RoboBoat 2023: Technical Design Report

1st Audrey Chen, 2nd Herbert Turner, 3rd Ansel Garcia-Langley, 4th Jessica Lam
Massachusetts Institute of Technology, Cambridge, MA, USA
auds@mit.edu, hmturner@mit.edu, anselgl@mit.edu, jeslam@mit.edu

***Abstract*—Arcturus is a RoboBoat team representing the Massachusetts Institute of Technology. This technical design report explains the design of the autonomous surface vehicle (ASV) built for RoboBoat 2023, *Ship Happens*. The team approached every task to maximize our upper limit for point accumulation and give many opportunities for partial credit. We did not attempt the ocean cleanup task since the additional weight of a ball collection system and the cost of a hydrophone did not outweigh the potential additional points scored. Our design strategy was to create a robust simulation platform, a stable yet maneuverable hull design, and a water gun/ball launcher that performs consistently with our autonomous programming.**

I. Competition Goals

Last year was Arcturus' first year of competition and the experience provided the foundation for this year's competition. After an extensive team debrief, we settled on our strategy for the upcoming season: "Jack of all trades [tasks]." Although completing all of the course tasks is a tough challenge, the decision would allow for the best learning experience and provide each member of our ever-growing team the opportunity to make meaningful contributions. To ensure that we weren't spread too thin, we built upon what we had learned and developed last year. To allow us more time to focus on testing and building task-specific mechanisms, we reused our vessel *Ship Happens* for the next competition. However, due to the struggles we encountered during the competition, we overhauled our electrical and propulsion systems. We also decided to not perform the Ocean Cleanup Task because the additional weight of a collection system and the cost to purchase a hydrophone didn't justify the potential points. While we were able to complete only two tasks during the 2022 competition due to the amount of time we spent working on hardware, we believe our efforts showed the promise of our autonomy software and meant that we didn't need to start from scratch. The focus for the upcoming season is to complete all the modules that were unfinished, especially the implementation of machine learning in our perception system.

A. Course Approach

Because tasks and course objects can move around throughout the competition course, as seen in Figure 1, we don't have GPS locate tasks or even a planned task order within each run. Instead, we build global maps of our environment and rely on a global planner to determine where tasks are and how to start them. To do this, we rely on a state machine where our vehicle is in 1 of 3 modes: exploration, traveling, or completing.

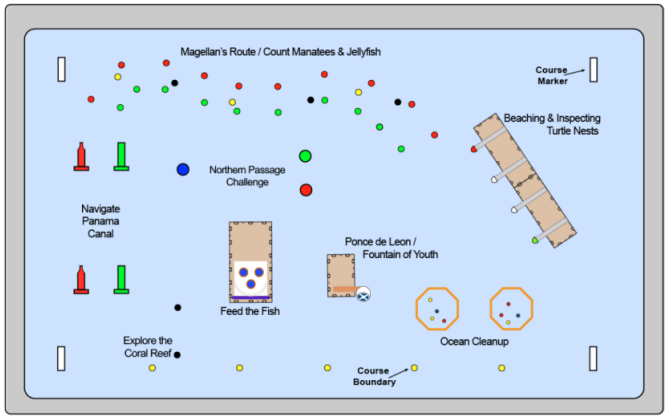


Fig. 1: Preliminary Qualifying Course for the Roboboat 2023 Competition.

When our global planner doesn't have enough information about the course to determine where any task is, we enter exploration mode. In exploration mode, we use one of our several search strategies to travel around the course and identify the positions of various course objects. Once we've identified enough objects of a course to determine its starting point, we enter the traveling phase. In the traveling phase, the task starting point is passed to our navigation system, which handles sailing there, including collision avoidance. Once we arrive at the starting point for a task, any supporting processes to complete the task are activated. These processes are also responsible for notifying the global planner when the task has been completed and it can determine a new state. To ensure we don't get stuck on one task, we have a timeout period for each task based on the maximum amount of time we would like to spend completing it. This in turn is based on how valuable we think the task is given the scoring system and our success rate in testing.

