RoboBoat 2026: Technical Design Report

Jack Bolte, Janelle Cai, Ved Ganesh, Brendon Jiang, Panagiotis Liampas,
McKinley Pestano-Young, Teagan Sullivan
Massachusetts Institute of Technology, Cambridge, MA, USA

***Abstract*—This season, Arcturus focused on improving the reliability of our vehicle, Fish ‘N Ships by addressing key issues in the autonomy and electrical system. On the software side, Arcturus incorporated simultaneous localization and mapping (SLAM), implemented fallback approaches for tasks, and redesigned the controller for more reliable obstacle avoidance. On the electrical side, Arcturus incorporated redundancy in the emergency stop system and switched to the CAN bus communication protocol to improve communication between the shore and boat. To ensure the success of these architecture changes, Arcturus ran over 240 hours of full system tests. To allow frequent testing, Arcturus adopted a phased mechanical design strategy that focused on developing a new vehicle for RoboBoat 2027 while making necessary but non-invasive updates to Fish ’N Ships. These decisions enabled Arcturus to complete its most extensive physical testing to date.**

I. Competition Strategy

Ultimately, our team’s goal this season is to achieve core capabilities for all tasks and advanced capabilities for half, with an emphasis on doing so reliably. We first focused on ensuring that major foundational changes to the system, including implementing SLAM and updating the electrical system, were reliable through extensive on-water testing. Because so many tasks are based on navigating around buoys, indicators, obstacles, and banners, this set us up to tackle basic capabilities for almost all tasks. Then, we worked on reliable implementations of advanced capabilities for Entry & Exit Gates, Debris Clearance, and Emergency Response Sprint because these tasks placed a strong emphasis on navigation. Our next priorities were Supply Drop and Navigate the Marina, with Harbor Alert as our lowest priority.

A. Course Approach

In order to identify the various objects in the course, we are using a three-camera system consisting of a forward-facing ZED 2i Stereo Camera and two diagonally-backward-facing OAK-1 Lite W cameras. An HDL-32E Velodyne LiDAR gives us information about the exact location of the identified objects with respect to the robot.

However, data from the cameras and LiDAR alone is insufficient, since we want to reliably track objects throughout a course runthrough, even if not continuously detected. To that end, we are using the ZED-F9P Real-Time Kinematic (RTK) system, with an onshore base station, and the Pixhawk 6x as an Inertial Measurement Unit (IMU), to get an estimate of the robot’s position and velocity, and the course objects’ locations, as described in Section II-D.

Using the robot’s state and the map, we implemented a stack of navigation algorithms to move the robot between arbitrary points within the course while avoiding obstacles, as described in Section II-C. This stack is the basis of the approaches that we adopt to successfully and robustly complete all of the competition tasks (see Fig. 2) and is described below.

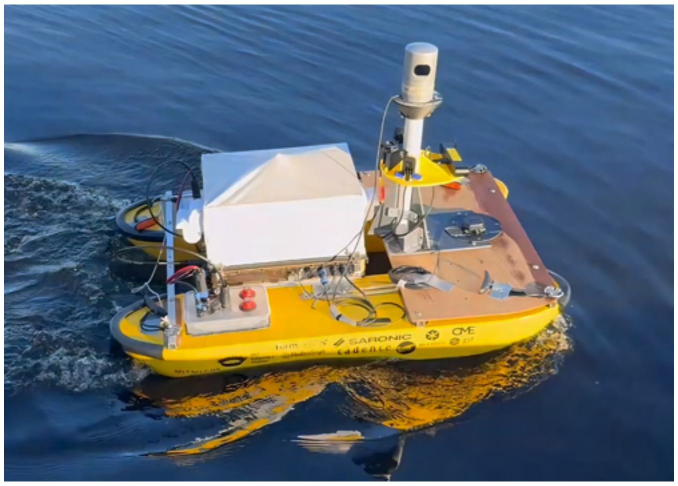
Fig. 1. Fish ’N Ships with changes made for RoboBoat 2026.

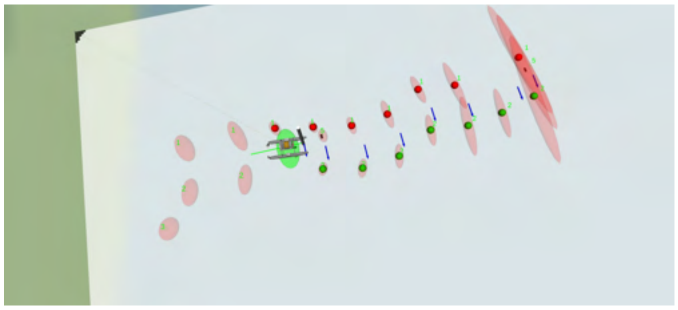
Fig. 2. Our Simultaneous Localization And Mapping (SLAM) and navigation stack used to identify and navigate through the gates of Tasks 1 & 2.

B. Tasks 1 & 2: Evacuation Route & Return and Debris Clearance

Both tasks involve identifying a sequence of buoy pairs based on their color and navigating between them using the high-level controller described in Section II-E. We track the sequence of buoy pairs and continuously update next-pair assignments as the global map is updated, based on perpendicularity to the path and distance to the previous pair. We account for false detections and drift by tracking the pairs’ positions according to the global map and discarding them if necessary.

For Task 2, searching for and circling around any green indicators is done using the same logic as in Task 3, which we describe below. Retracing the path is done by tracking the last pair in the original sequence and recomputing the rest of the path from there. The overall behavior for this task is implemented as a state machine with states for finding the first gate, following the path, searching for the green indicator, navigating around it, and returning through the path, with transitions that account for failure at each state.

C. Task 3: Emergency Response Sprint

Task 3 involves passing through red/green buoy gates and circling a yellow buoy. While the task incentivizes circling the yellow buoy as fast as possible, entering and exiting the buoy gates require more precision than possible at top speed. Thus, we employ a multi-stage approach to this task: `start`, `probing`, `circling`, and `return`.

The `start` and `return` stages involve passing through a buoy pair, implemented the same way as in Tasks 1 & 2. Through testing, we found that the perception system cannot immediately identify the yellow buoy from a distance, so the vehicle probes for the yellow buoy by moving in a direction perpendicular to the buoy gate. When the vehicle detects the yellow buoy, it enters the `circling` state. This state uses a dedicated PID control loop that outputs velocity commands to keep the distance between the vehicle and the yellow buoy constant, enabling a continuous, circular trajectory while maintaining near maximal speed. Finally, the vehicle returns to the starting buoy gate by a recorded waypoint.

D. Task 4: Supply Drop

This task requires detecting the target banners and shooting water and balls at them using an appropriate mechanism. For this, we use two independent controllers: one for navigation and station-keeping in front of each target, and one for aiming and shooting.

We identify the target as the closest banner, navigate to its general location using the high-level navigation controller, and finally station keep in front of the banner using a PID controller similar to the one used in Task 5.

To aim the turret, we implemented a separate PID controller that rotates the shooting mechanism, using a dedicated camera to center the detected target in the image frame.

E. Task 5: Navigate the Marina

For the core capabilities of the task, we pick the closest unoccupied slot, using the positions of the dock and boat banners in the global map. For navigation towards the slot, we are using the same logic as Task 4, with high-level navigation up until some distance and PID control for alignment with the dock and entering it. We also incorporate the dynamic obstacle avoidance capabilities, described in Appendix E to avoid collisions with the dock.

The advanced and disruptive capabilities of the task only affect the selection of the docking slot, with the added perception challenge of identifying the banner numbers and indicator colors. We implement the task behavior as a state machine, with states for searching for a valid docking slot, high-level navigation, PID, canceling navigation in the case of a misdetection, and picking a new slot to dock.

F. Task 6: Harbor Alert

For this task, we must identify the alert with a microphone and navigate to a yellow buoy near a task. To locate the buoy without ever seeing it, we approximate its general position in an estimate of the course layout. The estimate is based on the first task's marker buoys, which are easily identifiable and can be used to estimate the course's orientation. If we detect a yellow buoy while navigating to the estimated waypoint, we replace the initial guess with the buoy's exact position.

The key difficulty of this task is interrupting the execution of a task and resuming this task without significant overhead. To preserve and "freeze" task execution state, we do not perform any perception-related updates to its variables, and we store the robot's position from when the alert was called. We navigate back to the recorded position before unfreezing the task.

G. Fallback Approaches

Motivated by the localization problems we had during last year's competition, we decided to implement localization-independent fallback approaches for each task, only using the local map & PID controllers. These approaches are similar in principle to the primary approaches, but are designed to avoid tracking global information such as buoy positions over time. We detail the task-specific fallback approaches in Appendix F.

II. Design Strategy

Last season taught us that successfully integrating new hardware updates requires full-system testing and debugging, often at the cost of time-consuming on-water autonomy testing. However, last season also showed that high-level architecture changes to Fish 'N Ships, particularly the electrical system and hull form, could significantly improve reliability, hydrodynamic performance, and weight.

As a result, we adopted a phased approach. For RoboBoat 2026, we decided to continue to compete with our 2025 vehicle Fish 'N Ships. We focused on improvements to the vehicle that would not prevent us from testing our autonomy system. Though a significant architecture change, the electrical system's switch from I2C to CAN was judged both low-risk and necessary, as detailed later in Section II-B. The I2C to CAN switch was completed over the summer to avoid interrupting testing during the regular season. In parallel, we began designing a new vehicle for RoboBoat 2027 with the aim of addressing key shortcoming of the current hulls. The details of those shortcomings as well as our progress so far in designing, simulating and testing components of the 2027 vehicle are described in Appendix G.

