RoboBoat 2025: Technical Design Report

Jack Bolte, Janelle Cai, Ved Ganesh, Noah Haefner
Amy Shi, Teagan Sullivan, Toya Takahashi
Massachusetts Institute of Technology, Cambridge, MA, USA

***Abstract*—Arcturus is debuting a new autonomous surface vehicle (ASV), *Fish 'N Ships*, for RoboBoat 2025. After a season away to build a more maneuverable, modular, and well-integrated platform, we plan to attempt every task at competition. While our team has prepared for every task, we prioritized the navigational challenges to ensure a vehicle with reliable basic functionalities before expanding its capabilities. Our design strategy is focused on adaptability and modularity to create a smooth initial integration process with room for continuous development. Following through on our commitment to system integration and testing, Fish 'N Ships has been thoroughly tested in various environments to ensure consistent performance.**

I. Competition Strategy

This season, our primary objective is to create a vehicle capable of handling both navigational and mechanical challenges. With that, our goal is to attempt every task at competition; however, we prioritized building a reliable base platform first and then building up capabilities to maintain a realistic workload. Guided by this strategic vision, we concentrated on navigation tasks (Tasks 1, 2, and 6) to refine and enhance our existing software stack. Meanwhile, docking and mechanical tasks (Tasks 3, 4, and 5) received less emphasis, as we prioritized achieving consistent performance before introducing additional complexity to the system.

A. Course Approach

Precise robot pose estimation is essential for navigation. We achieve this by combining GPS data from the ZED-F9P dual-antenna Real-Time Kinematic (RTK) system, which provides centimeter-level accuracy through error correction with a stationary onshore base station and IMU data using an Extended Kalman Filter (EKF) [1]. This fusion ensures robust and reliable pose estimation for consistent task execution. For obstacle avoidance in path planning, we generate an occupancy grid using 3D point clouds from the HDL-32E Velodyne LiDAR. Although the ZED 2i Stereo Camera offers high-resolution depth images, we opted for LiDAR due to its 360° horizontal field of view, consistent $2\,\mathrm{cm}$ accuracy, and robustness to variations in weather and lighting conditions.

B. Buoy following tasks: Task 1 and Task 2

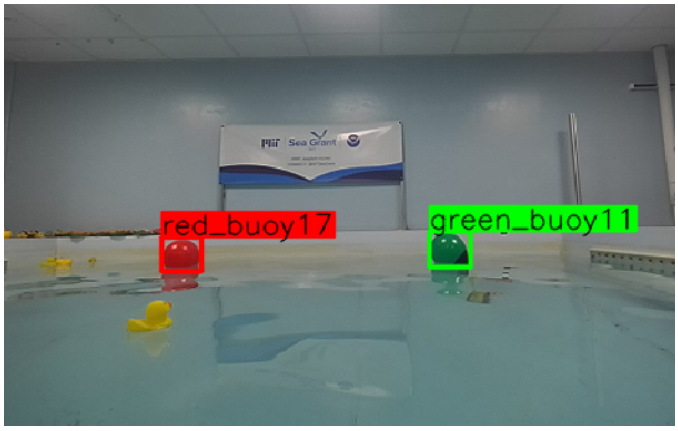


Fig. 1: A pair of red and green buoys detected by our fine-tuned YOLOv8 model.

To achieve buoy following, we identify the locations of sequential red and green buoy gates. Buoy colors and types are detected using a fine-tuned YOLOv8 model, chosen for its real-time speed and accuracy [2]. The positions of the buoys are extracted from LiDAR data using Euclidean clustering, and their locations are projected onto the camera feed using a calibrated extrinsic transformation and the camera's intrinsic matrix. Finally, buoys detected by the camera and LiDAR are matched by pairing those that maximize the total Intersection over Union (IoU).

Once we localize the buoys, we iteratively grow a sequence of buoy gates by repeatedly adding the buoy pair closest to the end of the current sequence. The boat then follows a series of calculated GPS

waypoints between the buoy gates, navigating along a discretized, obstacle-free path calculated by our A* path planner, which uses the 2D occupancy grid produced by our mapping system.

C. Task 3: Docking

To execute the docking task, we designed a state machine to navigate the two-sided dock, ensuring that the vehicle docks in the correctly labeled area. The vehicle can be in one of six states: `approaching`, `checking_camera`, `shifting`, `orbiting1`, `orbiting2`, and `docking`. During the `approaching` state, the dock location is determined by matching a pre-existing point cloud template with the LiDAR data using point cloud registration. If the current dock is occupied by a vessel or does not match the correct color or shape (determined using our YOLOv8 model), the vehicle transitions to either the `shifting` or `orbiting` states. In the `shifting` state, the boat moves to check the next dock on the same side, while in the `orbiting` state, it switches to the other side of the dock. Once an appropriate dock is located, the vehicle enters the `docking` state, where it navigates to the center of the dock using our path planner.

D. Task 4: Speed Challenge

The Speed Challenge is a lower-priority task for us, but the point bonus for completing all tasks makes it more significant. Given the point values of the Speed Challenge components and the slightly constrained time frame, we have decided to focus solely on the stationkeeping subtask. Once a set of four rectangular buoys is located after completing the buoy-following tasks, the vehicle uses its PID controller to maintain station at the holding bay until the LED turns green. To detect the color change, we utilize color segmentation to detect when a rectangular patch of pixels changes color, which offers the advantage of not requiring training data, unlike learning-based approaches.

