it allows for the same communication between an actual system and simulation, with the actual system communications routed through UART and the simulation through TCP. The vision system runs a GStreamer connection to share camera feeds. One pipeline offers an RTSP stream for live footage during robot testing, while another records to disk so real pool images can be used for vision model testing. The vision processing code uses the last pipeline to move the robot according to the targeted game object.

3) *Building, deployment, and version control:* Cargo is Rust’s native build system, package manager, and unit testing

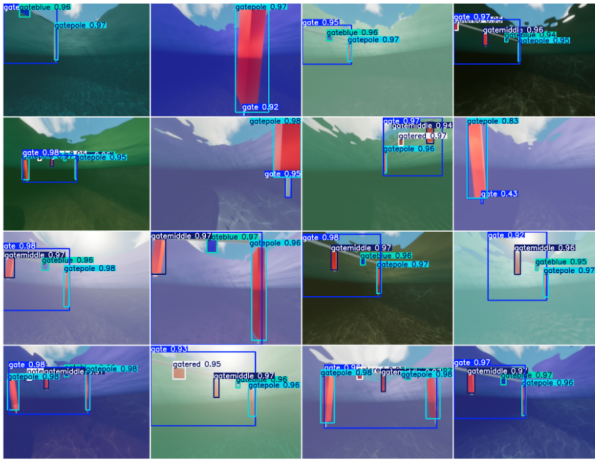


Fig. 5: Identification of gates in various simulated environments using ML

suite. It contains code to compile and interface with Nvidia Compute Unified Device Architecture (CUDA) [4] code. SeaWolf VIII's Nvidia Jetson Nano uses an ARM architecture CPU and a specialized operating system, which complicates compilation. To solve this, we developed a cross-compilation tool that allows us to easily compile code for the Jetson from any computer. We use a containerized development environment to allow any member to quickly set up their device and to ensure more reproducible builds.

All code changes are pushed to our central GitHub repository. We create branches for each new feature addition and each pool test. We have made extensive use of continuous integration to automatically build and unit test our code to ensure the quality of the code we deploy to SeaWolf VIII.

E. Sensing

SeaWolf VIII's sensing systems use both traditional and machine learning (ML) based computer vision as well as a Blue Robotics Ping360 Sonar. Task complexity determines which approach we utilize for a given mission. We utilized a traditional CV approach for tasks with simple polygonal shapes, an ML approach for tasks with complex features, and sonar for sensor fusion between these two where depth data is needed. Detection of the Path with traditional CV is depicted in Fig. 7 with the object identified and the direction of travel shown by an arrow. Fig. 5 shows a ML identification of gates in a simulated competition environment. Fig. 6 shows objects identified from sonar data using HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) clustering.

1) *Traditional CV*: Edge and line detection is the foundational technique our traditional CV uses. Edge and line detec-

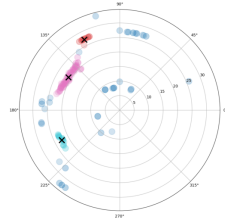


Fig. 6: Identification of objects in sonar data via clustering

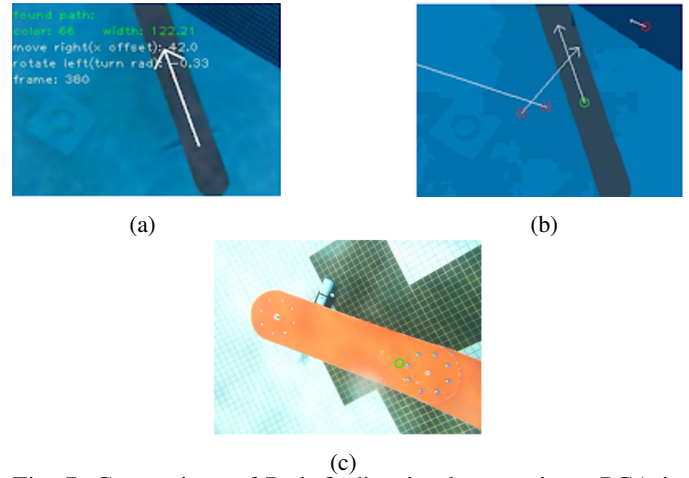


Fig. 7: Comparison of Path-finding implementations: PCA in Python (a), PCA in Rust (b), and minimum area rectangle (not drawn) in Rust (c)

tion are image processing techniques that distinguish outlines and line segments in an image. This design was referenced from software developed for RoboSub 2022, as this method worked well. We rewrote the algorithm for better integration into SeaWolf VIII's Java-based software architecture in ML-2023. We again rewrote the algorithm in Rust [5] to align with our recent shift to the language. The adapted model, seen in Fig. 7, reduced images to 4 RGB colors by combining localized and global K-means to segment an image more consistently. We obtained location and directional information using Principal Component Analysis (PCA) with color and size filters to determine each color's mean and covariance. After these numerous rewrites, the PCA based algorithm was replaced with a simpler one based on contour detection and minimum area rectangles. This algorithm is easily understood and can be tuned with only two parameters: a color and area range. Features that fall within the specified color and area ranges are identified, then the feature with the greatest area is selected. Next, a rectangle with the smallest possible area is constructed around the feature, allowing us to extract a center position and angle.