A. Mechanical Systems

The team's focus on extensive navigation testing drove this year's mechanical design strategy. Crucially, Fish 'N

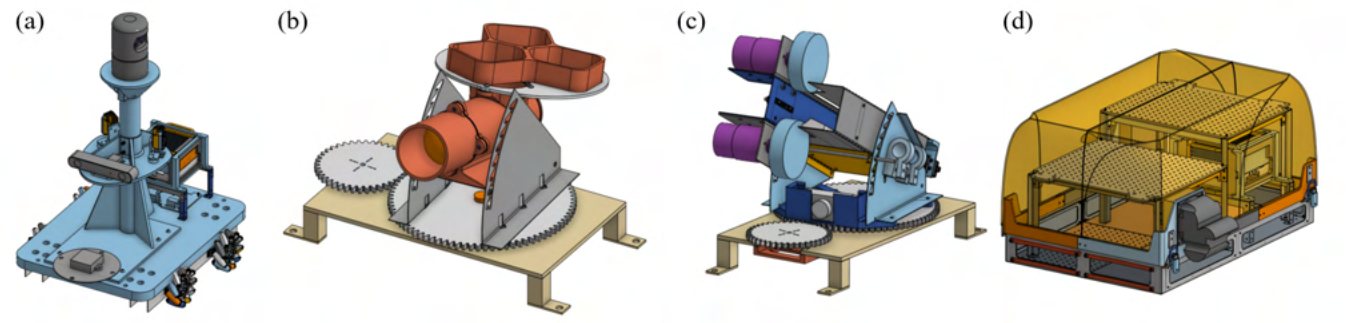

Fig. 3. (a) Three-camera sensor mast on its test cart; (b) pneumatic ball launcher; (c) vertical flywheel ball launcher; (d) electronics enclosure. (b)-(c), the two ball launchers are interchangeable; the vertical flywheel launcher is intended as a low-risk alternative to the pneumatic launcher.

Ships experienced zero mechanical failures at RoboBoat 2025, indicating we had a reliable platform for testing autonomy on the water. We therefore split mechanical projects into small, necessary updates to Fish 'N Ship's existing architecture motivated by changes in the autonomy and electrical systems, shown in Fig. 3, and large, exploratory projects for our 2027 vehicle.

1) Three Camera Sensor Mast: The team's switch to SLAM motivated motivated adding two additional cameras to the sensor mast. Supporting the larger camera payload required a complete redesign of the sensor mast, which allowed us to introduce an additional interfacing subsystem: a test cart. Modeled after the rover developed by [1], the test cart enables rapid, out-of-water testing and in-lab validation.

To stabilize the top-heavy mast, we added fins and a separate base plate. The base plate integrated the boat's perception and navigation sensors—three cameras, LiDAR, GPS, and external compass—into a single stack that can be shuffled between the test cart and boat without changing relative sensor positions. Maintaining consistent transforms eliminated the need for recalibration and ensured test-results were representative. The mast, base, and fins were milled for accurate sensor placement. We constructed the mast from PVC, instead of metals, to preserve the GPS's unobstructed view of the sky.

2) Mechanisms: Ball Launcher + Water Delivery: Last year's ball launcher used a fixed-angle, counter-rotating flywheel architecture to achieve a 3m range with minimal footprint. This year, improvements to the perception stack extended reliable detection to 5m. To match this capability, we designed a fixed-angle pneumatic launcher with a higher muzzle velocity. Because the team had limited pneumatics experience, we also developed a fixed-angle vertical flywheel launcher as a low-risk fallback. This design achieved the 5m range by shaping the trajectory through controlled backspin.

We retained the 2025 water delivery system after testing confirmed a reliable 7.5m range, exceeding requirements. Like last season, we integrated the water nozzle and ball launcher into a single turret. Unlike last year, we decoupled the launch angles of the two systems to account for differences in muzzle velocity and projectile dynamics.

3) Electronics Box (EE Box): The electrical system's transition from I2C to CAN bus increased board count, requiring a larger EE Box. In light of their positive reception, we retained salient features of the original—namely its 1:3 base-height-to-lid-height ratio, elevated pegboards, and removable faceplates—while incorporating user feedback from the 2025 season.

The base of the new box was laser cut from an aluminum sheet folded and brazed to remove the joining, waterproofing, and caulking required by the previous plywood design. This reduced box weight by 50% and provides an integrated antenna ground plane. To prevent shorts, the floor is lined with thin PLA rafts with captive nuts—creating internal mounting points without piercing the enclosure. Pre-bent aluminum beams clamp the removable faceplates in place of many screws, enabling rapid reconfiguration of the box's connector layout while reducing screw count by 75%. A zipper was added to the fabric lid to improve on-water access, sidestepping the fiddly process of realigning the original over-center latches on the water. Finally, the pegboards were hinged to allow full access to the bottom layer. An annotated model of the box is presented in Appendix H.

B. Electrical System

At a high level, the electrical system is responsible for distributing power, ensuring reliable communication, and coordinating control between subsystems. This year, our focus was on improving reliability, safety, and functionality. In order to do so, we changed the communication architecture from I2C (Inter-Integrated Circuit) to CAN (Controller Area Network) bus. The addition of contactors, along with improved Battery Management System (BMS) boards, enhances protection for the batteries, boat, and surrounding environment. In terms of functionality improvements, we have added high-power mechanisms power supplies, a 915MHz radio to communicate with the onboard Jetson computer, a logger board for easy debugging, as well as an first-person view (FPV) video system to improve ease of operation during manual control. A full system diagram can be found in Appendix J

1) Communication: During last year's competition, we experienced connection issues with WiFi. To improve this, we use a Ubiquiti bullet connected to a 2.4 GHz Yagi antenna in order to maintain longer-range WiFi connection. Due to the cluttered RF environment on the WiFi bands during competition, we also use Digi XBee XR 900 radios on the 915MHz ISM band for emergency stop (ESTOP) and data transmission, which utilize FHSS (Frequency Hopping Spread Spectrum) and feature built-in SAW (Surface Acoustic Wave) filters, allowing for a stable connection in noisy RF environments.

2) Emergency Stop (ESTOP): Last year, our ESTOP was implemented solely in the BMS, leaving the system susceptible to a failed short transistor. This year, we have redundant ESTOP mechanisms: both a transistor on the BMS or a contactor can shut off the battery power. All PCBs receive the ESTOP signal, which is an active low signal. This means that the ESTOP will be triggered in the case of a broken ESTOP wire. Both the remote ESTOP and button on the boat trigger ESTOP by pulling the line low through a transistor, whose gate is pulled up as a failsafe measure. On the shore side, a custom controller sends ESTOP data, as well as manual drive data. If ESTOP is activated on the shore side, or connection is lost, ESTOP is triggered on the boat.

3) Battery Management System (BMS): The BMS monitors battery health and performs battery shutoff. It protects against undervoltage, overcurrent, and cell imbalance. Last year, we implemented these functions with a microcontroller, leading to noisy measurements and unreliable performance. This year, we switched to the TI BQ76942, a battery monitor and protector for 3s-10s batteries. This chip provides many useful features. A built-in charge pump allows for a high side NFET, meaning communication with the rest of the system doesn't need to be isolated like the previous BMS. The DFETOFF pin can be directly attached to the ESTOP line, reducing the chance buggy firmware would fail to turn off the FET in case of ESTOP. Finally, various quantities, including stack voltage, cell voltages, current, and FET state can be accessed over I2C from a microcontroller unit, which then sends these values over CAN to the rest of the system.

4) Power: The boat is powered by three batteries: a 6-series (6s) battery for electronics, and two 4-series (4s) batteries for thrusters and high power mechanisms. The main power board contains buck converters that take the voltage from the electronics battery and convert it into 3.3V, 5V, 12V, 19V. The value of current provided by each of these rails is sent over CAN. We have two high power mechanisms boards used to power a ball launcher and pump, which feature a single DC-DC converter (using the TI LM251772 controller) that takes the voltage from a thruster battery and converts it to any voltage from 3.3V to 36V, with a max current of 25A. These converters feature adjustable voltage and current limits, as well as output current monitoring, all over the CAN bus.

5) Mechanisms: This year, we built a mechanisms testing board, allowing for the mechanical team to test the ball launcher and water delivery subsystems without the overhead of the autonomy stack. This board can control servos, steppers, and DC motors, and can be configured using potentiometers or through a simple UI. This board connects to the mechanisms power boards, allowing ground testing to be performed with the same power supplies as the boat. From there, a translation to the boat mechanisms board is simple, only requiring firmware to interface with the CAN system.

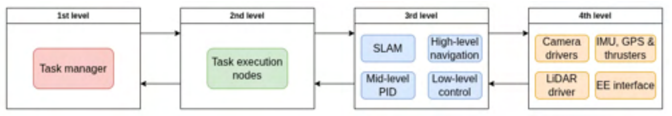

Fig. 4. An overview of the hierarchy of ROS nodes in our system.

C. Software System

Our autonomy system is based on the Robot Operating System 2 (ROS 2) [2] middleware and runs on a Jetson AGX Orin computer onboard. The ROS structure consists of nodes that perform specific tasks and messages passed to topics using a publisher-subscriber model. This enables running many processes in parallel without significantly trading off performance. This is important because it allows the fundamental subsystems—SLAM, mid and low-level control, and high-level planning and navigation—along with their dependencies (e.g., camera, LiDAR, and IMU driver nodes), to run concurrently in a modular architecture that can be extended to implement higher-level task logic. A diagram of our system architecture can be found in Appendix I.

Our system is split into 4 hierarchical levels, as depicted in Fig. 4. At the 4th and lowest level are the software drivers that expose interfaces with the robot's hardware, like the sensors and ESCs. The 3rd level is the fundamental subsystems described in Sections II-D and II-E, providing information about the robot's and course objects' positions, while exposing interfaces that allow other nodes to navigate to a set waypoint or perform other lower-level control strategies. These interfaces are implemented using ROS action servers, which ensure that we can reliably send a goal to any available process (service) and stay informed about its status at any time.

At the 2nd level are the task-specific nodes, which execute the algorithms described in Section I-A using the information and interfaces provided by the lower-level nodes. Then, at the 1st and top level is a task manager node that implements a state machine that calls the respective task nodes given the ones that have succeeded/failed before, handles their response (task couldn't start, aborted, or completed), and potentially calls a search node to find the next task to attempt.

