Roboboat 2025: SEALs Technical Design Report

Bryan Chan, Caleb Lewis, Liam Bray, Alec Jensen, Alejandra Jimenez, Tammy Lin, Bennett Rodriguez, Christina Tran, Jerry Minh Tram

I. ABSTRACT

Upon completing the Roboboat 2024 competition, we present our new approach to combat the tribulations of the past. Our team's main focus was designing our vessel, *Bruce,* for hydrodynamic efficiency, stability, and buoyancy. We prioritized hull design and motor placement to solve cavitation issues seen in the years prior. In programming, a new approach was taken to solve navigation issues by incorporating an in-house graphical simulation and replacing waypoint-based navigation with low-level reactive control for increased versatility. These improvements helped provide the MHS SEALs with more force and control over the competition.

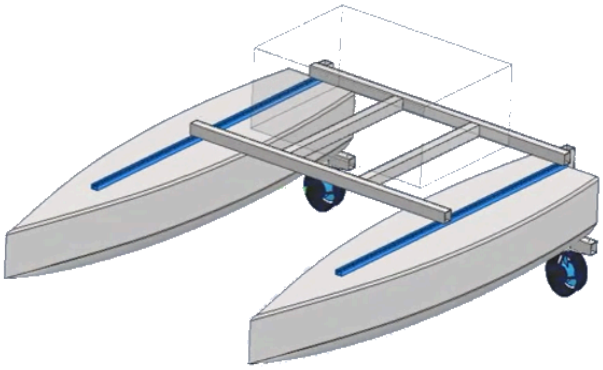

Figure I - Render of "Bruce"

II. COMPETITION STRATEGY

2.1.0 Strategy Overview

Autonomous navigation in *Bruce* is handled by an event-driven finite state machine called *Boat Control* with two primary functions: **search** and **run**. The **search** function uses the estimated poses of key features, such as buoys found within *Bruce*'s frame of reference, and compares them to known course configurations to identify tasks. The **run** function starts upon task recognition and uses a combination of reactive algorithms and waypoint planning. Tasks that are found are given a *TaskStatus* state, corresponding to either incomplete, finished, or failed.

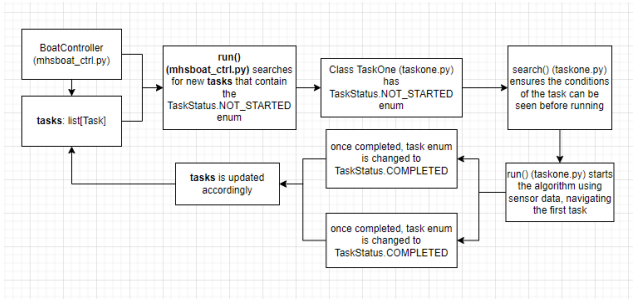

Figure II - Boat Control State Handler System

The functionality of *Bruce* heavily relies on the localization of surrounding features. To achieve the highest possible accuracy, sensor data from the *Intel DepthSense Camera D435i* and *Velodyne VLP-16 LiDAR* are fused to take advantage of computer vision feature detection and precise positional measurements. Using live 3D point cloud detection of the LiDAR, the DBSCAN algorithm works in conjunction to cluster groups of points that could indicate a course object. These detections are then projected onto the 2D plane of the camera and compared to detections from a YOLOv11 computer vision model to confirm a course object has been detected and assign the cluster an object type.

The main method of navigation for many of the challenges is assigning a waypoint to detected pairs of buoys, or to other features that LiDAR and camera can reliably detect. Depending on the orientation of *Bruce* to the midpoint, *Bruce* will activate either left or right thrusters using our custom Python-MavROS thruster controller at a constant speed, adjusting every second to weave through the buoys with ample reaction time.

2.1.1 Navigation Channel

To complete the **Navigation Channel** task, *Boat Control* is activated, and will immediately proceed to mapping the closest buoys. The depth-sense camera is being used to identify the buoys detected with the DBSCAN algorithm from LiDAR data. When two green buoys and two red buoys are identified with the correct estimated distance and 90-degree angles with their relative shape, *Boat Control* will proceed to place waypoints between the midpoint of each pair of buoys and activate *Bruce's* Python-MavROS thruster controller to navigate between both pairs of buoys.