E. Task 5: Object and Water Delivery

The Object and Water Delivery task is completed in between other tasks while target vessels are within proximity. If faraway vessels are identified during other tasks, their locations are recorded. The vehicle will then navigate to them at the end of its current task, provided the total path length does not exceed a specified threshold. A separate webcam mounted on the turret is used to detect the vessel's shape, with the centroid of the shape serving as the target for turret aiming using a PID controller.

II. Design Strategy

After our last competition season, we recognized the need to place a greater emphasis on system integration and testing. To ensure seamless operation across all subsystems, we specified a list of design requirements for our vehicle, prioritizing a lightweight and easily transportable design to make testing easier. Adapting our previous boat to meet these requirements proved infeasible. As a result, we decided to build a new vehicle, focusing on maneuverability, modularity, and testability to better meet our goals.

A. Hulls

Catamarans are optimal in the context of the RoboBoat competition over monohulls for their high stability and low drag characteristics [3]. The hulls are symmetric for better maneuverability during the docking and navigation tasks. To increase metacentric height and therefore stability, the batteries were placed within the hulls and protected by plastic containers [4].

The hulls went through two major iterations. The first pair of hulls weighed $36\,\mathrm{lbs}$, exceeding design specifications by 105%. To reduce weight, we shortened the hulls from $1.8\,\mathrm{m}$ to $1.2\,\mathrm{m}$, reduced fiberglass layers, and employed vacuum bagging technology, decreasing total weight to $12\,\mathrm{lbs}$.

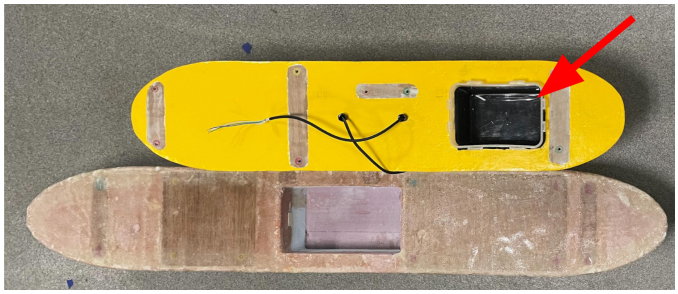


Fig. 2: Top to bottom: new hulls and old hulls. Note the cavity in each hull for holding battery boxes.

B. Propulsion

To optimize maneuverability, four T200 thrusters are mounted in a vectored configuration, making the boat holonomic. However, since this is the first year using this system, experimentally fine tuning the angle of the thrusters was necessary to find the best forward and side to side speeds. Because of this, a piece of detachable marine plywood is sandwiched between the hulls and thrusters, which allow the thrusters to be mounted in 22.5° increments. To protect the protruding thrusters from damage, thruster cages were designed to be both lightweight and durable (see Appendix B).

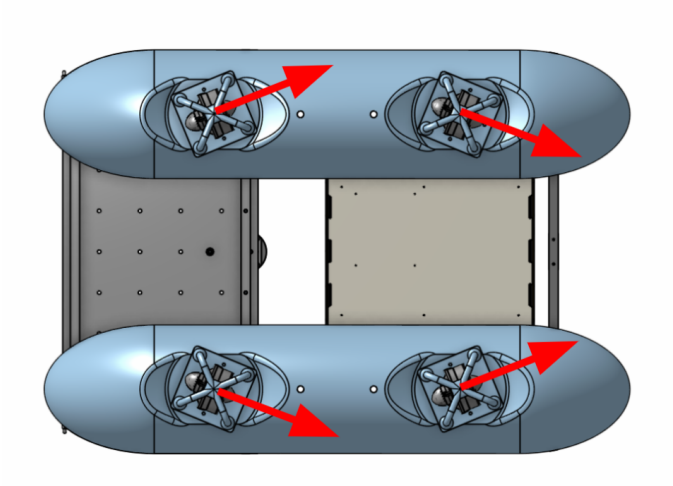


Fig. 3: CAD of thruster vectored configuration. Each T200 thruster can be manually rotated and mounted to the hulls at 22.5° increments. Red arrows indicate direction of thrust on the water.

C. EE box and Sensor Mast

The ASV's electronics enclosure (EE box) is designed to be (1) lightweight, (2) debuggable, (3) waterproof, (4) breathable (to prevent overheating), and (5) compact. While off-the-shelf options from Pelicase, Blue Robotics, Condition 1, and Tupperware were considered, all candidates failed to meet at least one of the five design requirements. Therefore, the team designed a custom EE box using primarily $^1/_8$" plywood and a Polyurethane Laminate (see Appendix C). These choices resulted in an 86% increase in container volume/weight over the leading off-the-shelf option (from Condition 1) and better breathability. The EE box features a base height to lid height ratio of 1:3 and elevated pegboards, allowing easy access to electronics from both above and below. Removable faceplates enable easy addition of new passthroughs as the ASV's electronics payload evolves. Inset faceplate gaskets, an H-gasket between the base and the lid, and sealed seams waterproof the EE box to a tested depth of several cm–well beyond design specifications. Finally, a center pole was added to prevent water pooling on the lid and double as a self-locating pin for easy lid to base alignment.