D. Localization and Mapping

This year, we decided to treat localization and mapping as a single, unified problem and adopted a simultaneous localization and mapping (SLAM)-based approach. This is motivated by sensor inadequacies we observed during last year's competition, particularly GPS unreliability, and by realizing the course objects themselves can be useful for localizing the robot in the case of GPS drift. SLAM enables us to fuse measurements from multiple sensors over time, significantly

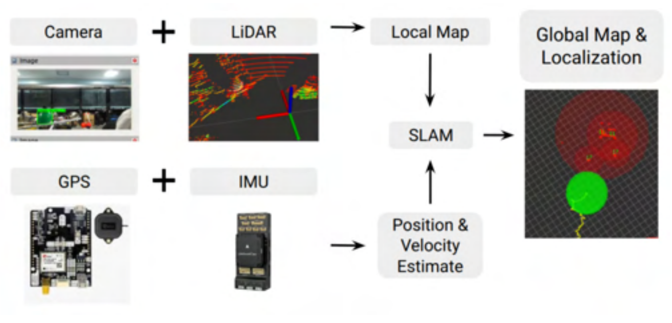

Fig. 5. Our perception, localization, and mapping pipeline.

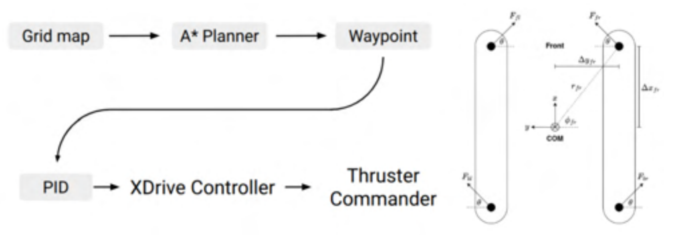

Fig. 6. Our high-level navigation and mid and low-level control pipeline.

reducing localization and mapping error and improving overall system robustness.

The two newly added RGB cameras gives us a combination of image and depth data over a ∼300° field of view. This gives us a significant boost in the accuracy and robustness of our localization and mapping stack. For object detection, we fine-tuned a YOLOv11n [3] model, which is a deep learning-based object detection model, ensuring robustness under various lighting conditions and viewing angles. We improved our LiDAR and camera fusion system by filtering out outliers from the point cloud, and incorporated Random Sample Consensus (RANSAC) [4] for plane detection of banners. More details on improvements to our perception system can be found in Appendix D.

This infrastructure allows us to create a local map of the environment relative to the robot and incorporate a globally consistent estimate of the robot's and course objects' positions. Tracking the global position of the buoys and banners using SLAM greatly reduced the number of duplicate/misplaced or mislabeled instances of observed obstacles in the robot's map, which had been greatly affecting task execution performance, and improved the overall object localization accuracy. This is also important given this year's competition's emphasis on reporting the global coordinates of the robot and the course objects, which are directly provided by our unified localization & mapping system depicted in Fig. 5.

E. Navigation and Controls

Given an up-to-date map of the course objects, the vehicle must navigate around those objects in the shortest time and without collisions. This demands the ability to perform both long-range navigation (e.g. moving between different tasks or between buoy pairs) and close-range dynamic maneuvers (e.g. avoiding previously unmapped or incorrectly mapped obstacles, circling light indicators, and accounting for GPS drift).

This incentivized splitting our navigation stack, depicted in Fig. 6, into three hierarchical levels: high-level, mid-level, and low-level control. Low-level control consists of converting a desired velocity (provided by the joysticks for manual control, or a ROS node for automatic navigation) to thruster values by solving for the physical constraints and resulting velocity of the X-Drive configuration.

Mid-level control achieves waypoint and path following, and is implemented using 3 independent PID controllers that output the robot's desired velocity in the x, y, and rotational directions, which is then fed into low-level control. Mid-level control also requires a fast and robust implementation of collision avoidance behavior that responds well to any potential obstacle, which is necessary for all tasks. This is implemented by computing a potential field [5] of the LiDAR point cloud, which is truncated and filtered using voxelized downsampling. More details on the implementation of the mid-level controller can be found in Appendix E.

Finally, in order to support high-level waypoint navigation, we maintain a planar grid map that details the valid locations throughout the course that the vehicle can traverse, continuously updating it using the unlabeled LiDAR clusters and performing Bayesian Inference on the cell occupancy probabilities. Then, to enable obstacle-avoidance waypoint navigation, we plan a path from the current vehicle location to the specified waypoint using the A* algorithm on the grid map. The A* algorithm allows transitions from the current cell to any cell within a 2-cell radius, guided by the Euclidean distance heuristic. The resulting path produced by A* is then downsampled to reduce jagged corners created by the transitions. We are also replanning the path when large inconsistencies occur in the global map.

III. Testing Strategy

To validate our changes to the autonomy and electrical stack, Arcturus tested Fish 'N Ships extensively. Our testing strategy was broken into three components: subsystem testing to validate system functionality prior to integration, simulation testing as a low-cost method to validate autonomy development, and full-system testing on water to assess the integration of the full system. Details on test plans can be found in Appendix A, and test results can be found in Appendix B.

A. Testing Timeline

From March to August, we focused on testing fundamental improvements to our perception and localization stack, which we validated using the sensor cart and in simulation. From September to December, we conducted tests on the water and focused our tests on navigation-heavy tasks, such as Evacuation Route & Return, Debris Clearance, and Emergency Response Sprint. During this time, we also began the testing and development of our delivery mechanisms. From January to

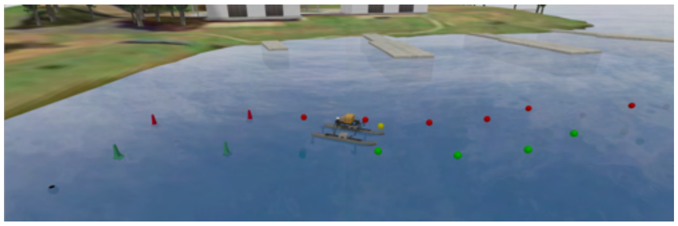

Fig. 7. Our simulation testing environment.

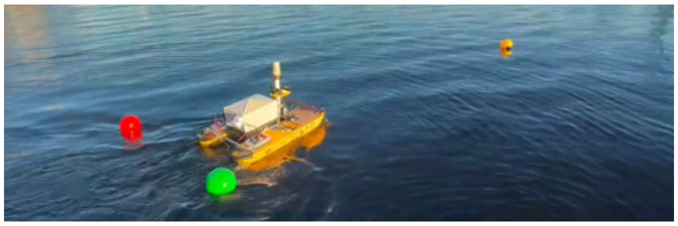

Fig. 8. Testing Emergency Response Sprint outdoors.

February, we conducted indoor tests and focused on finalizing task approaches and sequential task execution.

B. Subsystem Testing: Mechanisms

The short turn around between tasks release and competition this year required compressing the mechanisms timeline. To achieve this, Arcturus adopted a gray-box modeling strategy. Rather than developing full subsystems, testing, and iterating, the team built mathematical models based on idealized physics, calibrated for non-ideal behaviors using targeted, proof-of-concept tests instead of full parameter sweeps, and used the calibrated models to develop robust mechanisms. More details on our mechanism testing approach can be found in Appendix C.

C. Simulation Testing

In order to rapidly iterate on our software stack and enable the development of more advanced algorithms without the additional overhead of outdoor testing, we conduct simulation testing using the Virtual RobotX (VRX) Gazebo simulator, depicted in Fig. 7. We created custom Simulation Description Format (SDF) files to test our algorithms on various tasks and course layouts, testing the robustness of our algorithms in a reproducible manner before deploying them on the robot and putting it on the water. Testing the robot in simulation was also important for mitigating the risk of undefined behavior that could result in damage to the robot, the dock we are testing next to, and other equipment that's important for testing, like the buoys and their anchors.

D. Outdoor Testing

Every 1-2 weeks, we conduct full-scale integration tests of the simulation-tested autonomy subsystems on the Charles river (see Fig. 8). Learning from our GPS issues last year, we prioritized testing localization approaches including SLAM and other sensor fusion algorithms early in the design cycle.

After we have ensured fundamental localization functionality through multiple tests in varying outdoor conditions, these outdoor tests progressed towards running and logging practice runs of the RoboBoat tasks such as Debris Clearance and Emergency Response Sprint. These tests allowed us to verify individual task capabilities with the physical hardware setup. Issues discovered in these integration tests are usually caused by outdoor testing factors, such as GPS signal, IMU feedback, and lighting conditions, that are difficult to identify in simulated testing.

E. Indoor Testing

Because of freezing temperatures, our last few months of testing from December to January are in an indoor pool. These tests have similar goals to the outdoors ones, but with greater focus on finalizing the algorithms of all task approaches and testing sequential task execution.

A key limitation of indoor tests is the lack of GPS satellite visibility, requiring a different localization approach. Our SLAM-based localization stack allows us to remove GPS position measurements and replace Pixhawk velocity estimates with LiDAR-based RF2O odometry [6], fused with IMU acceleration using an EKF [7]. This allows the system to produce consistent position estimates, enabling effective indoor testing without GPS.

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, Anduril, Cadence, and MIT Gordon Engineering Leadership Program.

We would also like to acknowledge those who donated materials/tools or lent us testing facilities: Formlabs, PTC, Ansys, MIT Sea Grant, MIT Sailing Pavilion, MIT Marine Autonomy Lab, MIT AUV Lab, MIT Architecture and Design (MAD) Lab, N51 Edgerton Center Team Shop, Edgerton 6C Student Shop, the AMP Graduate Machine Shop, 4-409 Edgerton Student Project Laboratory, and MIT Information Systems & Technology.

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 and Professor Dave Barrett's insights regarding all things hydro and manufacturing. We would also like to extend thanks to all of our alumni who continue to support us with guidance.

References

[1] I. Rusiecki, C. Wieczorkowski, T. Ujazdowski, et. al., "RoboBoat 2025: Technical Design Report SimLE SeaSentinel Team," RoboNation, March 2025, https://robonation.org/app/uploads/sites/3/2025/02/TDR_SeaSentinel_RB2025.pdf.

[2] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, 'Robot Operating System 2: Design, architecture, and uses in the wild', Science Robotics, vol. 7, no. 66, 2022.

[3] G. Jocher, J. Qiu, and A. Chaurasia, "Ultralytics YOLO (Version 8.0.0)," 2023, https://github.com/ultralytics/ultralytics

[4] M. Fischler and R. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” Association for Computing Machinery, New York, NY, USA, 1981, doi:10.1145/358669.358692.