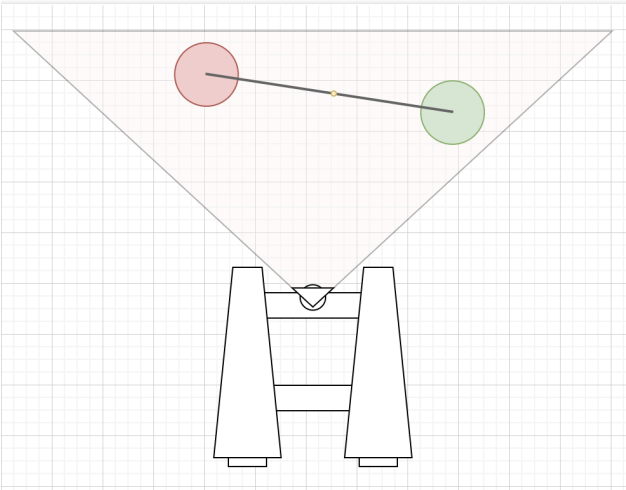

Figure III - Navigation Channel Design

2.1.2 Mapping Migration Patterns

The **Mapping Migration Patterns** task follows a similar structure to the previous task. *Bruce* identifies pairs of red and green buoys and uses the buoy furthest from the yellow buoy to calculate the midpoint between them. This calculation yields the center of the widest gap for *Bruce* to travel through for efficient, safe travel. However, we recognized that the ball buoys may become out-of-sight due to their shorter height. Thus, we choose to use lidar-odometry library *KISS_ICP* to reliably backtrack to *Bruce*'s position when the ball buoy was last seen. This ensures that *Bruce*'s sensors will continue to function properly for challenge-specific algorithms.

2.1.3 Treacherous Waters

Docking is achieved with exclusion zones placed at the perimeter of the docking area, permitting *Bruce* to center itself between these zones with LiDAR data tracking relative position. Once the correct banner has been identified, the vessel will move into position.

2.1.4 Racing Against Pollution

Navigating between the gate buoys in **Racing Against Pollution** uses the same primary mechanism as the **Navigation Channel**; however, additional complexity is introduced by identifying the starting queue and avoiding obstacles while navigating around the blue buoy.

Initially, the position of the detected red light panel is compared to any newly detected green light panel objects in the same location, indicating a state change. Once a change from red to green from our YOLOv11 computer vision model is detected, *Bruce* drives through the second pair of gates and reactively avoids all objects until it detects the blue buoy. Upon detection, *Bruce* will attempt to keep the buoy at the same position relative to *Bruce* on the left side while moving forward. Once *Bruce* has fully turned around and detects the gates again, it will continue to avoid obstacles until it can pass through the gates again.

2.1.5 Rescue Deliveries

After completing the **Racing Against Pollution** challenge, *Bruce* will attempt to survey the course by setting waypoints in open space until a black triangle shape or a black plus shape is identified, and it will continue to move to *Bruce* until the symbol takes up a majority of the screen.

2.1.6 Return to Home

When **returning home**, the vessel will reposition itself until it identifies two black buoys using our computer vision model and LiDAR data, and calculate the midpoint to navigate to using MavROS.

III. DESIGN STRATEGY

3.0.0 Design Overview

This year's design builds upon the successes and lessons learned from last year's hull design. Our key design objectives included refining the shape of the hull, optimizing material usage, and ensuring efficient placement of the motors.

3.1.0 Hull and Frame Design

The basis for the hull's design was improving upon last year's vessel, *The Meg,* to reduce the time needed to design and construct our hulls. Therefore, we took the basic frame of *The Meg* and added more curvature to the bottom of the hull by adjusting the shape of the ribs in order to achieve greater stability and hydrodynamics. The team concluded other manufacturing methods would require time or equipment we lacked access to. Therefore, we took a cue from traditional marine vessels and constructed a wooden frame. We wrapped a fleece cloth across the wooden frame forming the desired ensemble that was later covered with several layers of fiberglass sealed with polyester resin. To reduce the risk of air bubble formation through a more refined shape, we built the first two layers from huge sheets of fiberglass to provide the strength, with the final layer containing smaller sheets overlapping each other like scales. Overall, we used the same method for constructing the hulls as in previous years as we had become familiar with the process and were able to produce favorable results.