II. Design Strategy

A. Hull Design

With long-term reusability in mind, we wanted *Ship Happens*, seen in Figure 2, to serve as a testing platform for future projects. As a result, we decided to build a 5 ¾' catamaran with a bridge that spans nearly the entire area, to give more flexibility for component placement. The hulls are made of six layers of rigid foam board cut to shape, wrapped in fiberglass, and sealed together using epoxy resin. Interlaid between the foam layers are marine plywood and a carbon

fiber undercarriage to further brace the hulls. In order to make our boat even lighter, we replaced the aluminum supports between the hulls with carbon fiber. Just like last year, we then created a support bridge for our main platform out of marine plywood. The joints between the bridge and hulls consist of 3-inch steel studs screwed into small wooden planks embedded underneath the epoxy and fiberglass layers of the hulls. In total, our updated hulls and deck, shown in Figure 3, weigh 42.5 pounds and can hold a maximum load upwards of 300 pounds. This means the changes we made resulted in a reduction of 5 pounds in weight between last season and now.

Fig. 2: *Ship Happens* in the Charles River in October 2022.

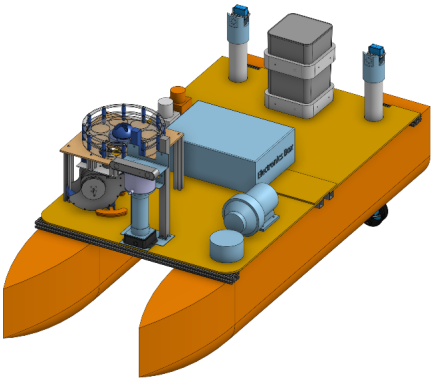
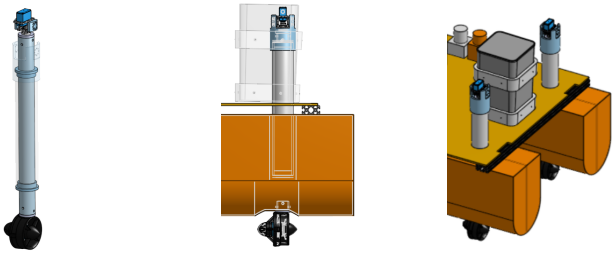


Fig. 3: CAD of *Ship Happens'* new design.

B. Propulsion System

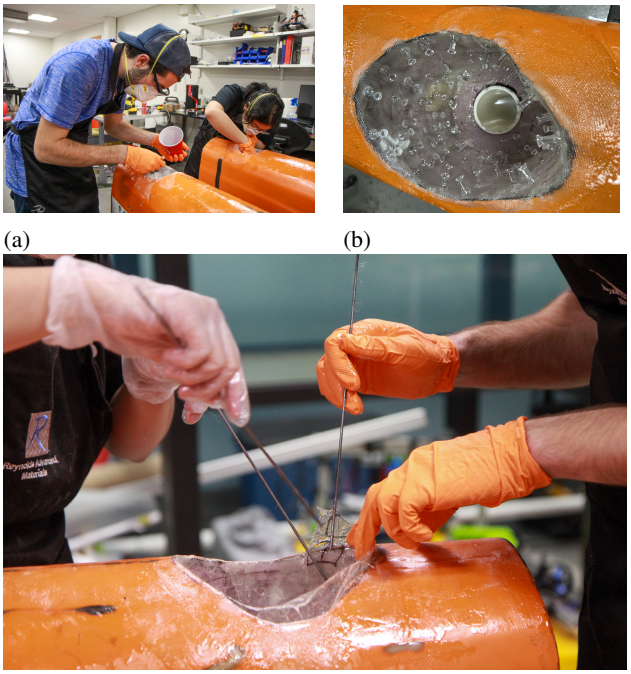
To increase our maneuverability and prevent us from drifting into obstacles, we developed an azimuth thruster pod design which allows us to rotate our Blue Robotics T200 Thrusters in place. Each T200 thruster is attached to a servo motor through a PVC pipe, allowing us to reorient the thruster relative to the hull. The system is mounted inside a larger PVC pipe to create a telescoping mechanism, allowing us to lift the thruster into the hull during transport and deploy the thrusters in the water. By mounting the thruster this way, the force is redirected into the bearings, preventing shearing of the mount. The PVC extends up through the hull and above the deck to maximize the space between the thruster and the upper bearing, thus minimizing the force on the bearings caused by the cantilever. Figure 4 depicts a CAD a section view of the thruster assembly and integration into the hulls.