[5] E. Konstantinova and A. Kravchuk, “Distinct eigenvalues of the Transposition graph,” arXiv, 2023, https://arxiv.org/abs/2306.01627.

[6] M. Jaimez, J. Monroy, and J. Gonzalez-Jimenez, “Planar Odometry from a Radial Laser Scanner. A Range Flow-based Approach,” 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 4479-4485, doi: 10.1109/ICRA.2016.7487647.

[7] T. Moore and D. Stouch, “A Generalized Extended Kalman Filter Implementation for the Robot Operating System,” Annual Meeting of the IEEE Industry Applications Society, 2014, https://docs.ros.org/en/lunar/api/robot_localization/html/_downloads/robot_localization_ias13_revised.pdf.

[8] MHSeals, ‘Buoys Dataset’, Roboflow Universe. Roboflow, Dec-2025.

[9] J. García, J. Molina, and J. Trincado, “ A Methodology for Design and Analysis of Sensor Fusion with Real Data in UAV Platforms,” preprints, 2018, 10.20944/preprints201801.0077.v1.

[10] F. Raheem and M. Badr, “Development of Modified Path Planning Algorithm Using Artificial Potential Field (APF) Based on PSO for Factors Optimization,” 2017, American Scientific Research Journal for Engineering, Technology, and Sciences. 37. 316-328.

[11] K. Koide, S. Oishi, M. Yokozuka and A. Banno, ”General, Single-shot, Target-less, and Automatic LiDAR-Camera Extrinsic Calibration Toolbox,” 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 11301-11307, doi: 10.1109/ICRA48891.2023.10160691.

Appendix A
Test Plans

In order to verify the effect of the significant changes performed on our vehicle since last year’s competition, and ensure adequate performance and reliability of every subsystem on our robot, we had to adopt an extensive testing timeline of over 160 hours of simulation and 80 hours of physical testing.

Testing was split between subsystem testing and full-system integration testing, with separate objectives for each, as detailed in Table I.

For every test, we wrote up a list of objectives, results from the test, and key takeaways. Detailed documentation from every test we conducted can be found on our team website under the testing blog.

Subsystem testing is mostly performed within each subteam (Autonomy, Electrical, and Mechanical). We perform subsystem testing with small-scale tests (physical or simulation) that verify that a newly developed or modified subsystem is working properly and in a way that’s aligned with the team’s competition goals. This ensures time efficiency and enables rapid iteration on individual components of the boat without hindering the development of other ones.

Full-system testing focuses on the integration of all of the robot’s subsystems and the autonomous performance of competition tasks. These require coordination between all of the subteams to ensure that all of the tested subsystems are reliably working, and to debug any issues that arise.

A. Simulation Tests

All of our autonomy software, after being implemented, is first tested in simulation before it’s submitted to physical subsystem or integration tests. Using the Virtual RobotX (VRX) Gazebo-based simulation environment, we can rapidly iterate over our algorithms and ensure that they perform as desired in a sandbox environment before running them on the robot.

This mitigates the various hazards that could arise from running untested code directly on our robot, significantly reduces the development time by eliminating the overhead that’s related to setting up and running the boat on the water, and allows for simultaneous development and testing of the algorithms by all Autonomy members.

This simulation environment requires a relatively powerful Linux computer to run, and thus presented a significant bottleneck, since not all of our members’ computers satisfied these requirements. We found three workarounds for this problem. First, we are utilizing some of the Toughbook laptops available at MIT Sea Grant, dual-booted with Linux, which are set up to be able to run the simulation environment and test our codebase.

Second, we developed a minimalistic version of the simulation environment that only incorporates the components required for the design and testing of task execution algorithms, removing most of the computational burden of running the entire simulation environment. We packaged this simulation environment, along with our codebase and the required ROS dependencies, into a Docker container, enabling its rapid setup and use on any computer, proving extremely useful when we were onboarding new team members and familiarizing them with the codebase.

Finally, in collaboration with MIT Information Systems & Technology, we were able to acquire powerful loaner laptops, which we dual-booted with Linux and set up appropriately, for our team members, in order for them to be able to develop and test more advanced algorithms in the VRX simulation environment, ensuring adequate performance and robustness before using them in full-system tests.

B. Physical Tests

Testing individual subsystems on land, as well as the entirety of our system in the water, and evaluating its performance in completing competition tasks under suboptimal conditions has been of great focus this year. This decision was driven by inadequacies presented during last year’s competition, thus we wanted make sure that our system is robust this year.

The first issues that were targeted were those related to our perception and localization stack, as well as a complete overhaul of the Electrical system to improve its reliability and modularity. In order for those to be addressed in parallel, and without slowing each other down, we had to come up with a way to test our perception stack independent of the rest of the boat. This led to the design and construction of a mobile sensor mast test cart, presented in Fig. 3 and described in Section II-A1. This enabled extensive development and testing of a new perception (including the addition of two cameras), localization, and mapping stack over the summer with increased efficiency, since it also eliminated the overhead associated with in-water testing while being almost equivalent in terms of environmental conditions and lessons learned.

Then, once the electrical boards were redesigned and the new perception system was ready to test in the water for the completion of navigation tasks, the majority of our testing happened in the Charles River, using the infrastructure provided by the MIT Boathouse and the MIT Sailing Pavilion, where we could place our equipment in the respective docks. There, we were able to set up parts of the competition course to test the capabilities of our system.

This involved anchoring Polyform buoys of different colors, using weights tied to them with rope, in a configuration similar to that of Tasks 1, 2, and 3, testing Tasks 1 & 2 consecutively and Task 3 separately. We also constructed and used light indicators, with the support structure beneath them and the addition of floaters, in order to test advanced capabilities for Tasks 1, 2, and 3.

To test Task 4, we used banners with the appropriate shapes placed onto them, as well as props simulating the delivery vessels. Task 5 testing was based on similar banners (with the numbers on them instead), incorporating a foam dock with a single slot and the addition of the light indicator during our latest tests.

When outside conditions, specifically cold weather and the Charles River freezing, prohibited testing in the river, we moved to indoor testing using the facilities of the MIT Zesiger Sports and Fitness Center, specifically utilizing 3 lanes of an indoor pool that was available to use for research experiments. This presented the difficulty of lacking GPS satellite visibility, which we resolved by using the same SLAM framework we are using for outdoor navigation in combination with RF2O LiDAR-based odometry, as described in Section III-E.

Physical testing incorporated the need for taking safety measures and ensuring they are met at all times, by being in constant communication and collaboration with the managers of the MIT Boathouse, Sailing Pavilion, and the Zesiger Center. This involved wearing life jackets when on the dock, verifying our boat's remote emergency stop and thrust control before deploying it, and implementing a low-battery alert system that requires user acknowledgment and ensures timely battery swap.

Appendix B
Test Results

Below, we describe the three stages of our system testing timeline, detailing our objectives and results, including problems we encountered.

A. *Early Stage Testing*

Focus: Perception & Localization and Mapping

Dates: April 2025 – Aug 2025

Objectives: Our goal was to overcome the issues presented during Roboboat 2025 by implementing and adopting a Simultaneous Localization And Mapping framework described in Section II-D. We tested the incorporation of two additional cameras onto a new sensor mast, and the performance of SLAM, using the sensor mast cart presented in Section II-A1.

Results: After successful calibration of the cameras with the LiDAR and configuration of the Pixhawk and the GPS (including the radio link between the base station and the onboard GPS) to acquire adequate odometry measurements, we successfully got an implementation of SLAM working on land. The testing setup involved placing buoys on the ground, around the cart, and moving it by hand to verify successful localization & mapping, with minimal duplicates, at slow speeds.

Some issues encountered were:

- Difficulties in target-less calibration of the cameras with the LiDAR using Direct Visual LiDAR Calibration [11], requiring experimentation with different environments and settings. Required fine-tuning by hand, evaluated by projecting the camera image detections onto the point cloud.
- Configuring the GPS RTK system to acquire corrections from the base station at a sufficient rate, and achieving sufficient accuracy without surveying for really long times (which is hard to do in competition).
- Ensuring the alignment of the compass orientation and the Pixhawk coordinate frames with the global frame, to be able to fuse odometry with GPS.
- Experimentation with various methods for fusing odometry with GPS and ensuring robustness to high-frequency vibrations, resulting in the selection of the internal Pixhawk EKF3 framework as the way to go.
- Optimizing the sensor fusion algorithm to mitigate the selection of background points in the point cloud.

B. *Middle Stage Testing*

Focus: Navigation & Controls for Tasks 1,2,3

Dates: Sept 2025 to Dec 2025

Objectives: After getting the SLAM framework to work on land, we were ready to test navigation and controls of the boat outside, in order for it to complete competition tasks 1, 2, and 3 (initially with last year's tasks as a reference, and then incorporating this year's additions).

Results: After optimizing the sensor fusion and SLAM algorithms in terms of both performance and robustness to suboptimal conditions, the localization and mapping accuracy and speed were adequate for navigation and task execution. This, combined with improvements to the algorithms used to identify and keep track of the buoy pair path, starting gates, and buoys (or light indicators) to circle, including adaptive replanning and a new path following logic as described in Appendix E, enabled successful completion of tasks 1,2, and 3, verified through many runs.

During that process, we encountered the following issues:

- SLAM was initially only working at slow navigation speeds, due to improper synchronization of the topics the node subscribed to and incorporation of the published data (especially odometry). This was resolved with a timeout logic, pre-integration of odometry in a separate node, and utilization of differential GPS measurements as an odometry supplement.

- In the water, we encountered various issues with buoy detection, like red buoys being detected as yellow under glare, dark areas of background being detected as green buoys, and the reflection of the buoys on the water being picked up as well. These issues were resolved by collecting data from in-water tests, hand-labeling them, and fine-tuning our YOLO model.
- There were some issues with sensor fusion and the distinction between buoys and the background, as well as partially occluded buoys that appeared in the bounding box of others, which were resolved by fine-tuning the fusion algorithm.
- The clustering of raw LiDAR point clouds, used for SLAM (partially) and grid map computation, resulted in an inaccurate representation of the free space in our testing setup next to one of the docks, requiring fine-tuning and additional voxelized downsampling before clustering.
- There were some cases where identifying the buoy pairs was not working properly in the presence of duplicates or when the first pair identified was ”diagonal” with respect to the path, requiring improved pair identification and refinement strategies.
- In cases where there was drift in the global map (due to localization inaccuracy at the beginning of the run, which is mitigated as SLAM processes more data), we needed to replan the computed path using a new position of the respective waypoint.
- Our old path following logic, which consisted of one-at-a-time waypoint following with a distance threshold to switch to the next one, resulted in frequent acceleration and deceleration of the boat and significant shortcutting, so we switched to a new continuous-setpoint controller described in Appendix E.