2) *ML*: For images with complex features, neural networks were more suitable. They enable extracting distinct features in complex images, such as torpedo targets. We used YOLOv5 nano [6] as our model and developed a Unity [7] simulator to generate photorealistic images of tasks to train the model, as in Fig. 5. The simulator can generate thousands of auto-labeled, synthetic images in different environments. We developed the ML model using Rust OpenCV [8] and ONNX [9] formats. We use CUDA kernels for pre and post-processing, which offloads computation from the CPU to the GPU. This offload leads to a speed-up in the overall vision pipeline since we are far from full saturation of the CUDA cores on the Jetson Nano's GPU. The CPU is thus able to process more frames per second.

3) *Sonar*: Sonar is best suited for tasks where distance data is required, vision systems perform poorly, or long range detection is required. We first apply a filter based on an estab-

lished noise floor, then apply the HDBSCAN [10] clustering algorithm to extract objects from individual sonar data points (See Fig. 6). Our sonar unit offers a sub-2 millimeter range resolution, allowing us to determine the width of clusters with very high accuracy. We can then interpret this data based on the known measurements of the tasks and the current context. We also vary the scan range and resolution to those best suited for the task at hand. For close range navigation, we can perform quick, high-resolution scans. For longer range scans, we lower the resolution to maintain an acceptable speed.

F. Manipulations systems

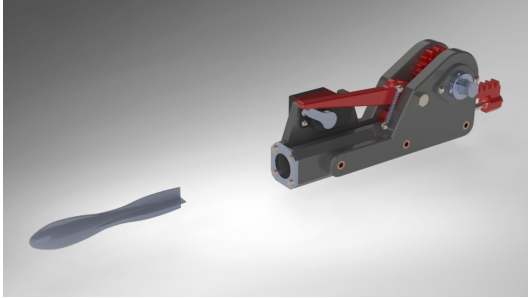


Fig. 8: New SeaWolf VIII torpedo launcher and torpedo. Design mock-up generated in Solidworks.

1) *Torpedo*: SeaWolf VIII utilizes two spring-loaded torpedo launchers that consist of a combination of FDM and machined components. The mechanical design of the launcher allows the spring to be loaded with a standard wrench that drives a rack and pinion mechanism. A ratchet gear and pawl prevent counter-rotation as the spring is loaded and hold the launcher in the loaded position until the robot is ready to fire. The mechanism is released using a watertight servo motor, which disengages the pawl from the ratchet gear, allowing the spring to unload and fire the torpedo. This reload mechanism is a significant improvement over the previous iteration, which required a full shutdown of the robot and limited the power of the spring we could feasibly use. Several FDM printed projectiles were designed, with the chosen fusiform shape performing most favorably. Fig. 8 shows the chosen subsystem design. It is simple, minimizes the required amount of additional electrical infrastructure, and enabled iterative prototyping. This allowed for rapid development using an FDM printer while still meeting task requirements.

2) *Dropper*: The dropper mechanism for SeaWolf VIII consists of three major components: a 5V electromagnet, a magnetic 440C stainless steel ball, and a 3D-printed housing. The electromagnet is seated in the upper portion of the housing via a screw-on lid. The stainless steel ball, or marker, rests in the lower



Fig. 9: SeaWolf VIII dropper enclosure with electromagnet and marker stored internally. Design mock-up generated in Solidworks.

part of the housing. Current continuously runs through the electromagnet to create a magnetic field, which holds the marker secure in the housing while SeaWolf VIII is powered on. When the current stops, the marker falls out of the bottom of the housing. Fig. 9 depicts one of a pair of these droppers, which are attached to the bottom of SeaWolf VIII.

G. Acoustics

Acoustic navigation via the passive sonar system requires two central components: analog pre-processing and digital signal processing. Both components work in tandem for our acoustic system to produce reliable results for the robot to use for navigation.