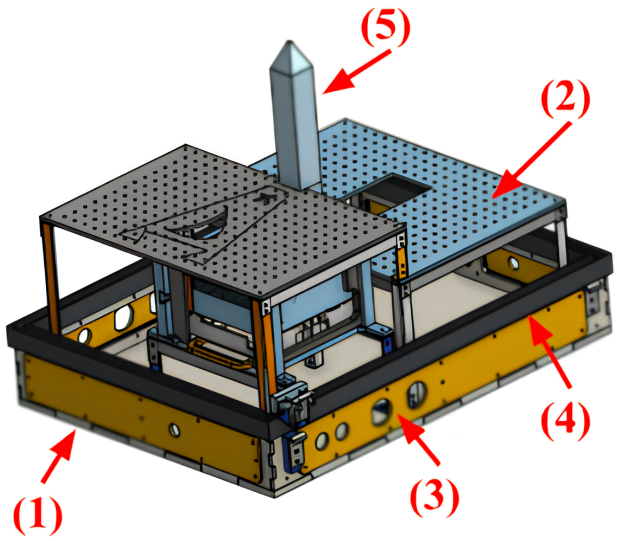


Fig. 4: EE box CAD. (1) short base (2) elevated peg boards (3) removable faceplates (4) H gasket and (5) center pole. Lid not pictured.

The LiDAR and ZED are mounted on an aluminum sensor mast to increase range of vision. Vibration-damping mounts, adjustable levels, and multiple points of contact securing the sensor mast to the deck and aluminum beams ensure that the sensors are level and experience minimal vibrations.

D. Mechanisms (Ball Launcher, Turret, Water Cannon)

The mechanism for delivering balls and water to the vessels around the course consists of three subsystems: the turret, the ball launcher, and the water cannon. By integrating these subsystems into a single assembly, we create a reliable method for aiming at targets.

1) Turret: The objective this year was to create a lightweight, precise turret that could withstand both the weight of the ball launcher and water

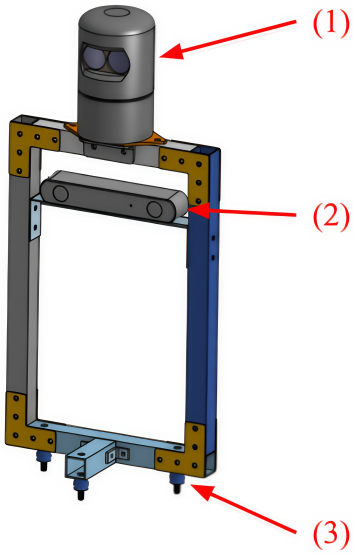


Fig. 5: Sensor Mast CAD. (1) LiDAR (2) ZED (3) height adjustable vibration-damping mounts.

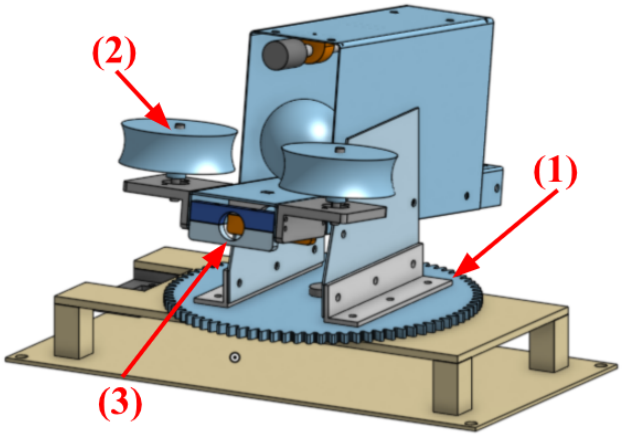


Fig. 6: Full Mechanism assembly. (1) Gear turret system (2) Ball launcher (3) Water cannon

cannon. Initially, a belt-driven turn-table riding on ball bearings was used for rotation. However, testing showed that the bearings would get stuck, so the bearings and belt were removed, and acrylic gears were used to drive the rotation.

2) Ball Launcher: The ball launcher is designed to achieve a velocity high enough to minimize the effects of parabolic motion, simplifying the aiming process (see Appendix D). The launcher consists of two inverted curved edge wheels rotating in opposite directions, powered by a pair of $2000\,\mathrm{rpm}$ motors. The wheels slightly compress the balls to maximize contact and launch them in a straight path without spin. A lead screw mechanism feeds the balls into the wheels.

3) Water Cannon: The water delivery system features a pump that directs water through tubes that run underneath the ball launcher. This ensures that the water is delivered at the same angle and incline as the launcher. The pump delivers a flow rate of $271.2\,\mathrm{cm}^3/\mathrm{s}$ with a $1.27\,\mathrm{cm}$ outlet diameter (see Appendix E). Assuming constant volume-flow rate, we designed a resin-printed, replaceable nozzle with a reduced outlet diameter of $0.635\,\mathrm{cm}$. This nozzle allows the water stream to travel up to $7.36\,\mathrm{m}$ without manually altering the turret's angle.

E. Electrical System

In previous years, our electrical system proved functional but was bulky, unreliable, and difficult to fix. To mitigate these issues, we designed custom, modular printed circuit boards (PCBs) that implement the majority of the boat's electrical functionality.

1) System Overview: Our system consists of four primary boards. The Central Hub distributes power and signals to all peripheral boards, and it provides the USB interface to the onboard computer. The Buck Board converts power from the 6S battery with $19\,\mathrm{V}$, $12\,\mathrm{V}$, $5\,\mathrm{V}$, and two adjustable buck converters. It provides overvoltage, short-circuit, and thermal protections. The Battery Management System (BMS) protects the batteries from damage. Each of the three batteries on the boat, two 4S batteries for the thrusters and one 6S battery to power the rest of the boat's systems, have their own BMS. The E-Stop communicates the emergency stop information, as well as commands to manually drive the boat. See Appendix F for the electrical system diagram.