C. Late Stage Testing

Focus: Indoor testing of Tasks 4,5 and full-course runs

Dates: Jan 2025 - Feb 2025

Objectives: Having developed and verified our navigation logic and succeeded in consistently performing tasks 1, 2, and 3, our next priorities were the more banner-based tasks 4 and 5. Due to the cold weather and subsequently the Charles River freezing, we had to perform most of our late-stage testing indoors, in a pool. Thus, an additional objective and prerequisite for this stage of testing was using RF2O LiDAR-based odometry to achieve indoor, GPS-denied localization, mapping, and navigation using our SLAM framework. Finally, we wanted to ensure the consistent execution of tasks in sequence and as part of full-course runs.

Results: After we managed to get indoor SLAM and navigation working, we first tackled navigating to and stationkeeping in front of banners, which is our key strategy for tasks 4 and 5. Then, we incorporated our custom-made docking slot and fine-tuned our potential field-based obstacle avoidance controller to be able to dock properly, without crashing the boat. Finally, we implemented and verified new perception-focused components for the advanced capabilities of this year's tasks, like light indicator and banner number detection, while continuously testing consecutive execution of tasks and improving our strategy for identification of the tasks within a full-course run. This included the implementation and testing of task aborting and restarting capability that's required for the successful completion of Task 6.

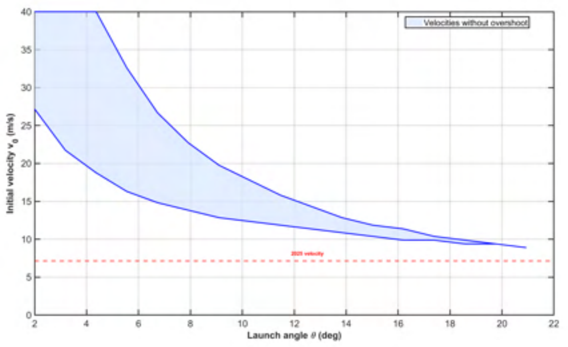

Fig. 9. Based on an estimated target height of 0.5m and desired maximum delivery range of 5m, the team used drag-included projectile motion analysis to calculate the region of workable velocities and launch angles. It is clear from this analysis that last year's fixed-angle, horizontal flywheel launcher, which reached a measured v_0 of 7.15 m/s would not meet these design requirements.

Appendix C
Mechanism Testing

A. Projectile Motion: Air Drag

To determine what angle to operate our fixed angle launchers at, and to help define the target exit launch velocity we should design for, our team performed a projectile motion analysis of the racquetballs with air drag. The dynamics of the ball's motion, simplified to a 2D problem, are:

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}, \quad \dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ -\frac{1}{2}\frac{\rho}{m} C_{dx} A\, \dot{x}\, |\dot{x}| \\ -g - \frac{1}{2}\frac{\rho}{m} C_{dy} A\, \dot{y}\, |\dot{y}| \end{bmatrix}$$

where C_d in both x and y is velocity dependent. For simplicity, we assumed C_d of the ball to be either 0.5, in laminar flow, or 0.25, in turbulent flow, with a transition at $Re = 3e5$. By sweeping through feasible launch angles, $2° \leq \theta \leq 60°$, and feasible (and somewhat infeasible) exit velocities, $0 \leq v_0 \leq 40$ m/s, and calculating the trajectory of the ball at each swept point using the state space matrix defined above and Matlab's ode45, we determined the region of valid exit velocities and launch angles, as recorded in Fig. 9.

This analysis was validated by empirical tests with the 2025 ball launcher which showed that it could at best reach a target 3m away without overshooting the 0.5m height maximum.

B. Pneumatic Ball Launcher Model Calibration

The team's limited pneumatics experience meant getting a working proof-of-concept of the pneumatic launcher as fast as possible was extremely desirable. In conjunction with the proof-of-concept barrel, the team also developed a mathematical model to gain insight into how to improve launcher performance without performing a laborious and time consuming parameter sweep.

The dynamics of the system are described as follows:

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{U} \end{bmatrix} = \begin{bmatrix} v \\ \dfrac{A(\frac{U}{V_0 + A\,x})(\gamma - 1) - P_{atm}}{m_{ball}} - \dfrac{c}{m_{\text{ball}}} v \\ -\dfrac{U}{V_0 + A\,x}(\gamma - 1)\,A\,v - c\,v \end{bmatrix},$$

where the first two ODE's come from Newton's second law and the third ODE come from applying the first law of thermodynamics to a constant mass of expanding ideal gas. Crucially, this description of the system relies on a constant mass of expanding gas, i.e., no leakage past the sphere. This model leaves two parameters for tuning. The first is $P_{\text{diss}} = c * v^2 + F_f$. F_f is well understood and fairly small compared to losses due to linear friction. The second parameter is an overall efficiency parameter η which is implemented to capture non ideal losses in work transfer between the gas and the ball. To tune these parameters we again ran a parameter sweep using the math and then verified the output using a couple of targeted empirical tests.

Using compressed air from the shop compressor we fired two shots, one from a long-barreled chamber and one from a short-barreled chamber. We then performed a test where we ramped up pressure in the chamber slowly (using a regulator) to determine the amount of pressure required to initiate movement. We found that with the short barrel the launcher reach nearly 5.5m while with the long barrel it reached barely 1.5 m. We also found that the ball began moving, and thus adiabatic expansion, began when pressure in the chamber reached about 50 PSI. Using these data points we estimated that the launcher had an overall efficiency of about $\eta = 0.1$ and linear damping coefficient $c \approx 10$. The parameter sweep is recorded in Fig. 10 and informed a slightly shorter barrel selection and sizing of the upstream components of the system, including reservoir size.

C. Vertical Fly Wheel Model Calibration

We develop the following state space model of the vertical flywheel ball launcher:

$$\mathbf{x} = \begin{bmatrix} x \\ v \\ \Omega \\ \omega_1 \\ \omega_2 \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{\Omega} \\ \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} v \\ \frac{F_1(x,v,\Omega,\omega_1) + F_2(x,v,\Omega,\omega_2)}{m} \\ \frac{r_b[F_1(x,v,\Omega,\omega_1) - F_2(x,v,\Omega,\omega_2)]}{I_b} \\ -\frac{R_w F_1(x,v,\Omega,\omega_1)}{I_w} \\ -\frac{R_w F_2(x,v,\Omega,\omega_2)}{I_w} \end{bmatrix} \tag{1}$$

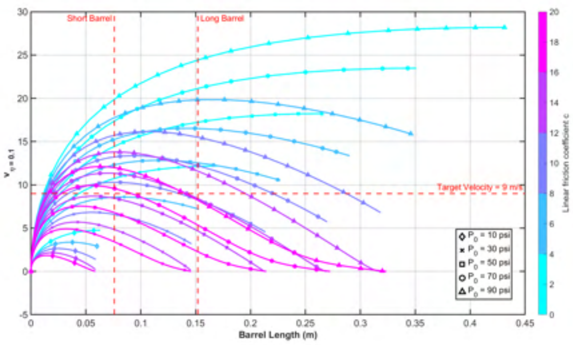

Fig. 10. Based on an estimated initial pressure of 50 PSI and the difference in behavior between the two barrel lengths we assumed the launcher had an overall efficiency of ≈ 0.1 and damping coefficient of ≈ 10. This informed design with a slightly smaller barrel and sizing of the upstream components.

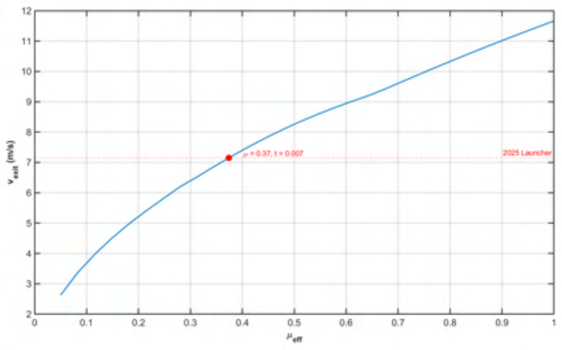

Fig. 11. Based on our model of the launcher dynamics and empirical tests of last year's launcher we were able to obtain a good estimate of the effective friction coefficient between the ball and the PLA flywheels and contact time.

where F_1 and F_2 are defined as non linear functions of compression, δ.

$$F_1(x,v,\Omega,\omega_1) = \mu\,N(x)\,\tanh\left(\frac{R_w\,\omega_1 - (v + r_b\,\Omega)}{v_0}\right) \tag{2}$$

$$F_2(x,v,\Omega,\omega_2) = \mu\,N(x)\,\tanh\left(\frac{R_w\,\omega_2 - (v - r_b\,\Omega)}{v_0}\right) \tag{3}$$

$$N(x) = K\,\delta(x) \tag{4}$$

δ is derived from geometry and K is derived from empirical tests of the ball's stiffness using an instron.

Tests of last year's ball launcher showed it reached an exit velocity of about 7.15 m/s. By plugging the design parameters of last year's ball launcher into our model of the launcher dynamics we were able to calibrate the one remaining free parameter of the model, μ_{eff}, the effective friction coefficient. The results of that calibration are presented in Fig. 11.

The resulting estimate of the effective friction coefficient $\mu_{eff} = 0.37$ is quite reasonable and helped both validate and complete our modeling strategy.