1) *Signal capture and pre-processing*: The acoustic signal from the pingers occupies 25kHz-40kHz frequencies. This pure tone signal motivates a simple pre-processing system, which accounts for signal attenuation, white noise, and other audible frequencies. Signal capture uses a linear array of four hydrophones spaced at 2.5 cm apart. The hydrophones are phantom-powered. The system includes buffer circuitry, isolating the hydrophones from the rest of the system, and biasing circuitry, which removes the need for a negative voltage rail by using a 0dB gain op-amp circuit to change ground reference to the half supply voltage. A 10.4dB gain non-inverting op-amp circuit is used as a pre-amplifier to ensure a larger input signal capture. It is fed to four cascaded Chebyshev bandpass filters with a peak gain of 0dB or small attenuation. Without significant pass band gain in the filtering stage, the pre-amplifier becomes necessary, as small attenuation in the pass band can compromise the signal integrity of small signals captured at hydrophones. We use a digitally controlled linear amplifier, the LTC6910, for post-amplification. This stage amplifies the filtered analog waveform such that the peak-to-peak voltage occupies the entire 5V range of the analog-to-digital converters (ADC) that follow.

Different amplification is necessary because path loss will change with distance by the inverse square law as we approach the pinger. The entire response of the system is depicted in Fig. II-G. The Chebyshev filter gives a steep frequency cutoff at the corner frequencies of 25 KHz and 40 KHz ensuring the only received signals are those in the band of interest that are used by RoboSub.

2) *Digital signal processing*: We deploy digital signal processing (DSP) on a Digilent Basys 3 FPGA development board. The onboard FPGA, Xilinx Artix-7, contains numerous DSP cores and several intellectual property cores that we use in development.

The base system is a Bartlett direction of arrival estimator, similar to what is described in [11]. The competition environment only has a singular signal source active at a given moment, implying that we do not need to estimate the angle of arrival for several signals in a single time window. This lends nicely to the Bartlett beamformer as it is simple to implement, but is only very successful with few signal sources [12]. The resolution of the Bartlett is also lesser than the

comparable direction of arrival estimators such as minimum variance distortionless response (MVDR) and multiple signal classification (MUSIC), but with only a singular source at a time, this weakness matters less, leading us to favor the Bartlett strategy which is simple to implement requiring comparably very few matrix operations.

III. TESTING STRATEGY

A. General testing strategy

Our approach to testing our subsystems requires simulation-based and in-field testing to validate that our systems work. Simulation-based testing pinpoints flaws and ensures that our designs provide expected results. In-field or “pool testing” validates all systems proven to operate successfully via simulation. We aimed to have at least two monthly pool tests to ensure sufficient testing of our custom control board and software architecture. Using a replica of the RoboSub course at pool tests allows the team to observe progress and re-evaluate goal timelines. See Appendix B.

B. Software and CV

Automated tests and continuous integration checks validate system functionality. Through a combination of simulation and field testing, we evaluate the performance of our software and identify areas of concern. We use simulations to train and tune our CV algorithms as described in II-E1. The Unity engine [7] is our choice of simulation environment. This simulator’s ability to generate thousands of photorealistic images in various environments in minutes enables a quicker turnaround for tuning our software. See Appendix B-III.

C. Manipulation

1) *Torpedo*: Once assembled, the torpedo system was fired repeatedly in and out of the water to ensure independent actuation of each servo and characterize projectile motion in water. This method validated the design and identified an optimal projectile shape. See Appendix B-IV.

2) *Dropper*: The dropper electromagnet was connected to a DC power supply at 5V and tested repeatedly in dry and wet environments. These tests were successful and repeatable. Pool testing was also successful, with the marker staying stable in the housing through various underwater maneuvers and deploying consistently in the pool.

D. Acoustics

All testing of the acoustical direction-of-arrival (DOA) system was performed through theoretical verification and both digital and physical simulation. Utilizing MATLAB [13], sinusoidal waveforms were generated at our test frequency and discrete additive white Gaussian noise was inserted to model a realistic operation environment. In code, the simulated arrival angle of the sinusoid is controllable, allowing us a benchmark for the performance of the Bartlett beamformer DOA algorithm at all possible angles. The Bartlett beamformer proved accurate at numerous angles, but would see spatial aliasing toward broadside angles.

The implementation of this algorithm, shown in Fig. 10, involved writing and integrating Verilog modules. Utilizing

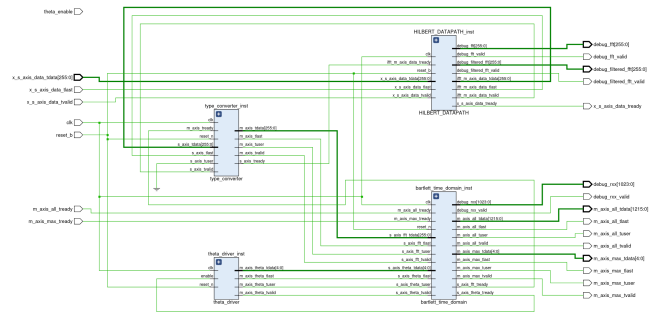


Fig. 10: Bartlett Beamforming DOA Algorithm

AXI4-Stream, an efficient pipe-lined digital signal processing design was created. With a data-path to execute the beamformer, fixed-point hardware-specific testing was able to be carried out. A combination of simulation in Vivado and ideal data from MATLAB allowed for a large range of test points to be compared. The results allowed for verification of the algorithm on an Artix-7 FPGA. Inherently, tests are limited to giving insight into how the system reacts to specific circumstances, but a large volume of testing instills the confidence in the system necessary for deployment.