(a) (b) (c)

Fig. 4: Azimuth thruster subsystem **(a)**, Cross-section of thruster assembly **(b)**, Thruster assembly integrated into the hulls **(c)**, Thruster assembly integrated into the vehicle.

To accommodate our new thruster pods inside the hulls, we drilled a vertical hole through each one. We were able to accomplish this by drill pressing with a 2-½" Forstner bit on one side and a 5" hole saw on the other. We then ensured the hulls were watertight once again by resealing them using fiberglass strips soaked in epoxy resin, as seen in Figure 5a. To ensure good contact with the walls of the hole while the epoxy dried, we used push pins through the strips for the first layer, as seen in Figure 5b. For later layers, we found it was easier to presoak the entire strip in epoxy, then use welding rods as chopsticks to move the sticky strips into place (Figure 5c).



(a) (b)

(c)

Fig. 5: The layup process we used to seal the holes we drilled through the hulls.

C. Electrical System

To best support the onboard subsystems and their ease of use and robustness, we designed the layout of our hardware to promote a smooth testing and debugging process as shown in Appendix B. We also implemented several new features which would make *Ship Happens* further adaptable to different circumstances and applications. Learning from our previous experiences, we integrated a more in-depth overcurrent protection system. We placed optocouplers at integral locations to isolate our propulsion power bus from some of our more sensitive electronics, and we streamlined much of the power distribution within our electrical system through the design of a custom-printed circuit board to eliminate convoluted wiring and conserve valuable space within our main electronics box. Our Nvidia Jetson AGX Xavier processing unit is at the heart of our system, acting as the delta for all of our sensor data. Then, our Pixhawk and Arduino Mega microcontrollers are the intermediaries between the Xavier and all of the thrusters and task-specific motors on the vessel. There are also three buck converters to allow different power supply configurations to our components in order to optimize performance. This design choice is especially important for our computer (the Jetson Xavier), which requires certain specifications for performance as well as our assortment of motors and their drivers.

We also made some changes to our physical setup to allow for easy access and designed custom heat sinks for our Blue Robotics electronics speed controls (ESCs) that help wick away heat from the board while remaining spatially optimal. To enable quick access, our entire electronics system disconnects from its water-resistant enclosure so that we can access every component for maintenance and development. We strategically designed an ideal, power-efficient electrical subsystem to ensure that as much of our battery capacity as possible went towards our external-facing systems like the mechanisms and the thrusters. As a result, we selected components like buck converters (due to their high efficiency) to step down our voltage sources and Solid State Relays to be used in place of Electro-Mechanical Relays to minimize power consumption and electromagnetic interference.

D. Auxiliary Systems: Remote Visual Signaling and Thermal Management

This year, we included a remote visual signaling system in the form of an LED Tower. The purpose of the LED Tower was to visually communicate the status of the onboard systems from afar so as to be conducive to a more efficient debugging and tracking process while the ship is in the water. It had three different colored (red, green, and yellow) lights and a single pitch speaker, which together could signal up to 16 different statuses. To conserve space, instead of using electro-mechanical relays, we used MOSFETs as switches to turn the lights and sound on and off. They were further controlled by 5V digital signals from the Arduino Mega.