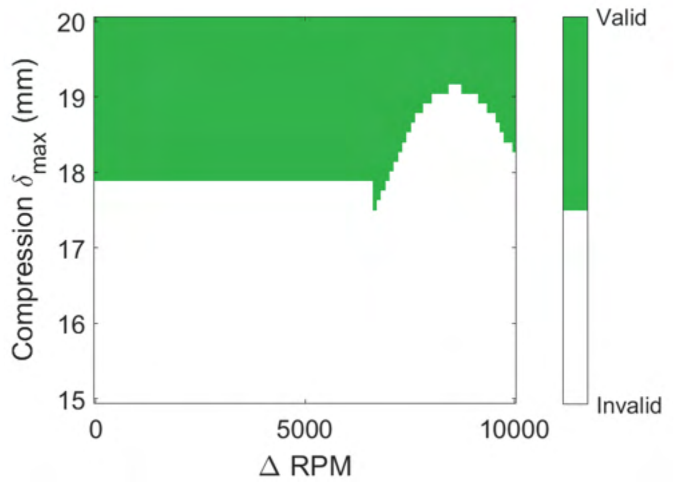

Fig. 12. Based on our calibrated model of the launcher dynamics, we were able to visualize the design region in which trajectory and range constraints were met. The drop in required compression at $\sim 6000RPM$ represents an optimal operation point because smaller compression corresponds to decreased load on the system.

Having decided to reuse the motors from the 2025 launcher for the 2026 launcher, we only needed to determine the optimal compression and backspin to achieve a 5m range without exceeding a trajectory height of 0.5m. To do this, we swept our model across different compression, δ, and differential flywheel spin, ΔRPM, and elevation angles, α, values and recorded ordered pairs that satisfied the range and trajectory heights for any elevation angle. The results of that sweep are reported in Fig. 12.

To finalize the design we needed a good understanding of the engineering trade-offs involved in choosing different ΔRPM hidden in the binary valid-invalid of Fig. 12. It is clear that minimizing δ_{max} is desirable because it minimizes the considerable reaction loads on the flywheels, motors, lead-screw, and the lead-screw servo required to compress the ball. We therefore fixed compression at 18 mm and compared the fractional increase in range and decrease in trajectory apex to the launch angle, θ, and difference in motor RPM, ΔRPM, as a heuristic for ball spin. The results are reported in Fig. 13.

These results informed our final decision to design the vertical flywheel launcher with a maximum compression of $\delta_{max} = 18$ mm, a launch angle of $\theta = 19°$, and a differential flywheel spin of $\Delta RPM = 6000$ between the lower and upper flywheels.

Appendix D
Perception

Identification of the various course objects and their respective colors or shapes is based on a pre-trained YOLOv11n model, which we fine-tuned using the MHSeals RoboBoat dataset [8] as well as images that we collected during last year's competition and system tests and labeled by hand, allowing us to resolve misdetections and false detections under various lighting conditions.

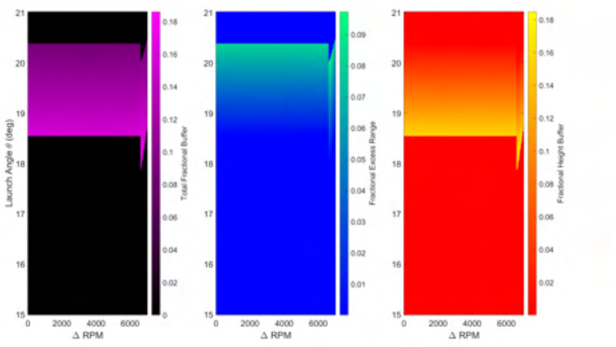

Fig. 13. Visualization of the fractional performance buffers for the flywheel launcher. *Fractional excess range* is defined as $\frac{range_{\text{max}} - range_{\text{desired}}}{range_{\text{desired}}}$, and *fractional height buffer* is $\frac{height_{\text{allowed}} - height_{\text{max}}}{height_{\text{allowed}}}$, where $range_{\text{desired}} = 5$ m and $height_{\text{allowed}} = 0.5$ m. *Total fractional buffer* is the sum of these two quantities and represents overall excess performance. Larger total fractional buffer values are desirable, as they provide margin to accommodate inefficiencies in the real design.

Then, using those bounding box detections, fusion with the LiDAR data to acquire range information for the respective objects is handled differently for buoys/indicators and banners, utilizing distinguished characteristics of those objects. The main characteristic of our algorithm is projecting the LiDAR point cloud onto the image planes to acquire the points corresponding to each object, and using Euclidean distance-based clustering to filter-out outliers (background, buoys partially overlapping with the bounding box, and reflection of robot splashback and leaves on the water). For buoys/poles, we additionally acquire the color of each point and filter out those that don't match the YOLO-provided label, and, for banners, we apply a Random Sample Consensus (RANSAC) [4] plane detection algorithm to get the center, size, and normal direction of the banner.

To tackle the challenge of accurately localizing the robot and the course objects, we designed a custom Extended Kalman Filter-based SLAM algorithm, allowing for optimizations based on the course structure and issues observed during testing. We use the position measurements provided by the GPS, and predict intermediate positions using velocity estimates provided by the Pixhawk's internal EKF3 [9] system, incorporating the GPS position to account for IMU acceleration measurement errors accumulating over time. We are also fusing object detections that are initially computed in the robot's coordinate frame (local map), as range and bearing measurements of landmarks that are incorporated in the SLAM state, keeping track and improving the accuracy of their position estimate over time.

Appendix E
Control

To compute the robot's velocity in order to reach a given setpoint, we implement 3 independent PID controllers that take in the x, y, coordinates of the setpoint in the robot's frame, and the theta orientation difference, as the respective errors,

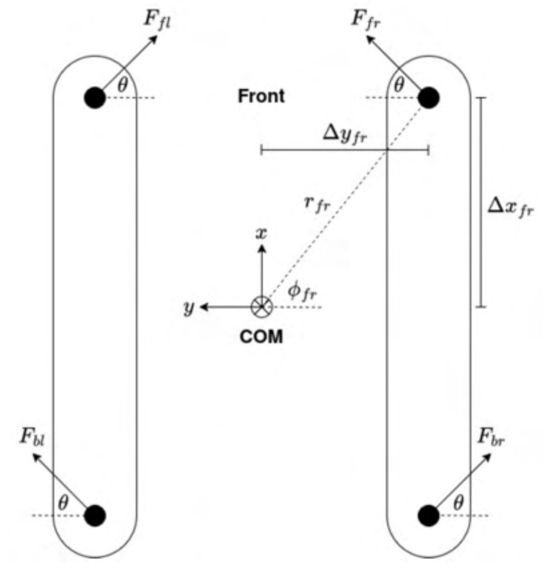

Fig. 14. The X-Drive configuration we are using to achieve holonomic motion of our robot.

and output the robot's x, y linear velocity (in its local frame) and the angular velocity omega, as depicted in Eq. 5-10.

$$e_x = (x_d - x)\cos\theta + (y_d - y)\sin\theta \tag{5}$$

$$e_y = -(x_d - x)\sin\theta + (y_d - y)\cos\theta \tag{6}$$

$$e_\theta = \theta_d - \theta \tag{7}$$

$$v_x = K_{x,p}e_x + K_{x,d}\dot{e_x} + K_{x,i}\int e_x dt \tag{8}$$

$$v_y = K_{y,p}e_y + K_{y,d}\dot{e_y} + K_{y,i}\int e_y dt \tag{9}$$

$$v_z = K_{z,p}e_z + K_{z,d}\dot{e_z} + K_{z,i}\int e_z dt \tag{10}$$

In order to convert the desired velocities to thruster values, we are computing the minimum norm solution of the linearized system that describes the physics of the boat and constrains its velocity to the desired one, as in Eq. 11, Fig. 14. $b_T,\ T \in \{\mathrm{fl}, \mathrm{fr}, \mathrm{bl}, \mathrm{br}\}$ are the maximum forces that the thrusters can exert (approximately 40N for the Blue Robotics T200 thrusters that we are using), r_T are the distances of the thrusters from the boat's center of mass, and C_x, C_y, C_z are the drag coefficients, which are empirically determined.

Achieving collision avoidance behavior can be achieved by defining a 2D cost function in the robot's local frame, increasing as we get closer to an obstacle (Eq. 12-13, Fig. 15), and creating a vector field by taking the negated gradient of the point field, giving us the optimal direction to avoid collisions. Adding the value of the vector field computed in the robot's position to the velocity output of the PID controller and weighing it appropriately results in the desired collision avoidance behavior.

$$U(\mathbf{q}) = \frac{1}{2}k \sum_{\substack{\mathbf{q}_{\text{obs}} \\ d(\mathbf{q},\mathbf{q}_{\text{obs}}) \le d_0}} \left(\frac{1}{d(\mathbf{q}, \mathbf{q}_{\text{obs}})} - \frac{1}{d_0} \right)^2 \tag{12}$$

$$\mathbf{v}_r = -\nabla U(\mathbf{q})\big|_{\mathbf{q}=\mathbf{q}_{\text{robot}}} \tag{13}$$

Finally, in order to enable smooth and fast path-following behavior, instead of following the path waypoints one by one, which resulted in frequent acceleration and deceleration, we are using a modified PID controller similar to the mid-level controller described above. In this case, we are using a continuously set waypoint, placed at a certain distance on the tangent to the path to its point that's closest to the robot, resulting in smaller deviations from the path and a consistent velocity along the path.

APPENDIX F
FALLBACK APPROACHES

In this section, we detail the logic for our fallback approaches to the tasks, which run in situations of sensor failure.

For Tasks 1 & 2, we are picking the closest red and green buoys (or poles), computing a setpoint in the robot's local frame as the midpoint of the segment connecting those buoys, and using an x,y, theta PID controller to navigate to it. We also take into account the black buoys and compute a waypoint between the black buoy and the green or red one in the case that it's in our path.

For Task 3, the fallback approach removes any dependency to waypoints. The approach uses multiple PID controllers to navigate towards the yellow buoy and circle it, using the camera stream as input to the PID controller.

For Tasks 4 & 5, the fallback approaches are really similar to the primary ones, since they only miss the high-level navigation part, and are just using a PID controller for the whole task, working with the local map (detections in the robot's frame) for banner/slot selection.

APPENDIX G
EXPLORATORY PROJECTS: 2027 VESSEL

In this section, we describe exploratory mechanical projects that were investigated during the season but not incorporated into the 2026 design. These efforts informed our understanding of design tradeoffs and may be revisited in future iterations.