IV. ACKNOWLEDGEMENTS

AquaPack Robotics is housed within North Carolina State University’s Electrical and Computer Engineering (ECE) department. We would like to thank all of the faculty and staff who have supported and continue to support the club. We extend a special thank-you to Dr. John Muth for serving as our advisor, as well as the Casey Aquatic Center at Carmichael Gymnasium for providing us a facility for pool testing. Lastly, we would like to extend our gratitude to our sponsors for providing us financial support and access to their technical products and software, which allows us to continue to develop and improve our robot each year. Our 2024-2025 sponsors are Analog Devices Inc., BAE Systems, Cadence, Caterpillar, NC State Engineer Your Experience, The Timken Company, Rooh, Onshape, and Solidworks. We would also like to give a special thanks to our individual donors David & Meg Gillikin (NC State Alum) and Alex Pendergast (AquaPack Robotics Alum).

REFERENCES

- [1] E. Fresk and G. Nikolakopoulos, “Full quaternion based attitude control for a quadrotor,” 2013.
- [2] B.-U. Lee, “Unit quaternion representation of rotation,” Ph.D. dissertation, Stanford University, 1991.
- [3] R. Pio, “Euler angle transformations,” *IEEE Transactions on Automatic Control*, vol. 11, no. 4, pp. 707–715, 1966.
- [4] NVIDIA, P. Vingelmann, and F. H. Fitzek, “Cuda, release: 10.2.89,” 2020. [Online]. Available: <https://developer.nvidia.com/cuda-toolkit>
- [5] N. D. Matsakis and F. S. Klock II, “The rust language,” in *ACM SIGAda Ada Letters*, vol. 34, no. 3. ACM, 2014, pp. 103–104.
- [6] Ultralytics, “ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation,” <https://github.com/ultralytics/yolov5.com>, 2022, accessed: 7th May, 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7347926>

- [7] U. Technologies, “Unity real-time development platform: 3d, 2d, vr ar engine,” accessed: 1st July, 2024. [Online]. Available: <https://unity.com/>
- [8] Itseez, “Open source computer vision library,” <https://github.com/opencv/opencv>, 2015.
- [9] J. Bai, F. Lu, K. Zhang *et al.*, “Onnx: Open neural network exchange,” <https://github.com/onnx/onnx>, 2019.
- [10] C. E. al., “Density-based clustering based on hierarchical density estimates,” *Advances in Knowledge Discovery and Data Mining*, vol. 7819, 2013.
- [11] M. Abusultan, S. Harkness, B. LaMeres, and Y. Huang, “Fpga implementation of a bartlett direction of arrival algorithm for a 5.8ghz circular antenna array,” 04 2010, pp. 1 – 10.
- [12] I. A. H. Adam and M. R. Islam, “Perfomance study of direction of arrival (doa) estimation algorithms for linear array antenna,” in *2009 International Conference on Signal Processing Systems*, 2009, pp. 268–271.
- [13] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