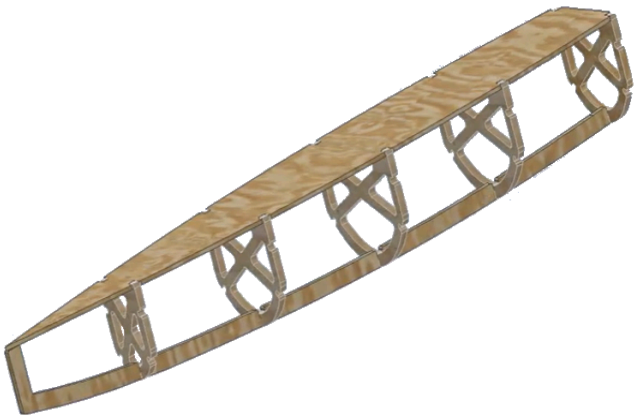

Figure V - Bruce's Hull Construction

This year, we have decided to forego the polycarbonate platform that our electrical components rested on in previous years and have opted to mount the container with them inside directly to the aluminum supports that attach to *Bruce*. We chose to do this in order to make the design sleek and easier to manipulate in an attempt to achieve a more stable vessel. The blue rails are attached directly along the middle of the hulls to provide rigidity and allow the team to adjust the location of the electrical components with ease to ensure a stable center of gravity. The other supports provide a resting place for the container housing the electrical systems.

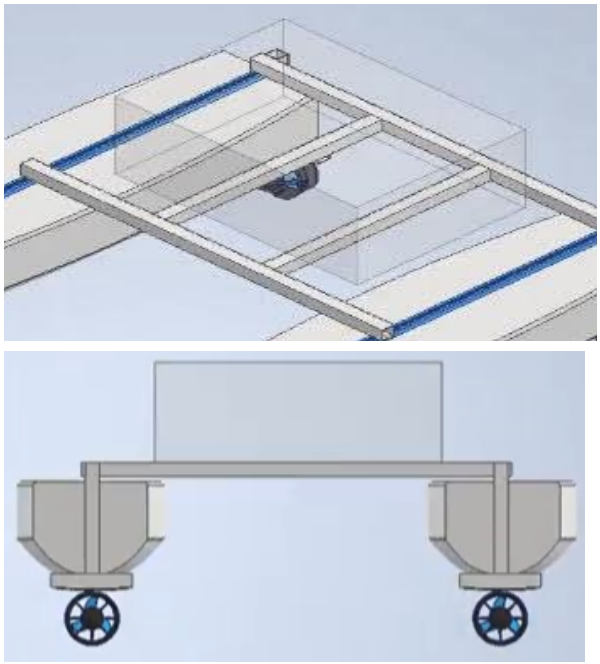

Figure IV - Aluminum Support Frame and Motor Mount

3.1.1 Propulsion System

Similar to previous years, we have used *Blue Robotics'* T200 thrusters as the main source of propulsion. In order to prevent cavitation from occurring as in previous years, we have redesigned our motor mounts and placed them in a different location on the hull. Our new design incorporates the experience we have acquired from past years to provide a solution to our problems.

3.1.2 Water Gun

Similar to previous years, our water gun system consists of an impeller pump driven by a 12V motor and 0.170 in. I.D. x 1/4 in. O.D. clear vinyl tubing in order to create a continuous output using the lake water. The water pump is mounted to the aluminum frame while custom 3D-printed mounts are used to position the tubing to the desired height. One half of the vinyl tubing is positioned underwater to provide the necessary ammunition. A custom nozzle was designed to increase the firing distance and accuracy of our pump. With the aforementioned design and components, *Bruce* is able to auto-adjust its position to stay on target for **Rescue Deliveries**.

3.1.3 Racquetball Delivery System

This year, to complete the racquetball delivery section of the **Rescue Deliveries** task, our team designed a mechanism to drop the balls into the object delivery boats. It consists of a 2⅜ inch pipe which is able to be adjusted to a desired angle. It stores 3 racquetballs by using 3 linear actuators with a ½ inch stroke to stop the racquetballs from dropping before we intend them to. When *Bruce* has positioned itself alongside the delivery boat, the linear actuators are retracted in succession to allow one ball to drop at a time before extending again to store the rest.