A. Hull Design + Testing

Although Fish N' Ships is a functional boat, RoboBoat 2025 and post-season testing showed that several high-yield elements of the hull design could be improved.

- Tow tank tests after construction showed significant deck wash over the bow, increasing risk to deck-mounted electronics and ingress into the battery boxes and EE Box.
- The hulls were designed with three inches of freeboard at the desired loading of 65 lbs, but later modifications added weight, decreasing freeboard and worsening stern trim.

$$
\begin{aligned}
&u^\star = \underset{u\in\mathbb{R}^4}{\operatorname{argmin}}\|u\| \\
&\text{s.t. } u \in [-1,1], \quad Bu = Cv^2, \text{ where:} \\
&B = \begin{bmatrix} b_{fr}\sin\theta & b_{bl}\sin\theta & b_{fl}\sin\theta & b_{br}\sin\theta \\ b_{fr}\cos\theta & b_{bl}\cos\theta & -b_{fl}\cos\theta & -b_{br}\cos\theta \\ b_{fr}r_{fr}\sin(\theta+\phi_{fr}) & -b_{bl}r_{bl}\sin(\theta+\phi_{bl}) & -b_{fl}r_{fl}\sin(\theta+\phi_{fl}) & b_{br}r_{br}\sin(\theta+\phi_{br}) \end{bmatrix} \\
&u = \begin{bmatrix} u_{fr} & u_{bl} & u_{fl} & u_{br} \end{bmatrix}^T \\
&v = \begin{bmatrix} v_x & v_y & \omega_z \end{bmatrix}^T \\
&C = \begin{bmatrix} C_x & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & C_z \end{bmatrix}
\end{aligned} \tag{11}
$$

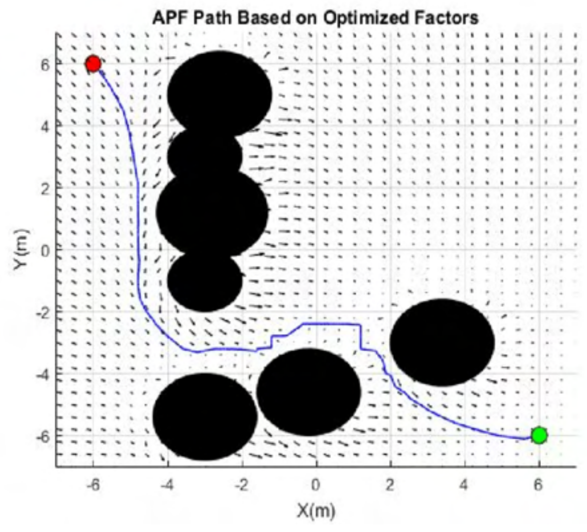

Fig. 15. A visual depicting the potential field-based obstacle avoidance approach we adopted. Image from [10]

- Fore-aft symmetry proved unnecessary since we rarely drive backwards.
- Reconstruction of the hulls without the foam interior could reduce weight and allow for the electronics to be placed inside the boat.

Given competition restrictions and lessons learned from the previous boat, the target weight for the 2027 design is 70 lbs, with a maximum allowable weight of 80 lbs, beyond which weight becomes detrimental to competition points. Since the thrusters will remain unchanged, the 2027 boat, like Fish ‘n Ships, is expected to operate in the displacement to semi-displacement regime (Fr = 0.39-0.67). Because only a small portion of the competition is speed-dependent, maneuverability was prioritized over speed. Competition rules limit the boat length to under 6 ft, and the hulls must be transportable by car. With these constraints in mind, the overall geometry was selected to balance stability, maneuverability, and transportability.

The team’s adoption of SLAM redoubled the importance of maintaining sensor visibility at all times by minimizing extreme motions even in the worst expected competition conditions. Excessive pitch or roll can cause temporary loss of LiDAR and camera visibility and increases water-ingress risk for deck-mounted electronics.

The existing X-configuration thruster system provides high maneuverability; although not the most hydrodynamic configuration, resistance was not a primary concern, and reconfiguration was avoided to reduce development time.

To satisfy transportability requirements, the team chose to limit maximum hull length to 4 ft. This limit was chosen based on past experience transporting the vessel and equipment to competition.

Design validation was conducted through a combination of CFD and towing tank testing. CFD was used to screen general hydrodynamic characteristics without the overhead of manufacturing a physical model. Physical validation was performed using modular SLA-printed hull sections, allowing rapid modification without rebuilding the entire model. Wave testing in head seas was used to evaluate pitching motion, deck wetness, and sensor visibility under worst expected competition conditions. Rolling or pitching exceeding 22.5 degrees results in complete loss of visibility, while motions between 10–15 degrees significantly reduce sensor range. Motion data were recorded using onboard accelerometer-gyroscope modules, and the model was ballasted to replicate expected full-scale loading.

B. *Hull Manufacturing + Testing*

Early in the post-season our team realized one of the most significant weight-deductions we could achieve would come from manufacturing hollow hulls using a negative mold. In addition to saving weight, hollow hulls enable sinking electronics into the hulls, resulting in a lower center of gravity and less windage with significant benefits for vessel stability and maneuverability. To determine the feasibility of this design the team conducted material and manufacturing tests. Material tests addressed questions of structural integrity. To determine how many layers of fiberglass were necessary to puncture-proof the boat, the team conducted three-point bending tests of fiberglass squares. We then used Hertz contact modeling and energy conservation to get a conservative overestimate of the forces experienced on collision with a rigid structure such as a dock. This analysis indicated that even at five layers hollow hulls would not be puncture proof. Manufacturing tests addressed questions of process. The team researched and conducted small scale trials of various mold releases, glass and carbon fiber weaves, and mold materials.

C. Sensor Mast Tilt Table

During the sensor mast redesign process, we realized that we did not have a rigorous understanding of how our perception stack's performance is affected by vibrations. To rectify this we designed a tilt table capable of forcing the sensor mast at different frequencies and amplitudes. By measuring the relationship between frequency, amplitude, and detection confidence we will be able to rigorously define design constraints for the 2027 sensor mast.

TABLE I
ROBOBOAT 2026 TESTING PLAN

Category	**Capability**	**Intended Testing Date**
Course setup	Figure out how to setup the course more reliably (buoy anchoring)	April 2025
Object detection	YOLOv11 buoy detection	June 2025
	YOLOv11 banner detection	Oct 2025
	YOLOv11 color indicator buoy detection	Jan 2025
Object localization - Buoy	LiDAR-camera fusion Single camera, implementation & working in sim	April 2025
	Single camera, working on sensor cart	May 2025
	Three cameras, synchronization and local map construction on sensor cart	June 2025
	Three cameras, working on boat in water	July 2025
	Minimizing occlusions & background picked up	Aug 2025
Banner localization - Banner	LiDAR-camera fusion Initial implementation working in simulation	Oct 2025
	Working in water	Nov 2025
Buoy SLAM	Initial implementation working in simulation	June 2025
	Working on sensor cart	July 2025
	Logically correct/mechanically working in water, slow speed.	Aug 2025
	Minimal duplicates & adequate performance for slow-speed navigation	Sep 2025
	External compass setup & integration w/ Pixhawk	Oct 2025
	Adequate performance for successful medium-speed navigation	Oct 2025
	Adequate performance for successful fast navigation	Nov 2025
	RF2O setup & running for indoor testing, successful navigation in varying speeds	Dec 2026
Banner SLAM	Initial implementation working in simulation	Oct 2025
	Working in water	Nov 2025
Task: Evacuation Route & Return	Initial implementation working in simulation	Sep 2025
	Working in water: vehicle can successfully navigate through the gates	Sep 2025
Task: Debris Clearance	Debris Clearance (follow the path) implementation & working in sim	Sep 2025
	SLAM approach (initial): boat can navigate through a sequence of red/green buoy pairs successfully, for a variety of paths (straight vs. curved).	Oct 2025
	Fallback approach (initial): boat can navigate through a sequence of buoy pairs without GPS.	May 2025
	SLAM + fallback approach (complete): boat can navigate through a sequence of red/green buoy pairs successfully, circle/avoid the debris, then return through the original path	Nov 2025
	SLAM approach: optimized path-following behavior & obstacle avoidance	Nov 2025
Task: Emergency Response Sprint	Initial implementation working in simulation	Oct 2025
	SLAM approach (initial): vehicle can successfully navigate around the target buoy.	Nov 2025
	Fallback approach (initial): vehicle can successfully navigate around the target buoy.	Oct 2025
	SLAM approach (improved): vehicle can successfully navigate around the target buoy starting from the red/green buoy gates and return to the red/green buoy gates successfully, and the vehicle is able to achieve a high speed.	Nov 2025
Task: Navigate the Marina (Docking)	Fallback approach: vehicle can align itself to the banner and dock successfully using only LiDAR and camera data	Dec 2025
	Complete approach: vehicle can use dynamic obstacle avoidance to dock successfully.	Dec 2025
Task: Supply Drop	Mechanism Controller Testing: test the controller of the turret which aims the water gun and ball launcher	Jan 2026
	Mechanism Navigation: the vehicle should be able to navigate to two banners in sequence, in order to complete the advanced capability for the supply drop.	Dec 2025
	End to end Mechanism Testing: combine the mechanism controller and mechanism navigation to complete the task of aiming and launching the water/ball at two targets sequentially	Jan 2026
Multiple Tasks	Sequential Task Execution: The vehicle can complete two tasks in sequence.	Jan 2026
	Full Course Testing: The vehicle can complete tasks in the full course.	Jan 2026
Mechanism Subsystem Testing	Verify Water Delivery	Nov 2025
	Calibrate Flywheel Model on 2025 Launcher	Nov 2025
	Pneumatic Feasibility Test	Dec 2026
	Calibrate Pneumatic Model on Proof of Concept	Dec 2026
	Pneumatic Mechanical Test	Jan 2026
	Flywheel Mechanical Test	Jan 2026
	Mechanisms Auto Test	Jan 2026