Our approach to thermal management was informed by the performance of our previous vehicles. The primary source of heat on *Ship Happens* is the electronics box, so efforts were focused there. We developed a system that would allow us to supply cool air and exhaust warm air to the sealed electronic components while minimizing the risk of water reaching them. We achieved this design aim by integrating fans into a series of water-resistant piping which contains bends and relatively long stretches of tubing to prevent water droplets from reaching the air inlet or exhaust. The system consists of two different parts: the fan mounts, made up of a pair of circular acrylic mounting plates and a cast silicone gasket; and the fan casing, made up of a 3D-printed hollow cylinder meant to enclose the fan and funnel the air into a PVC pipe as seen in Figure 6. The fan was selected based on the limits of the electrical power supply and the dimensions of the electronics box. Based on those two criteria, we decided to use a pair of Pano-Mounts QH8025IP2510 80 mm fans that operate at 12 V and 0.33 A. One fan would provide air at the inlet and the other would pull air from the exhaust of the electronics box. The size of the tube used in the water-resistant piping was determined based on a set of measurements taken from the first prototype of the fan assembly found in Appendix A.

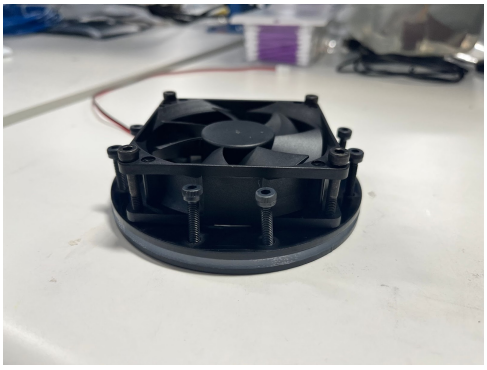


Fig. 6: The fan is placed on the acrylic mounting plates with a silicone gasket in the middle. The wall of the electronics box sits between the silicone gasket and the bottom mounting plate.

Custom heat sinks were machined from aluminum for the electric speed controllers and relay switches to allow for more flexibility in component placement within the box. All other components were left with their original heatsinks where applicable.

E. Task Approach

1) Software Platform: Our autonomy software makes use of our in-house Robot Operating System (ROS) package: AllSeaingVehicle. It's divided into several different suites that handle the various areas of autonomy, described in Table I. A full graph of the ROS nodes can be seen in Appendix C.

It's designed to be modular, hardware-agnostic, and well-documented. In particular, the pilot suite supports the ability to start and stop other ROS nodes based on a pre-planned queue or in response to starting or stopping a given task. This configuration allows us to easily remove and insert components to address tasks year to year in RoboBoat competitions. All of these attributes mean that it's already being used for other projects at MIT such as the *Oystermaran*, an autonomous oyster bag flipper out of MIT Sea Grant.

Sub-Package	Purpose
sensor_suite	Provides drivers and ROS nodes for interfacing with sensors
perception_suite	ML-powered object detection and classification
mapping_suite	Produces 2D maps and occupancy grids of global environment
pilot_suite	Provides path planners, motor driver communication, and interfaces for other common autonomy software such as MOOS-IVP
sim_suite	Gazebo-based simulation environment for ASVs and eventually AUVs

TABLE I: List of sub-packages used for AllSeaingVehicle.

Using AllSeaingVehicle, we address the competition tasks as discussed below.

2) Buoy Tasks: For Panama Canal, Magellan's Route, and Northern Passage Challenge, we rely on a simple lane-following algorithm using the RGB camera image where we consider the green buoys to form the left line of the lane and the red buoys to form the right. We take the closest buoy of each color and calculate the average of their x and y coordinates. If a red or green buoy is not found, the lines are assumed to be to the left or right side of the camera image. We then calculate the angle of the line from the bottom center of our camera image and use a PID controller to send velocity commands that adjust our heading to match the velocity. In the case of obstacles or other-colored buoys, we replace the left or right line with that buoy, creating a new lane on whichever side of the buoy leads to the widest gap.

3) Water Gun Task: Our strategy was to continuously shoot as much water as possible to increase the likelihood of hitting the target and decrease the time it takes to fill the tube. Using the distance we needed to spray and the amount of water that was necessary to fill up the tubes, we were able to calculate the necessary PSI and GPM of 15 PSI and 0.82 GPM, respectively. To improve our margin for error, we chose a pump with 70 PSI and 5.5 GPM, giving us a safety factor of over 4.5.

We wanted to be able to aim the water gun independently without moving the entire boat. We accomplished this by attaching the output of the pump to a platform that rotates in the x and y planes and adjusting the power to the pump to vary the z height. Using two waterproof servos, this design allowed us to manipulate the yaw and pitch of the end of the tubing where the water would exit.