3.2.0 Electrical System

The design of the electrical system of last year worked well, but there were several improvements that could be made to improve the organization, performance, and repairability of the design. Our team this year focused on these 3 things to be able to create a superior design to decrease friction for the other teams.

3.2.1 Organization

During Roboboat 2024, our team had to open the electrical system several times to access various components of *Bruce*. While our split-box design allowed for highly efficient use of space, the accessibility of components, and therefore repairability of the entire design, was highly limited, leading to friction for the other departments. Our new electrical design was carefully thought through to ensure easier troubleshooting access for the other departments.

3.2.2 Performance

Last year, our team ran into several performance issues with the electrical system. First, our circuitry for the battery was not designed to be able to handle the thruster current, so the thruster power was heavily limited. Second, the thrusters produced cavitation in the water due to improper placement, but also due to incorrect tuning of our IMU, the Cube Orange. Third, our circuitry had many unnecessary parts that could be eliminated for better organization, such as voltage regulators. To address these challenges, we designed a new schematic diagram to plan our electrical system. The new design has two separate batteries for our thrusters, ensuring they can deliver their maximum power. Additionally, it consolidates the many separate voltage regulators into one high-power regulator.

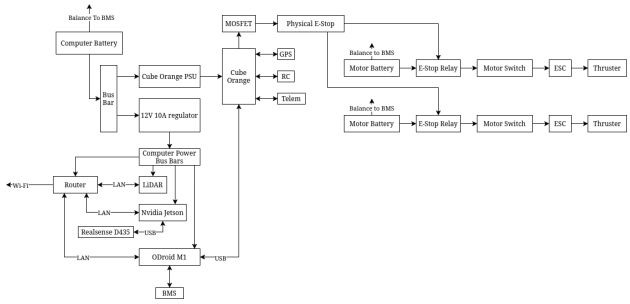

Figure VI - Electrical Block Diagram

3.2.3 Repairability

One major issue with our previous designs was the lack of repairability. When any component failed, it was difficult to open up the electrical system and repair or replace a component. This year, we have improved this by doing two things. First, we have organized our electrical system as mentioned previously. This organization allows us to be able to more easily diagnose problems and repair them quickly.

Second, we have used more repairable connections throughout our system. Rather than relying on relying on jerry rigged connectors as we have in previous years, we have learned how to use the proper tools for crimping and soldering custom JST, DuPont, and XT90 connectors. As a result, the modularity of our design has increased, allowing for the aforementioned organization improvements, as well as an increase in repairability.

3.3.0 Software Infrastructure

Our software infrastructure comprises two primary systems: the computer vision system and the motor control system. These systems communicate through the ROS2 framework using Python. We have prioritized a simple and adaptive autonomous system over a more robust software such as Nav2 to give us more low-level control of *Bruce*'s movement.

3.3.1 Computer Vision

Our new computer vision system allows *Bruce* to identify and determine the position of objects within its view. *Roboflow* was utilized to create a detailed model, and allowed us to upload images of buoys from various angles and environments. These images were meticulously annotated in *Roboflow* by marking bounding boxes around specific target objects, such as different types of buoys. The annotated dataset was then employed to train the YOLOv11s computer vision model. Trained on a dataset of over 25,000 images, the model achieves around 90% precision in object detection with confidence levels nearing 60%. Machine learning tasks are handled by a *Jetson Xavier NX*, which delivers robust performance in image processing and effectively manages the model's computational requirements.

3.3.2 Motor Control

Motor control is handled fairly simply. When a task is being run, the task code manually controls the forwards, backwards, and angular velocities of *Bruce*. These values are then passed through MavROS to the Cube Orange. Due to the modularity of the code this year, the thruster controls can either be linked to the actual boat or purely simulated for testing purposes.