2) Battery Management System (BMS): The BMS monitors battery health and performs battery shutoff. It protects against undervoltage, overcurrent, and cell imbalance. We chose to use a $200\,\mathrm{A}$ NFET transistor on the BMS to implement battery shutoff and E-Stop, preventing the need for heavy and power-hungry contactors. The transistor gate is pulled down, so the system will fail safe in the case of control system failure. The board was tested up to $70\,\mathrm{A}$ of load current, which is well in excess of our expected maximum operating current of $40\,\mathrm{A}$.

3) E-Stop Implementation: The previous, WiFi based E-Stop experienced problems with reliability

and range. Instead, we chose to use LoRa on the 915 MHz ISM band to communicate E-Stop and manual boat control for its long range and high noise immunity. E-Stop can be triggered by three conditions: 1) the E-Stop button on the boat is pressed, 2) the E-Stop button on the shore-side transmitter is pressed, or 3) the E-Stop loses connection with the shore-side transmitter for over one second. When any of these conditions occur, a signal is sent to both BMS boards on the 4S batteries, which then cut off the output power.

F. Software System Architecture

This competition cycle, the autonomy team prioritized modularity and reproducibility in our code, enabling rapid development and iteration. To achieve this, we organized our code base, `all_seaing_vehicle`, into several Robot Operating System (ROS) 2 packages, covering navigation, perception, controls, and task-specific functionality. This modular approach allows us to continually reuse and refine core features such as navigation and perception across competition cycles, while easily swapping out task-specific code. Additionally, we decided to transition away from another middleware, Mission Oriented Operating Suite (MOOS-IvP), due to two main factors: 1) the steep learning curve of its C++ implementation and features, and 2) the added complexity of the bridge between MOOS and ROS, which unnecessarily complicated the overall system architecture.

In addition to utilizing ROS nodes for continuous communication between processes through a publisher-subscriber pattern, we implemented a hierarchy of ROS actions to manage task execution and vehicle commands (Fig. 7). This request-response model allows for continuous feedback and the ability to cancel or abort tasks as needed. The action servers are organized in three layers, with each layer sending requests to deeper layers:

- Task manager: A ROS node responsible for deciding which task the vehicle is currently performing based on the current state.
- Layer 1 – Task execution: Action servers responsible for responding to high-level requests to execute RoboBoat tasks such as Follow the Path and Docking.
- Layer 2 – High-level commands: Commands requiring additional calculations before requesting Layer 3 commands. Examples include navigating to a target point while avoiding obstacles and completing the process of aiming at a target and shooting.
- Layer 3 – Core commands: Low-level commands for fundamental control of the boat such as waypoint following, station keeping, moving the turret, and firing water/racquetballs.

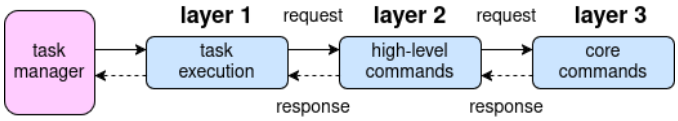


Fig. 7: all_seaing_vehicle's ROS 2 action server hierarchy with the task manager node responsible for sending requests to layer 1.

Underneath Layer 3, we have a network of publishers and subscribers for handling state estimation, map generation, object detection, and hardware interfacing. See Appendix G for more details.

III. Testing Strategy

From past experience, we knew that it was important to not have software testing bottle-necked by hardware development, and that frequent testing was crucial to developing a reliable system. Thus, we devised a plan to test our overall system both in simulation and physically on our sister test boat Minerva, which has a similar sensor and propulsion system to Fish 'N Ships, while the new vessel was under construction.

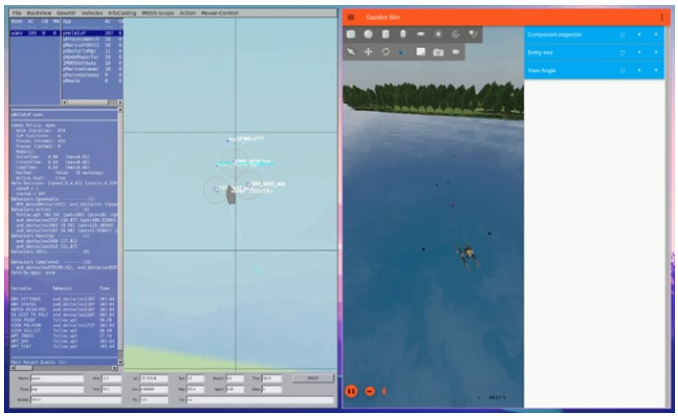


Fig. 8: Gazebo simulation testing.

To facilitate frequent testing of our autonomy stack, we rely on the Virtual RobotX (VRX) Gazebo simulator provided by the Open Source Robotics Foundation. In addition to the default

VRX worlds, we developed custom Simulation Description Format (SDF) files and Python scripts to model RoboBoat tasks such as Navigation Channel, Follow the Path, and Docking. While the simulation provides an idealized environment, it enables rapid prototyping and code validation without interfering with the mechanical team's hardware design process.

Fig. 9: Physical testing on the Charles River.