Appendix A: Component Specifications

Component	Vendor	Model/Type	Specs	Qty.	Custom/Purchased	Cost	Year Acquired
Waterproofing							
Main Hull	OnlineMetals	8" Aluminum Tube	0.25 x 25.75 in	1	Purchased	\$165.56	2022
Battery Hull	Blue Robotics	4" Watertight Enclosure	100 mm x 200 mm	2	Purchased	\$638	2023
Main Hull Endcap	Mecha Inc	–	8" 6061 Aluminum	2	Purchased	\$114.94	2022
Camera Enclosure	McMaster-Carr	–	–	2	Custom	\$33.2	2023
Waterproof Connector Plug	Fischer	S Series	–	22	Purchased	\$750	2022
Waterproof Connector Receptacle	Fischer	DEU Series	–	22	Purchased	\$750	2022
Electronic/Power System							
Load-Balancing Board (LBB): Ideal diode controllers	Digikey	LTC4359CMS8	150 μ A/4V-80V	2	Purchased	\$6.43	2021
Load-Balancing Board (LBB): MOSFET	Mouser Electronics	IXTN660N04T4 - I	40V/660A	2	Purchased	\$32.17	2021
Main Electronics Board (MEB): Launchpad	Texas Instruments	MSP430G2553	1.8 V-3.6 V	1	Purchased		2022
LiPo Battery	Gens Ace	GEA10K4S10E5	15.2V/10000mAh/100C	2	Purchased	\$154.99	2023
UBECs	SoloGood	–	5/3A Brushless Receiver Servo	3	Purchased	\$12.99	2023
Manipulators							
Dropper	–	3D Printed	PETG	250g	Custom	\$5.75	2023
Electromagnet	Adafruit	5V Electromagnet	5 Kg holding force	2	Purchased	\$9.95	2023
Torpedo Launcher	–	3D Printed	PETG	450g	Custom	\$11.50	2025
Servo Motor	Zoskay	High Torque Metal Gear Servo	25KG hold force, 6.8 V	2	Purchased	\$17.99	2025
Mechanical Systems Board (MSB): Microcontroller	Texas Instruments	MSP430FR2355	16-bit/24MHz	1	Purchased	\$12.99	2022
Controls							
Control Board: Microcontroller	Adafruit	ItsyBitsy	512 KB flash, 192 KB RAM32-bit Cortex M4 core	1	Purchased	\$14.95	2022
Control Board: IMU	Adafruit	BNO055	9-DOF sensor, ARM Cortex-M0 based processor	8 1	Purchased	\$34.95	2022
Thrusters	Blue Robotics	T200	Brushless DC motors	8	Purchased		2022
ESCs	Blue Robotics	Basic	7-26V/30A	8	Purchased	\$36	2018
Acoustics							
Active Sonar	Blue Robotics	Ping360	50m range, 300m depth, 5W	1	Purchased	\$2,750	2022
Hydrophones	Aquarian Audio	H2C	10 Hz-100kHz Range	4	Purchased		2019
Power Distribution	–	Custom PCB Solution Rev. 2	4 Way Distribution, 1 A/Channel, 5 Vdd, 2.5 V AGND		Custom	≈\$5	2022
Acoustics Front End	–	Acoustics Single Channel Rev. 1.2	500 KSpS, 25kHz-40kHz BPF, 20dB-60dB passband amplification	1	Custom	≈\$20	2022
Digital Signal Processing Unit	Digilent	Basys3	Xilinx Artix-7 FPGA, 90 DSP Slices, 1800 Kbits Block RAM	1	Purchased	\$165	2022
Software Architecture							
Operating System	Qengineering	Ubuntu 20.04	–	–	Open-Source	\$0	2022
Primary Language	Rust Foundation	Rust 1.79.0	–	–	Open-Source	\$0	2023
Development Language	Python Foundation	Python 3	–	–	Open-Source	\$0	2022
Serial Communication	Tokio	Tokio-serial	–	–	Open-Source	\$0	2022
Automated Testing	junit-team	Junit 4.13.2	–	–	Open-Source	\$0	2022
Deployment	OpenSSH	ssh	–	–	Open-Source	\$0	2022
Build Tool	Rust Foundation	Cargo 0.80.0	–	–	Open-Source	\$0	2022
Video Server	Gstreamer Team	Gstreamer 1.2.0	–	–	Open-Source	\$0	2023
Video Processing	OpenCV	OpenCV 4.6.0	–	–	Open-Source	\$0	2022
Vision							
Cameras	ArduCam	IMX219	4K 8MP	2	Purchased	\$34.99	2023
OpenCV	Big Vision LLC	4.6.0	–	–	Open-Source	Open-Source	2023
YOLO	Darknet	v5	–	–	Open-Source	Open-Source	2023
Frame							
Perforated Aluminum Side 1	Custom KB Metalworks	–	8 x 10.48 x 0.250 in		Custom	Donated	2018
Perforated Aluminum Side 2	Custom KB Metalworks	–	8 x 10.48 x 0.250 in		Custom	Donated	2018
Hull Cradle	Custom KB Metalworks	–	8 x 10.48 x 0.250 in		Custom	Donated	2018
Main Hull Threaded Rod	McMaster-Carr	–	8 x 10.48 x 0.250 in		Purchased	9	2018
Interchangeable Central Platform							
Acrylic Backplane	McMaster-Carr	–	.25" Cast Acrylic	1	Purchased & Customized	\$36	2023
Acrylic Truss	McMaster-Carr	–	.25" Cast Acrylic	4	Purchased & Customized	\$14.40	2023
Rings	–	3D Printed	PETG	1250g	Custom	\$28.74	2023