To identify targets, we utilized the ZED-2 camera on our vehicle, which provided both RGB and depth data. RGB data can be used to detect the target: we applied a threshold for the blue component of each pixel value (in HSV specifically, which is a better format for distinguishing colors). Then, all pixels which met a certain blue "threshold" could be considered to be part of the target. Before doing this, we also used the depth data as a mask: all pixel values which were detected to be beyond a certain threshold distance away are ignored in order to minimize unintended noise due to distant objects. After the pixels that corresponded to the target were detected, we took the median of their locations to approximate the center of the target. While we previously planned to use image segmentation in order to identify the target, we judged that this approach might be difficult to execute in real-time due to the computational complexity of these algorithms.

4) "Feeding the Fish" Task: Our launching mechanism for the "feeding the fish" task consists of three main parts: a rotating feeder, a turret, and a flywheel system. The top of the launching system is a rotating plate that can store and deposit the racquetball as seen in Figure 7a. Once the holes in the rotating plate are aligned with a hole in the base of the rotating feeder, a ball drops into the flywheel depicted in Figure 7b. A flywheel compresses the racquetball against a ramp and gives it spin to launch the "fish food." The entire flywheel system is mounted to a turret shown in Figure 7c; a set of gears controlled by a servo motor can rotate the shooting mechanism to aim independently of the rest of the boat. All three systems integrate into the launching mechanism shown in Figure 7d. The algorithm used for aiming is very similar to a water gun except that we use RGB pixel detection of the orange circles around the table's holes.

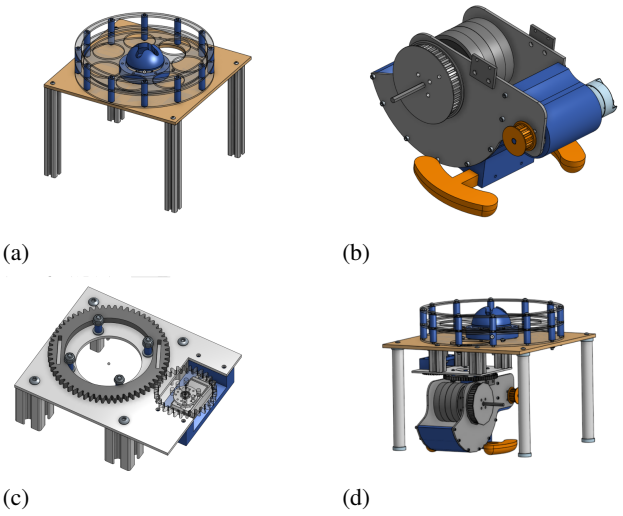


Fig. 7: Preliminary CAD design of the ball launching mechanism for the fish-feeding task. **(a)** Revolving holder and deposit system, **(b)** Flywheel system, **(c)** Servo powered turret, **(d)** Full assembly of ball launching mechanism.

5) Control Dashboard: Although it is possible to access the data from a computer's command shell, it is not always convenient. Some of the commands aren't very intuitive and visual data can't easily be displayed. As a solution, ROS offers a package called RVIZ which can display standard ROS topics including 2D and 3D data. However, RVIZ lacks the capability to easily display some of our custom messages as well as the ability to easily control actuators on the vehicle. ROS also offers RQT as a more general framework for developing GUIs; however, it does not come with as many built-in widgets as we would like. Instead, we used QT, software that expedites GUI development by providing libraries for widgets such as buttons, labels, tabs, and forms. In Python, a high-level programming language, we used PySide2 to develop a control dashboard with QT. Our dashboard, shown in Figure 8, allowed us to easily view data from our software system as well as help control the vehicle.