We conducted various physical tests to gain a more accurate representation of the competition environment and evaluate components such as sensors and communication systems that often perform differently than in simulation. To ensure consistent progress, we established a timeline to test the boat physically at least once every two weeks. This schedule provided time to iterate on unreliable systems while keeping us accountable to deadlines. For each test, we outlined specific objectives and goals, documenting the results to better plan for future milestones. We began with small-scale tests focused on isolated components (see Appendix B and C) and gradually progressed to validating the entire system through the completion of full tasks (see Appendix A).

We conducted small-scale tests at the MIT Sea Grant test tank to verify sensor and thruster functionality. Additionally, we performed system integration tests to ensure the mechanical, electrical, and software systems were working together as intended. These focused tests allowed us to validate design changes efficiently, avoiding the need for full-scale tests, which are more time-consuming and require additional planning.

Large-scale indoor physical tests were conducted at the MIT Z-Center swimming pool. Due to the lack of a reliable GPS signal indoors, we used Nav2's Adaptive Monte Carlo Localization algorithm [5], [6] on a pre-built map using Cartographer [7] for accurate localization of the vehicle. Although GPS systems could not be tested in this environment, we focused on isolating and testing tasks such as Follow the Path, Docking, and the Speed Challenge. These indoor tests proved especially valuable during the harsh winter months when outdoor testing was not feasible.

Finally, full-scale tests were conducted at the Charles River, where we extensively tested GPS, WiFi, and LoRa systems to ensure accurate robot localization and reliable communication between the shoreside and the boat. Beyond isolated tests, we evaluated the autonomy stack's ability to complete a sequence of tasks, simulating the competition environment. Outdoor testing also allowed us to collect training and testing data under various weather conditions, further improving the robustness of our system.

IV. Acknowledgments

Arcturus would like to thank all of our sponsors for making this incredible project possible for our team through their support in terms of design reviews/guidance and funding: Saronic, MIT Mechanical Engineering Department, MIT Edgerton Center, Cadence, Robosys, The COOP, Yamaha, Cambridge Science Fund, and MIT OME.

We would also like to acknowledge those who donated materials/tools or lent us testing facilities: Formlabs, PTC, MIT Sea Grant, MIT Architecture and Design (MAD) Lab, N51 Edgerton Center Team Shop, Edgerton 6C Student Shop, and 4-409 Edgerton Student Project Laboratory.

Finally, We would like to make a special acknowledgment to our mentors whose guidance and support have been instrumental to our growth. We are grateful for Dr. Andrew Bennett's technical and administrative guidance throughout the years. Another mentor instrumental to our growth is Audrey Chen, who has provided helpful design feedback and general team guidance.

References

[1] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," International Conference on Intelligent Autonomous Systems (IAS), 2013.

[2] G. Jocher, A. Chaurasia, J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.
[3] P. Couser, "An Investigation Into the Performance of High-Speed Catamarans in Calm Water and Waves," Mar. 1996.
[4] J. Mégel and J. Kliava, "Metacenter and ship stability," American Journal of Physics, vol. 78, no. 7, pp. 738–747, Jul. 2010, doi: 10.1119/1.3285975.
[5] S. Macenski, F. Martin, R. White, J. Clavero, "The Marathon 2: A Navigation System," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.
[6] S. Macenski, T. Moore, DV Lu, A. Merzlyakov, M. Ferguson, "From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2" Robotics and Autonomous Systems, 2023.
[7] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-Time Loop Closure in 2D LiDAR SLAM," Robotics and Automation (ICRA), 2016.

APPENDIX A
TEST PLAN AND RESULTS

Our team had physical, simulation, and bench tests to allow independent development of subsystems. To do so, the workload is broken down into three main sub-teams: mechanical, electrical, and autonomy. From there, we have even smaller project teams working on designing, prototyping, and testing subsystems of Fish 'N Ships. Every week, we have system integration meetings where we check in on each sub-team's progress and make adjustments to our testing plan as needed.

Subsystem: every subsystem has a design timeline as follows: 1-2 weeks for design/research, 1 week for design review and material lead time, 1-2 weeks for prototyping, 1 week for integration, and reiteration as needed. Please see detailed examples of our subsystem testing in Appendices B and C.

Full-system: after independent subsystems are tested, we integrate them onto the main vessel and deploy the vessel to ensure that the components are behaving as expected. Our full system tests follow the Testing section of the Gantt chart timeline in Appendix H, and more details are included in the Simulation Tests and Physical Tests subsections below.

A. Simulation Tests

Simulation testings were done through the VRX Gazebo simulator as mentioned in the Testing Strategy section of the paper. For these tests, we needed a Linux based computer to run the simulations. Thankfully, we were able to obtain a dual booted Toughbook laptop borrowed from the MIT Sea Grant lab in order for all the autonomy members to run the simulation. There are far fewer safety factors/risks to account for since we run these simulations in the laboratory space. The Autonomy section of the Gantt chart (see Appendix H) describes the simulation testing schedule, where each system is developed over the course of two to three months. The objective of these tests were to validate that the code is working as intended, and the simulations were run continuously throughout the development process to verify that the subsystems were working.