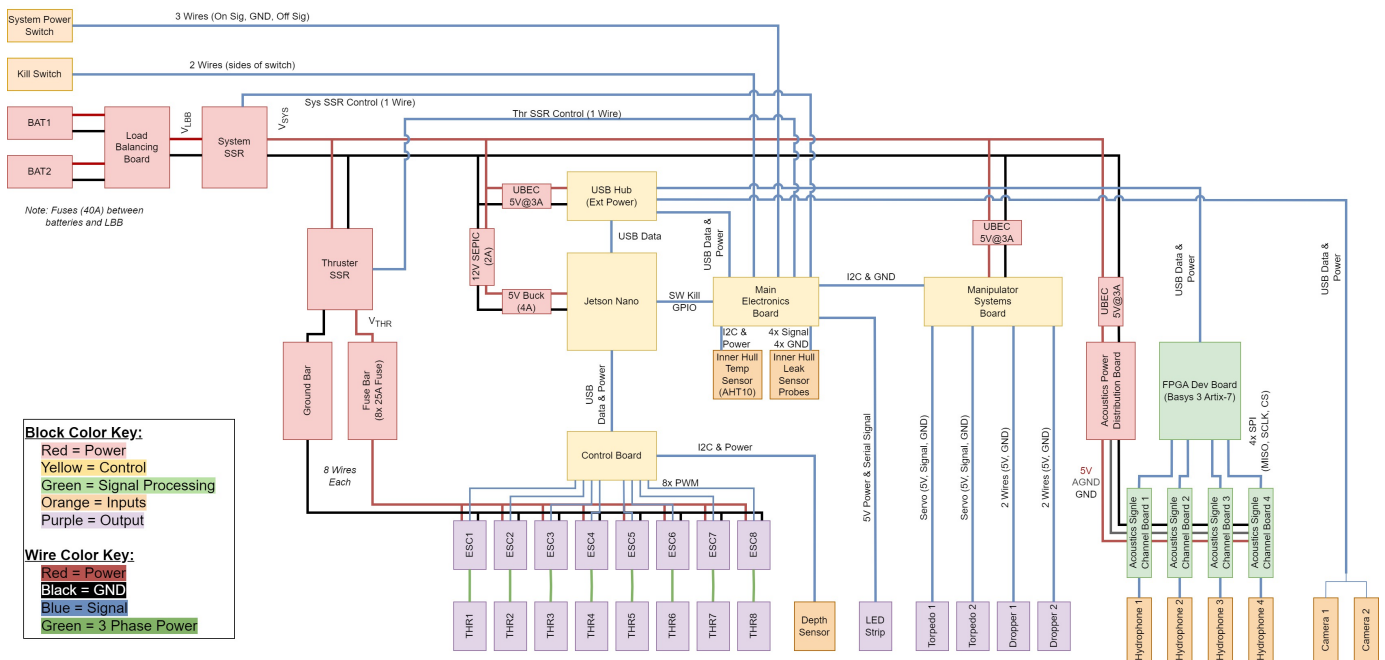


Fig. 11: Electrical System Block Diagram

Appendix B: Test Plan and Results

I. TEST SCHEDULE

We performed seven major types of testing to debug and validate our systems. These tests included our navigation systems, manipulation systems, waterproofing measures, and full system tests. A legend of what our main protocols for simulation and testing is listed in Table I below.

TABLE I: Legend of System and Simulation Test

Test/Simulation	Application
Acoustics Simulation and Testing	Consists of testing how well acoustics circuits can detect and process pings in simulation and at pool tests. This helps determine if acoustics system can detect and process pings in an environment similar to the RoboSub competition.
CV Simulation	Consists of running mission and vision code through a simulated environment consisting of a pool, robot, and tasks at RoboSub. Each test consists of both running simulation and debugging. On many occasions issues found in code was debugged/improved outside of that time frame and simulated again.
Leak Tests	Consists of sealing the robot and vacuum testing its main hull and battery hulls. For each test, a pressure of -25 mm/Hg was held for a specific duration of time. Vacuum was held on the main hull for 40 minutes and each battery hull for 10 minutes. Leak tests were performed before each pool test. In the event that there appears to an issue with air leakage during a leak tests longer tests are held.
Locomotion Simulation	Consists of development and testing of simulation to validate math and orientation of custom control board. Used to identify issues with control board and correct them without in-water testing time.
Manipulation Systems Test	Consists on testing each mechanical system via its trigger to determine if system works.
Pool Tests	Consists of testing SeaWolf VIII in pool. Tests conducted at pool tests include locomotion, acoustics, CV, mission code, manipulation systems, sonar data collection, and full system tests.
Sonar Data Collection Tests	Consists of stabilizing SeaWolf VIII in different areas of the pool facing different obstacles for data collection. Tests are conducted at different ranges, altered frequencies, and varying obstacles to fine tune the sonar module. Conducted to create a sufficient data pool for the software team to develop algorithms for missions.
System Dry Run	Will occur before each pool test and after major changes to the electrical system have been made. Longer sessions would consist of doing checks on all of the electrical subsystems to ensure expected output was occurring. Thrusters are also run to ensure proper communication and determine if re-calibration is necessary. Typically 30 minutes each.