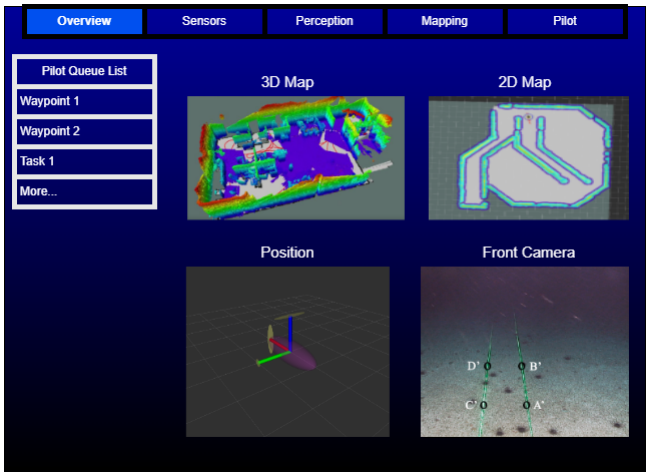


Fig. 8: A mockup of our control dashboard.

III. Testing Strategy

Our experience at the competition taught us that we needed to begin testing a lot earlier in the competition cycle. However, we also needed to balance testing with time to make hardware and software improvements to our system. To achieve a good balance, we developed two testing platforms. The first was our own simulation platform using Gazebo, an open-source 3D robotics simulator, on top of resources from RobotX and other open-source repositories on GitHub. This allowed us to simulate our sensors and actuators including the thrusters, Velodyne LiDAR, Pixhawk controller, and RGB camera view. Through SITL testing, we were able to validate our ROS nodes before deploying them on our vehicles for real-world testing. We also developed a smaller vessel named *Athena* shown in Figure 9, with a near-identical sensor and propulsion system to *Ship Happens*. This allowed us to carry out useful real-world testing while *Ship Happens* was being rebuilt. With these platforms, as well as *Ship Happens* when it is complete, we carried out a variety of system-validating tests detailed in Appendix A.

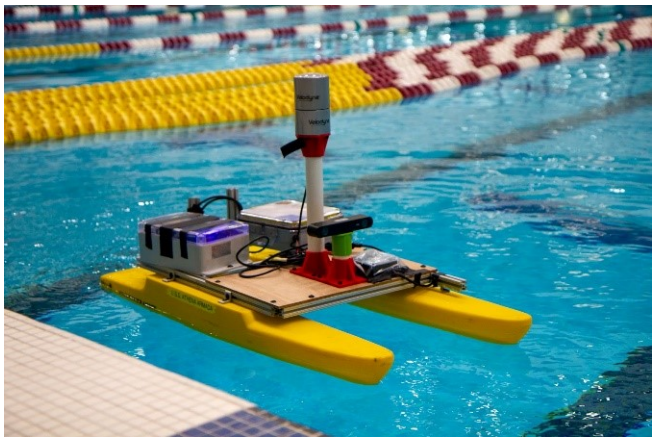


Fig. 9: Indoor testing of our small scale model vessel, *Athena*, on the water at the MIT Zesiger Athletic Center pool.

IV. Conclusion

Ultimately, we believe our design accomplishes the goals we set out at the start of the season and were conclusively proven using our testing strategy. Our ball launcher and our water gun mechanisms work and control consistently, while our new thruster design increases our maneuverability in the water. Furthermore, our electrical overhaul ensured that these mechanisms, as well as the boat itself, were streamlined and protected against voltage and current spikes. Our autonomy stack allowed us to easily implement and change our task strategies. Throughout this process, we were mindful of systems integration between our mechanical, electrical, and programming subsystems to ensure they interfaced properly. And by testing early and often, we were able to learn from our mistakes and make redesigns effectively before they became major issues. Overall, we are very proud of the progress we have made in a few short months and are excited to make a splash at competition!

Acknowledgment

We want to thank all of the sponsors whose generosity allowed us to not only build our boat, but also foster a space where students can learn and grow together.

1) MIT Departments of Mechanical Engineering, Mathematics, Aeronautics and Astronautics, and
2) Nuclear Science and Engineering
3) MIT Edgerton Center
4) RoboSys
5) ProjX
6) Coop Grant
7) Alex Soo
8) MIT Innovation Fund

We would also like to acknowledge those who gave us in-kind loans or donations of stock, testing materials, tools, or sensors.