B. Physical Tests

Physical tests allowed us to verify our design, ensure progress, and integrate independent components. Since our team built a new vehicle and redesigned all of the hardware, most of the base platform was developed independently. Once the base platform was integrated, we had biweekly full system tests that followed the schedule of the Testing section of the Gantt chart in Appendix H. To simulate the competition environment, we tested outdoors at the MIT Boathouse, where a dock was available by the Charles River for easy deployment of Fish 'N Ships. However, other groups on campus also utilized this space and it was only available on the weekdays, so we often had to work around conflicting schedules. To replicate the competition environment for practice, we built task props such as the dock and the delivery vessels. We also reused Polyform buoys from previous years for object detection testing. There were significantly more risks when it came to physical testing because we needed to make sure there were enough helpers for carrying the boat and that we were following safety measures in working with the river. To ensure the safety of our members, we coordinated tests with the boathouse manager to ensure the weather was appropriate and that safety protocols at the docks were followed. When the weather posed a severe risk for hypothermia and frostbite, we moved our tests indoors to the pools at the MIT Z-center to ensure the safety of our members. Below, we will describe the various stages of our full system testing and the corresponding objectives and results.

Early Stage — Completion of Fish 'N Ships

Objectives: We started testing our new vessel as soon as it was completed. During this stage, we

mainly focused on achieving teleoperation functionality with the new vehicle and adjusting the vessel to ensure a desired center of mass and thruster positions.

Results: We began testing our new vessel immediately after its completion, focusing initially on achieving teleoperation functionality and adjusting the vehicle to optimize the center of mass and thruster positions.

During this phase, we encountered several challenges:

- The customized electronics box was difficult to access, complicating debugging efforts.
- The sensor mount was unstable, hindering the effective use of the camera and LiDAR.
- Water accumulated in the cavities between the battery box and the hulls.

To remedy these issues, we iterated on the design of the EE box and sensor mount, and we waterproofed the cavities to prevent water pooling in future tests.

Mid-stage — Fish 'N Ships with New Features

Objectives: After the initial round of tests, Fish 'N Ships had new features integrated in terms of the electronics and the updated mechanical parts. Our objective for these tests was to validate that the new electronics function as expected, the updated mechanical components met our requirements, and the GPS and waypoint following portions of autonomy worked.

Results: We were able to successfully set up the new GPS system after a few tries, and most of the electronics behaved as expected. The newly designed electronics box was a lot more accessible. However, like most of our tests, unexpected issues arose:

- One of the tests happened on a windy day, and our vehicle was a lot more soaked than usual. We realized that we needed to better waterproof our connectors into the EE box.
- The sensor mount was still not as stable as we expected.

Late-stage — Competition Preparation

Objectives: Our goal was to simulate the competition course to practice the tasks, test the perception system of our autonomy stack, and integrate the latest iteration of the sensor mount.

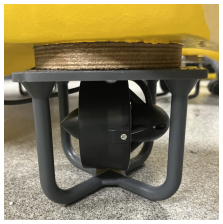
Results: We were able to record ROS bags for the autonomy team to test the perception system on. Some unexpected problems were:

- With more nodes running for our software functions, our onboard computer unexpectedly crashed.
- Our batteries drained a lot faster than expected even though the current the electronics system was drawing seemed to be reasonable.

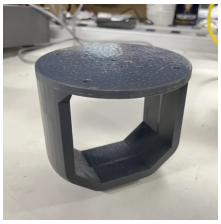
Appendix B
Thruster Cages

The initial cages weighed $1.9\,\text{lbs}$ per cage and together contributed 10% of the ASV's overall weight. To reduce weight, two alternative designs were proposed; the first design weighs $0.6\,\text{lbs}$ per cage and the second weighs $1.04\,\text{lbs}$ per cage, both were printed in Formlabs Tough 2K resin. Onshape FEA showed the first design deflected by approximately $0.0001\,\text{in}$, under loads equivalent to the boat being dropped from $1\,\text{m}$. Physical tests also supported this, as each cage could support over $150\,\text{lbs}$. Given these results, the first design is optimal considering its proven functionality and lower weight. The redesign decreased cage weight by 68.4%; the cages now contribute only 4% of overall vessel weight.

Initial Design (1.9 lb each)



Design 1 (0.6 lb each)



Design 2 (0.9 lb each)

Fig. 10: Different iterations of the Thruster Cages.

Appendix C
EE Box Lid Design

To calculate the maximum load on the walls of the EE box lid, we calculated drag force on the box

in $5\,\mathrm{m/s}$ gusts:

$$F_d = \frac{1}{2} \cdot c_d \cdot A \cdot \rho \cdot v_{\text{rel}}^2$$

where

$F_d =$ drag force,
$c_d =$ drag coefficient ≈ 1.05,
$A = \perp$ Area of Lid $= 0.14\,\mathrm{m}^2$ along long face,
$\rho =$ density of air at STP $= 1.2\,\mathrm{kg/m}^3$,
$v_{\text{rel}} =$ relative velocity of flow $\approx 7\,\mathrm{m/s}$.

$$\begin{aligned} F_d &= \frac{1}{2} \cdot 1.05 \cdot 0.14\,\mathrm{m}^2 \cdot 1.2\,\frac{\mathrm{kg}}{\mathrm{m}^3} \cdot 49\,\frac{\mathrm{m}^2}{\mathrm{s}^2} \\ &= 4.3\,\mathrm{N} \end{aligned}$$

The cross members make analytical methods difficult so we used Onshape's FEA tool:

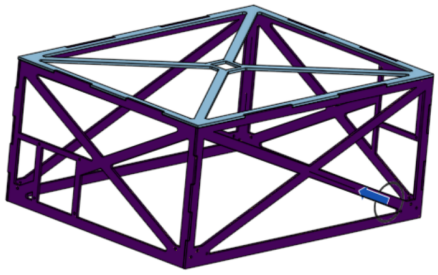


Fig. 11: FEA of EE box lid under $5\,\mathrm{m/s}$ gusts.