Table II presents the number of hours spent simulating and testing various systems and operations of SeaWolf VIII. This table includes planned hours of simulation and tests as well. Details of what each test entails is listed above in Table I.

TABLE II: Hours Spent Simulating and Testing Systems

	To-Date	Planned
Acoustics Simulation and Testing	20	20
CV Simulation and Testing	60	10
Leak Tests	24	3
Locomotion Simulation	50	2
Manipulation Systems Test	5.66	3
Pool Tests	74	12
Sonar Data Collection	8	12
System Dry Run	14	1.5
Total	255.66	93.5

Locomotion testing and validation were conducted during the fall semester as completion of missions are not possible without proper orientation calculations and reliable communication between the control board, MEB (Main Electronics Board), and the computer. Sonar data collection and further development of software architecture and communications were prioritized during the spring semester. Footage of props was collected for the purpose of testing simulated CV algorithms on real test cases. Summer pool tests prioritizes testing of missions and competition runs. Table III lists all completed and planned pool tests for the 2024-2025 academic year.

TABLE III: Completed and Planned Pool Test Dates

Semester	Fall	Spring	Summer
Dates	October 6th, 2024	January 26th, 2025	June 7th, 2025
	October 27th, 2024	February 22nd, 2025	June 14th, 2025
	November 2nd, 2024	March 1st, 2025	June 21st, 2025
		March 22nd, 2025	June 28th, 2025
		March 29th, 2025	July 12th, 2025*
		April 5th, 2025	July 19th, 2025*
		April 12th, 2025	July 26th, 2025*
		May 17th, 2025	
		May 31st, 2025	
Hours Tested	15	43	28

Planned Pool Test Dates are denoted by "*".

II. LOCOMOTION TESTING AND VALIDATION

Proper testing of the control board is critical to mission success. However, this is a complex task requiring significant amounts of testing time during development. Due to limited in-water testing time and to ensure sufficient testing time for mission code, communications, and mechanical systems, a significant portion of control board testing occurred in simulation.

The simulator was developed using an open source 3D game engine, Godot. The selection of this tool for simulation was solely motivated by prior familiarity with this game engine. Godot includes a 3D rendering and physics engine, allowing simulator development time to focus on vehicle modeling and control board math validation.

The simulator not only models SeaWolf VIII, but simulates a control board as well. This allows testing and validation of mathematical methods in a high-level language where math libraries are already provided (by the game engine). Additionally, it allows mission code unit tests to run under simulation without access to any control board hardware or sensors.



Fig. 1: Control Board Simulator

However, a simulated control board is only capable of validating the approach to the problem, not the actual device. The largest risk with simulation testing is that the real control board's firmware has an implementation error. Even if the math is correct, it can be implemented improperly or other firmware bugs can prevent proper operation of the device. To address this, the simulator was expanded to allow use of a physical control board to control the simulated vehicle. In this operating mode, the simulator provides simulated sensor data to the control board, and receives motor speeds from the control board. Thus, the control board firmware itself can be tested and debugged under simulation.

III. ML TRAINING AND RESULTS

Training and validation of ML algorithms is vital to their performance in detection and identification of targets. This requires an efficient and reliable environment for us to train, test, and validate our algorithms. These needs require a simulator with capabilities to support simulation of robotic systems and a powerful game engine to generate photorealistic images and environments to increase the model accuracy. As such, we developed our simulator using Unity because it provides sufficient support for simulating robotic systems in virtual environments and is a game engine that can produce hyper-realistic simulations.



Fig. 2: Simulation of slalom poles in pool environment (a) and detection of slalom poles in simulated pool environment (b)

The simulation consists of images ranging from blue to blue-green for simulating various water and visibility conditions. The generation of the environment also randomizes the position and rotation of the camera and pool which challenges the algorithm to become accustomed to undesirable conditions. The training/testing/validating split of datasets used is 80-10-10.