1) Milwaukee Tool: Donations of tools
2) Toyota Research Institute: Lidar loan

None of this would be possible if we were not given space and training to access the machine shops and maker spaces around MIT.

1) MIT Sea Grant
2) MIT Architecture and Design (MAD) Lab
3) MIT Laboratory for Manufacturing and Productivity
4) N51 Edgerton Center Team Shop
5) 4-409 Edgerton Student Project Laboratory

We would like to make a special acknowledgment to our mentors whose guidance and support have been instrumental to our growth.

1) Andrew Bennett: technical and administrative support
2) Aditya Nawab: path planning feedback
3) Franz Hover: design review of thruster layup
4) Jim Bales: electrical system feedback

APPENDIX A: TEST PLAN RESULTS

A. Software Tests

1) Simulation and Software In The Loop Testing: Using a Gazebo simulation environment, built upon the RobotX simulation environment, we tested our autonomy stack. We were able to simulate our LiDAR and camera sensor data as well as wave and wind dynamics. Combining it with a gazebo Ardupilot plugin and the Ardupilot Software-In-The-Loop simulator, we could control virtual vessels and execute tasks. Using this environment, we tested various tasks including the navigation demonstration task, as shown in Figure 10.

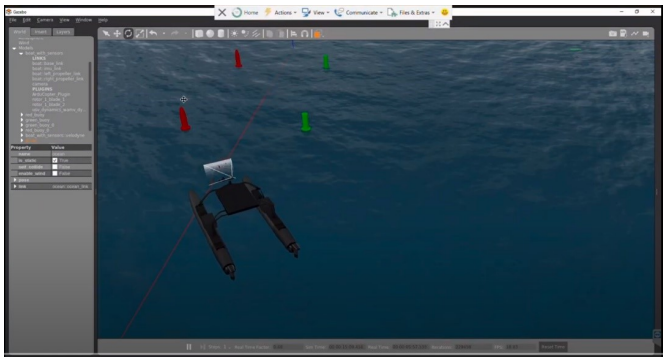


Fig. 10: Gazebo simulation of navigation demonstration task using WAMV vehicle.

2) Rosbag Testing: During the RoboBoat 2022 competition, we collected about 50 GB of sensor data, approximately 10 minutes of operation, from our test runs of the course. We used this data as another simulation source for testing our autonomy stack.

B. Hardware Tests

1) Thermal Management System: The experimental setup we used to find the optimal inlet diameter for our cooling fan system seen in Figure 11 consisted of a fan mounted on an acrylic panel that had a hole cut through the center. Around the fan we placed a 3D printed hollow cylinder, emulating the casing that surrounds the fans in the boat. The end of the cylinder away from the acrylic panel had a replaceable lid with holes of varying sizes allowing us to measure the effects of differently sized air supply tubes on the flow rate of the fan. On the opposite end of the acrylic panel a 4" PVC pipe was attached (referred to as the "outlet"), and an EA-3010U handheld anemometer was mounted on this pipe and on the inlet hole in order to measure the flow rate.

We found that the volumetric flow rate of air from the fan increased with inlet diameter until we reached an inlet diameter of 60 mm, at which point it was equivalent to a fan open to the environment at approximately 0.018 m^3/s (Figure 12). While the inclusion of the PVC elbow joint decreased the flow rate, their inclusion is necessary to prevent water from reaching the electronics.

Consequently, we opted for an inlet with a diameter of 60 mm, with the limiting factor in size being the space between the electronics box and the battery box. Including an elbow joint (L-joint) at the inlet did reduce the flow rate; however, we decided to include the joints in our final design in order to route the air supply and exhaust to other places on the boat.