With a safety factor >3.5, the lid is well within the specification.

Appendix D
Ball Launcher Calculations

To simplify aiming, the launcher was designed to hit the $0.584\,\mathrm{m}$ tall target from a maximum distance of $3\,\mathrm{m}$ without adjusting launch angle. By measuring the time to travel $1.82\,\mathrm{m}$ over 5 trials, we found that the muzzle velocity of a launched ball is $7.15\,\mathrm{m/s}$. Various launch angles were simulated to determine how many targets would be hit within a set distance.

The optimal angle is $28.6°$. This angle exceeds the $3\,\mathrm{m}$ requirement and peaks at the top of the target,

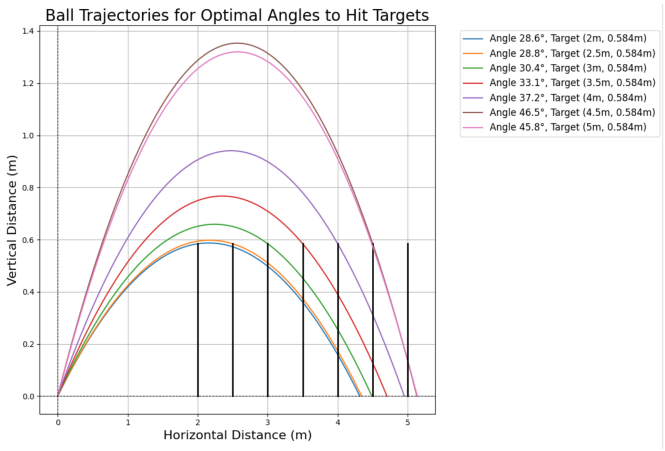


Fig. 12: Different ball trajectories depending on angle. Air resistance neglected. Black lines represent target height.

ensuring that any launch between 0 and $3\,\mathrm{m}$ will not overshoot the $0.584\,\mathrm{m}$ tall target. As long as the boat shoots within this range and in the correct x-y direction, it will hit the target.

Appendix E
Water Delivery Calculations

Our pump maintains a volumetric flow rate of $273.4 \cdot 10^{-6}\,\mathrm{m}^3/\mathrm{s}$. To determine the appropriate nozzle size to reach $s_{\text{min}} = 3\,\mathrm{m}$, we utilized the formula for projectile motion:

$$s = \frac{v^2 \sin(2\theta)}{g},$$

$s =$ horizontal range,
$v =$ fluid flow $\left(\frac{4Q}{\pi d^2}\right)$,
$g =$ gravitational acceleration $(9.81\,\mathrm{m/s}^2)$,
$\theta =$ initial angle (45° for maximum range).

Substituting $v = \left(\frac{4Q}{\pi d^2}\right)$ into the range equation gives:

$$s = \frac{16Q^2}{\pi^2 d^4 g}.$$

Using this formula and the given flow rate, and assuming that the pressure gradient along the tubing is negligible, we have found that a nozzle outlet diameter of $d = 1/4$" $(6.4 \cdot 10^{-3}\,\mathrm{m})$ will reach $d = 7.36\,\mathrm{m}$.

APPENDIX F
ELECTRICAL SYSTEM DIAGRAM

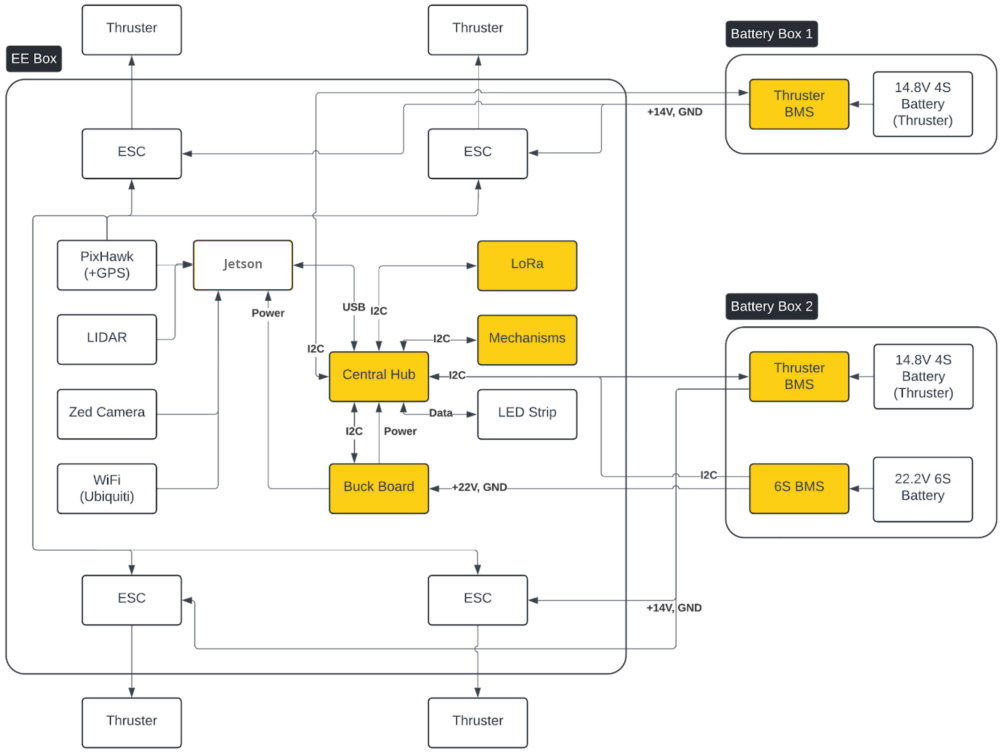


Fig. 13: Electrical system overview. Custom boards are indicated in yellow.