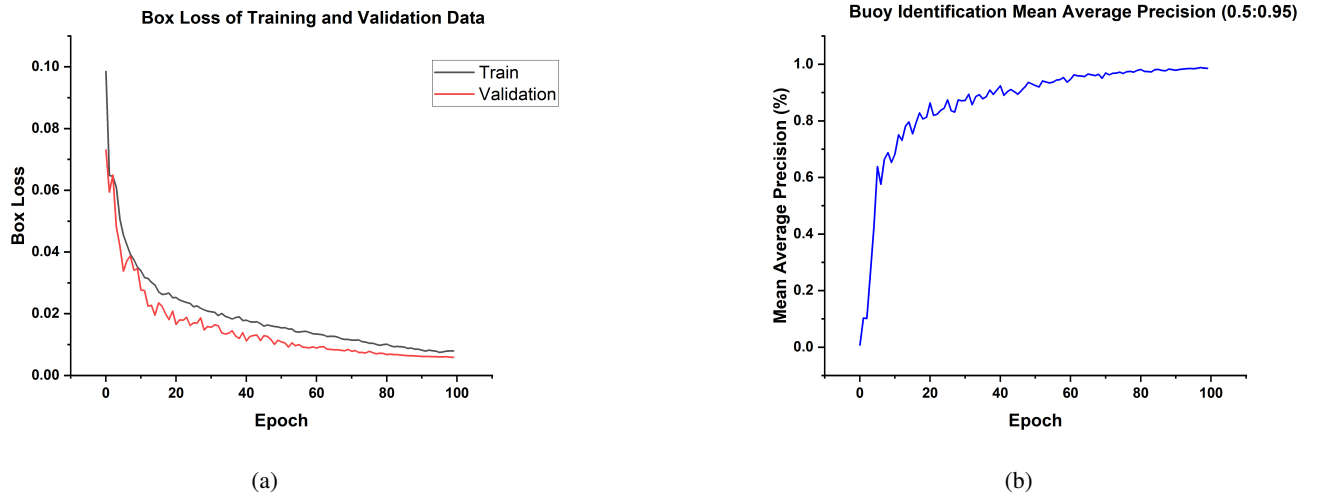


Fig. 3: Box Loss of ML algorithm in training and validation data over an epoch of 100 (a) and mean average prediction of Buoys over epoch of 100 (b)

A considerable risk in developing ML models is the datasets generated oftentimes is biased to the simulated environments which results in negative performance of the algorithm in real-world conditions. The drop in performance can be associated with a simulations inability to simulate all conditions possible in the real-world which affects the neural networks ability process its input in real-world environments. To mitigate these issues, the simulator can be easily adjusted using a color slider to fit additional water conditions previously not possible. Additional assets can be added to the environment to increase variety of datasets, resulting in a more robust model. Finally, one of the best methods to mitigate a model's bias to simulated environments is incorporating real data in its training, testing, and validating datasets to improve the model's accuracy.

IV. TORPEDO PROJECTILE TEST RESULTS

Primary testing of torpedo projectile shape was through in-field testing in air and water. The test procedure included firing the assembled torpedo system and video-capture of the range of the torpedo projectile as measured by a 36 in ruler. A visual of a torpedo projectile test in water can be observed in Fig. 4.

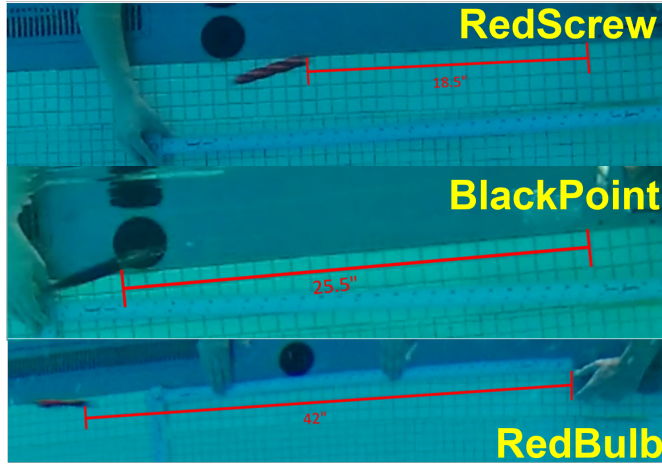


Fig. 4: Torpedo Projectile Testing

Table IV below summarizes torpedo projectile test results. Forward distance was defined as the distance the back end of the torpedo traveled away from the end of the launcher before sufficient momentum loss resulting in vertical deviation occurred. Side-to-side deviation is listed as ‘minimal’ if no qualitative deviation was observed.

TABLE IV: Torpedo Projectile Data

Projectile	Avg. Forward Distance	Max Forward Distance	Side Deviation (first 30cm)	Side Deviation (Total)	Pass/Fail
BlackPoint	23"	25.5"	Minimal	1-2"	Pass
RedScrew	17.5"	18.5"	Minimal	Minimal	Pass
RedBulb	36"	42"	Minimal	Minimal	Pass

All torpedo models met the desired minimum 12" (30cm) straight-line travel requirement with the RedBulb outperforming the other designs.