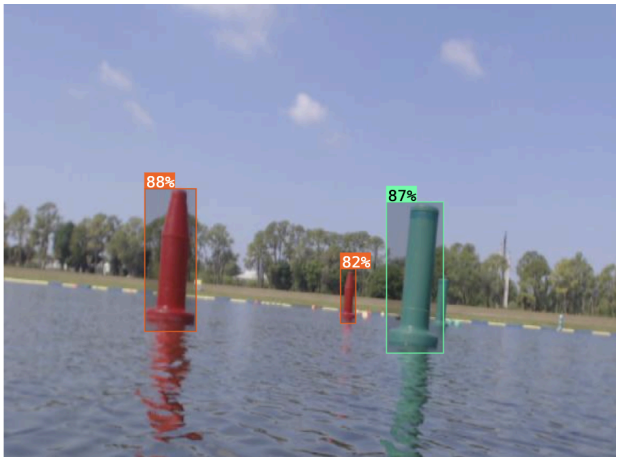

Figure VII - YOLOv11 model with high confidence

IV. TESTING STRATEGY

4.1.0 Testing Overview

Testing of *Bruce* was executed in three parts: initial navigation logic testing using simulated buoy positions and boat movement generated by a GUI, ROS sensor testing, and physical testing for all components to account for environmental conditions and construction imperfections. As an improvement from last year, we have successfully documented all of the necessary steps to run these tests for future SEALs members.

4.1.1 PyGame Simulator Interface

To best reflect real-world conditions, a GUI made with PyGame simulates buoy positions relative to *Bruce* in the same format outputted by real-world sensor outputs. The GUI is subscribed to the /mavros/cmd_vel topic, with pre-simulated sensors and sample data to test prospective algorithms. Using the buoy positions, the same navigation and kinematic calculations can be used to output desired linear and angular velocities.

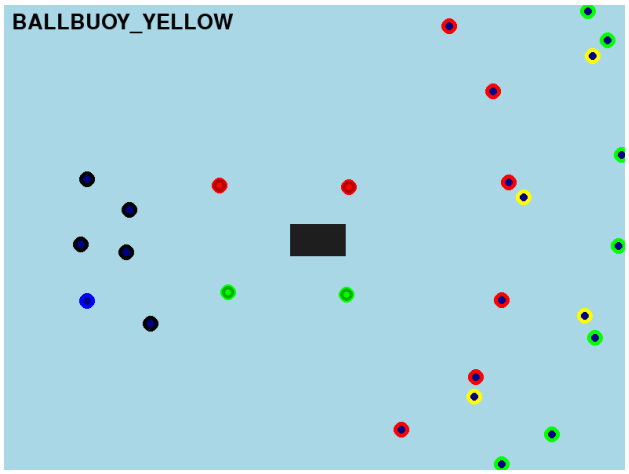

Figure VIII - GUI simulation of Task 2 and 4 for testing navigation logic

4.1.2 Sensor Visualization

We used rVIZ to visualize live LiDAR and depth-sense camera data to visualize ROS2 topics of interest. This ensured that the DBSCAN algorithm was clustering the point-clouds correctly at /velodyne_points, and the depth-sense camera was also publishing to the /camera topic at an expected frame rate.

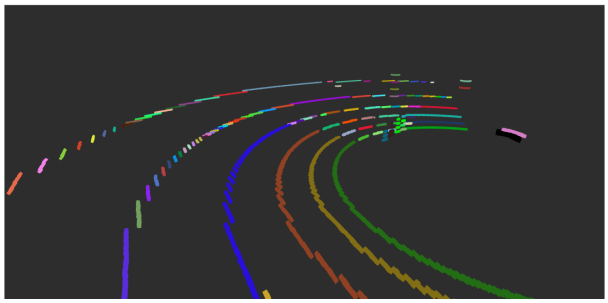

Figure IX - Clustered LiDAR data from real environment

4.1.3 In-Water Testing

After *Bruce*'s completion, testing began at a private pool to simulate the sensors, involving the identification and mapping of buoys. Much physical and software testing, configuring, calibrating, and adjustment of the camera and LiDAR were done during this phase. The testing of the accuracy after complete calibration was assessed inside of the pool. To best reflect competition standards, buoys were placed in a similar format to the competition.