Appendix H
Annotated EE Box CAD

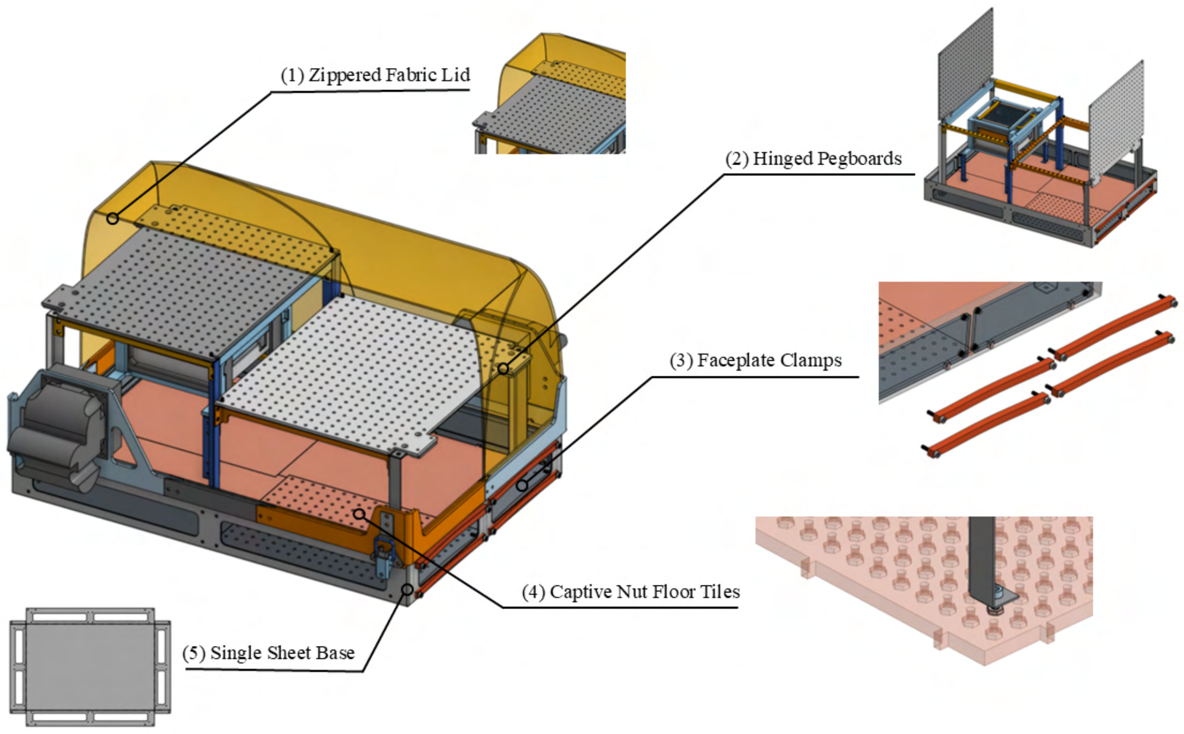

Fig. 16. New EE Box. Notable design updates include addition of a zipper (not pictured) to the fabric lid for dedicated on-water access; hinged pegboards for better bottom-layer access; faceplate clamps for easier connector swapouts without compromising splash protection; internal captive nuts; and single-sheet base design for ease of manufacture and splash protection.

APPENDIX I
SOFTWARE SYSTEM DIAGRAM

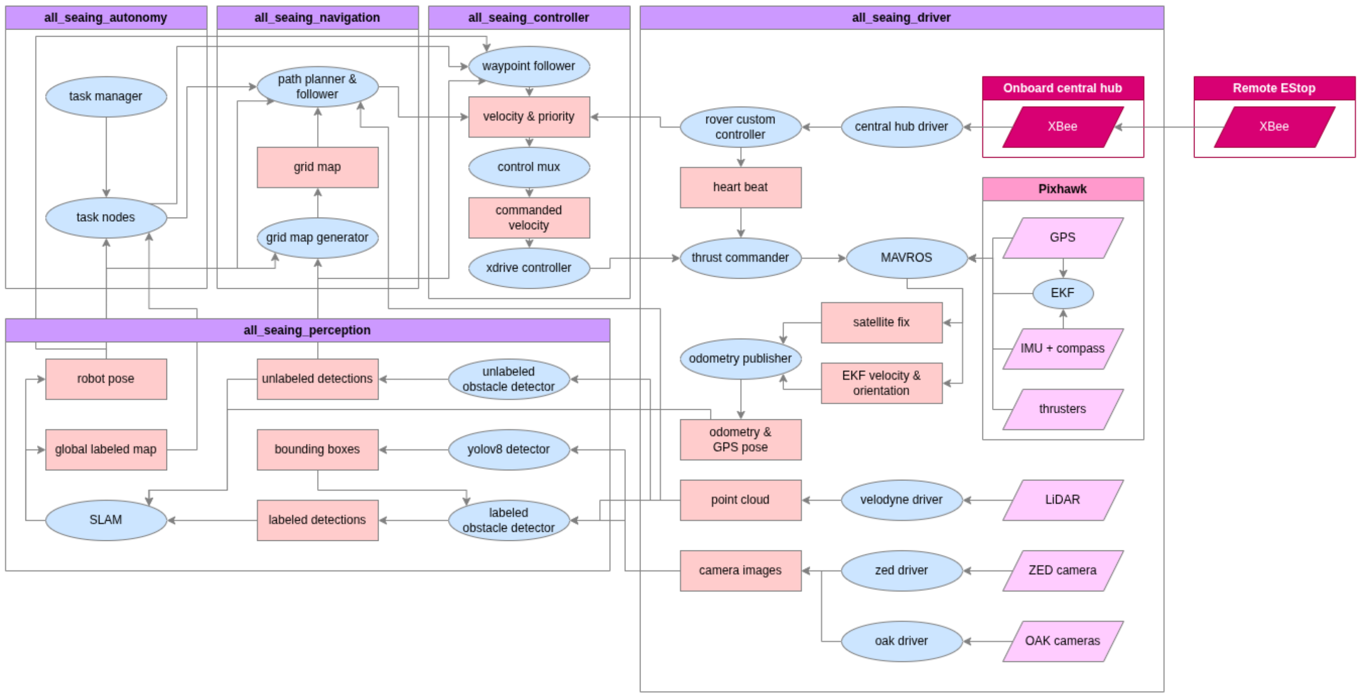

Fig. 17. A comprehensive diagram of our software architecture.

Appendix J
Software System Diagram

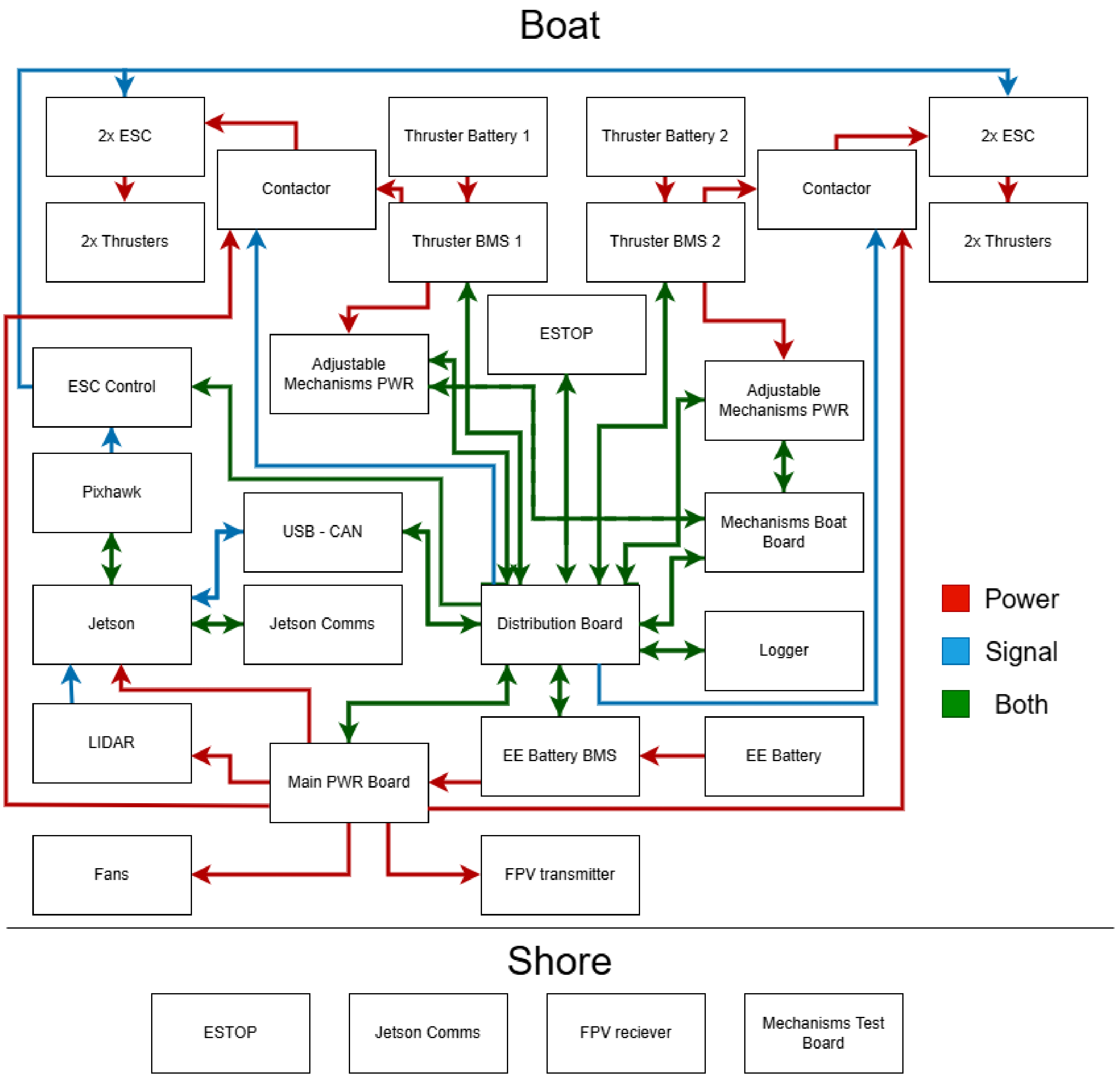

Fig. 18. A comprehensive diagram of our electrical architecture.