APPENDIX G
SOFTWARE SYSTEM ARCHITECTURE

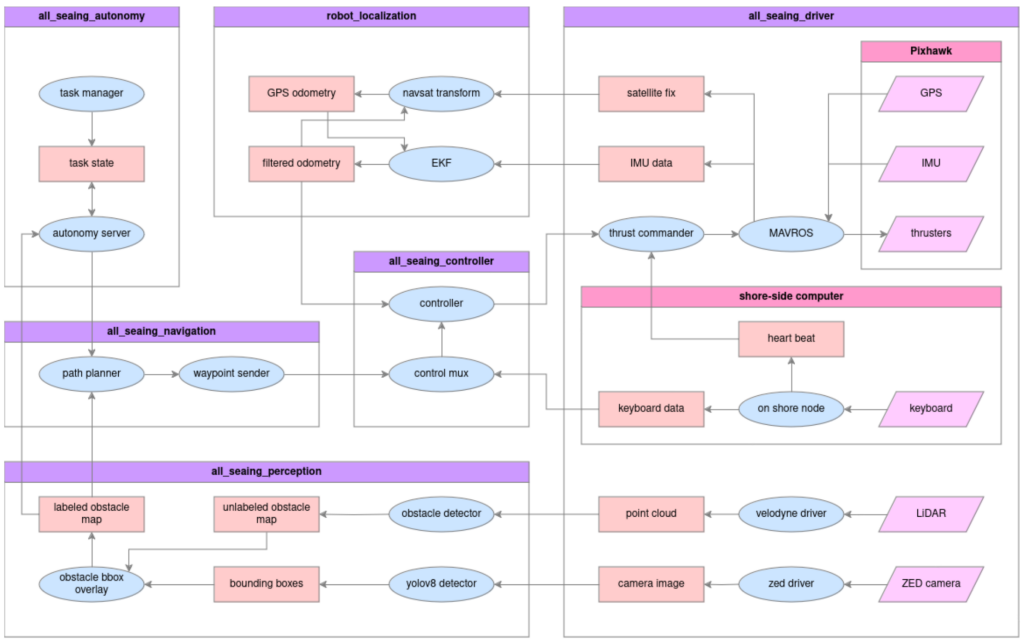


Fig. 14: Software System Architecture.

Appendix H
Arcturus Team Gantt Chart

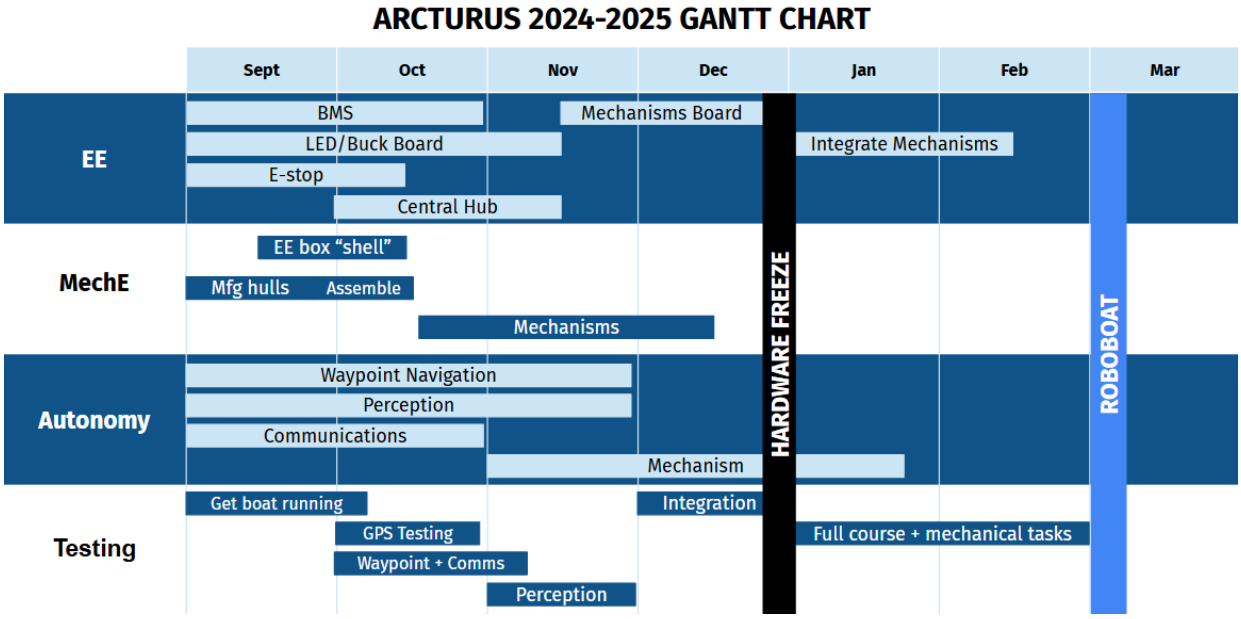


Fig. 15: The team Gantt chart consists of subsystem projects, organized by the respective sub team. The long bars that span horizontally indicate the relative timeframe for the design, prototype, and test of the subsystem.