Figure X - Testing new calibration

4.1.4 Knowledge Continuity

The computer science members from last year who designed *The Meg*'s software graduated from our high school with a rushed and messy code base. As a result, the code left behind was difficult to interpret without detailed comments in the code or guides on how to run ROS2 nodes critical for development. To practice knowledge continuity for future members, we have implemented Python type safety and detailed comments for every function, a new GitHub project board for organization, and dedicated members to continue documenting new discoveries and breakthroughs along the way.

V. ACKNOWLEDGEMENTS

We would also like to thank our mentors, **Joshua Ogg** and **Jerry Minh Tram**, who have lent their expertise in their respective fields. We would also like to extend a thank you to the institutions who have given support throughout the construction of *Bruce*: **Blue Robotics** for their exclusive discount on their thrusters and the **Robotic Vision Lab** at the **University of Texas at Arlington** for allowing us to use their labs for research. And finally, we express our deepest gratitude to the teachers and sponsors who have helped us along the way, **James Hovey, Jason Forsythe, and Laura Ebanks**. They have sacrificed their valuable time to work with us and offer guidance.

References

[1] “Writing a simple publisher and subscriber (Python) — ROS 2 Documentation: Humble documentation.” https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html (accessed Aug. 24, 2024).

[2] PRBonn, “GitHub - PRBonn/kiss-icp: A LiDAR odometry pipeline that just works,” *GitHub*, Sep. 30, 2022. https://github.com/PRBonn/kiss-icp (accessed Oct. 02, 2024).

[3] Ros-Drivers, “GitHub - ros-drivers/velodyne: ROS support for Velodyne 3D LIDARs,” *GitHub*, Oct. 31, 2024. https://github.com/ros-drivers/velodyne (accessed Nov. 04, 2024).

[4] Mavlink, “GitHub - mavlink/mavros: MAVLink to ROS gateway with proxy for Ground Control Station,” *GitHub*, Oct. 10, 2024. https://github.com/mavlink/mavros (accessed Dec. 03, 2024).

[5] IntelRealSense, “GitHub - IntelRealSense/realsense-ros: ROS Wrapper for Intel(R) RealSense(TM) Cameras,” *GitHub*, Dec. 03, 2024. https://github.com/IntelRealSense/realsense-ros (accessed Dec. 10, 2024).

Appendix A: Component List

Component	Vendor	Model/Type	Specs	Custom/ Purchased	Cost	Year of Purchase
ASV Hull Form/Platform	Home Depot/Own Design	Plastic Container	link	Purchased	9.98	2022
Propulsion	BlueRobotics	T200	link	Purchased	200.00	2022
Power System	HobbyKing	Lithium-polymer battery	link	Purchased	108.97	2022
Motor Controls	BlueRobotics	Basic ESC	link	Purchased	36.00	2022
CPU	Amazon	Jetson Xavier NX	link	Purchased	700.00	2023
CPU	Amazon	Odroid M1	link	Purchased	135.00	2023
Teleoperation	Amazon	Taranis Q X7	link	Purchased	137.99	2022
Compass	CubePilot	Here3	link	Purchased	290.00	2023
Inertial Measurement Unit (IMU)	CubePilot	Here3	link	Purchased	290.00	2023
Camera(s)	Intel	Intel RealSense Depth D435	link	Purchased	314.00	2022
Doppler Velocity Logger (DVL)	N/A	N/A	N/A	N/A	N/A	N/A
Hydrophones	N/A	N/A	N/A	N/A	N/A	N/A
Algorithms	N/A	N/A	N/A	N/A	N/A	N/A
Vision	Intel	Intel RealSense Depth D435	link	Purchased	314.00	2022
Localization and Mapping	CubePilot	Cube Orange	link	Purchased	485.00	2022
PyRealsense	Intel	N/A	link	N/A	N/A	2023
MavSDK	DroneCode	N/A	link	N/A	N/A	2023

Gazebo Simulator	Open Robotics	N/A	link	N/A	N/A	2023
ROS	Open Robotics	N/A	link	N/A	N/A	2022
Docker	Docker	N/A	link	N/A	N/A	2022
MicroPython	Raspberry	N/A	link	N/A	N/A	2023
QGroundControl	DroneCode	N/A	link	N/A	N/A	2023
Mission Planner	Ardupilot	N/A	link	N/A	N/A	2022