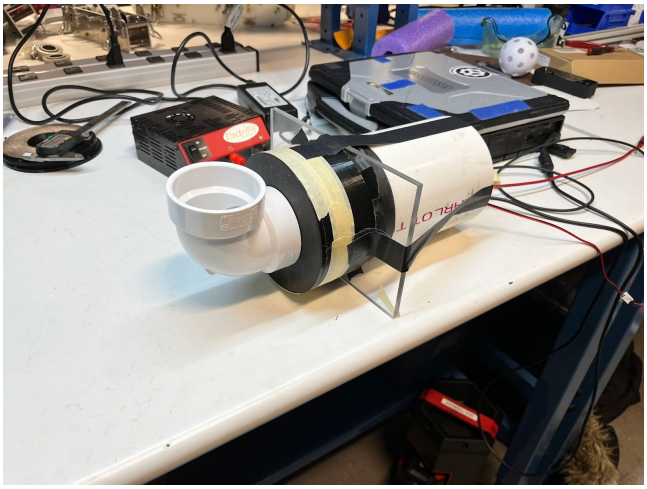


Fig. 11: The experimental setup of the fan assembly shown during testing. The inlet is located at the 90° elbow joint and based on the measured flow rate, we optimized the inlet diameter.

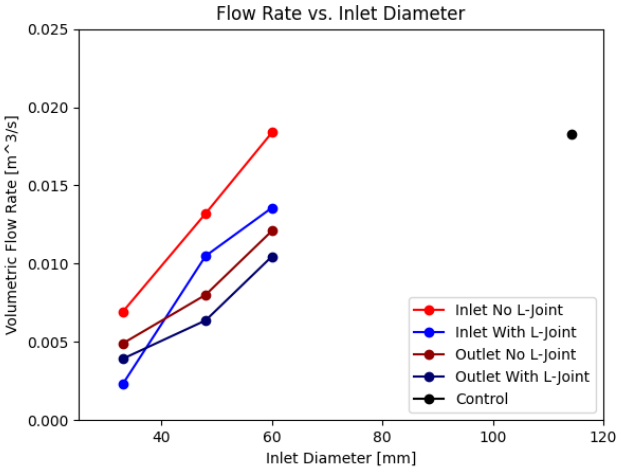


Fig. 12: Relationship between inlet diameters for varying pipe configurations and volumetric flow rate. The measurements labeled "inlet" and "outlet" were taken using different methodologies and should not be directly compared. The inlet and control measurements were taken using the same method.

2) Water Gun Testing: When testing the water gun, we wanted to identify not just how far we could shoot, but also for what part of the water's trajectory we could assume a linear path (Figure 13). If we determined we could model using a linear path, we could aim based on calculating the angle to the target point from the water gun. Through our testing of the water gun pump and nozzle we found that we had a maximum range of 24 ft. and could assume a linear path for 3 ft for any angle between 20 to 70 degrees.

Fig. 13: Experimental water gun mechanism set up at MIT Sea Grant to determine trajectory of the water output.

3) Ball Launcher Testing: For the ”Feed the Fish” task, we wanted to determine the maximum shooting distance and height as well as the consistency of the launcher. We tested four balls in a row, manually rotating the loading mechanism. Our test setup can be seen in Figure 14. After testing, we found that the ball has a parabolic trajectory and travels approximately a maximum of 5 ft horizontally and 2.5 ft vertically. We also found that it took about four seconds for the flywheel to return back to speed. If the balls were deposited too quickly after each other, the trajectory would be inconsistent. With the four-second grace period, the balls fell within a one-inch range. Based on these values, we are changing the ratio of the timing belt pulleys to increase the speed on the flywheel, which will increase the shooting distance. We are doing additional testing to determine the relationship between the speed of the wheel and the trajectory.

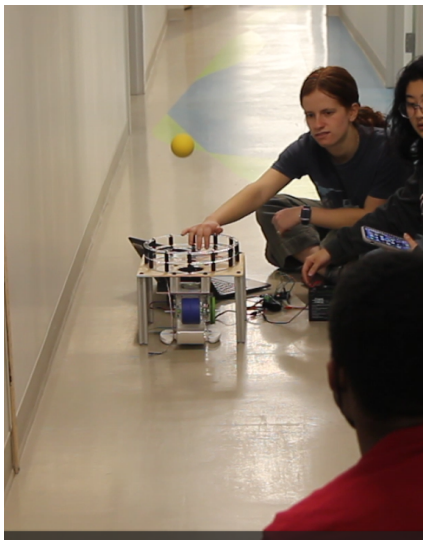
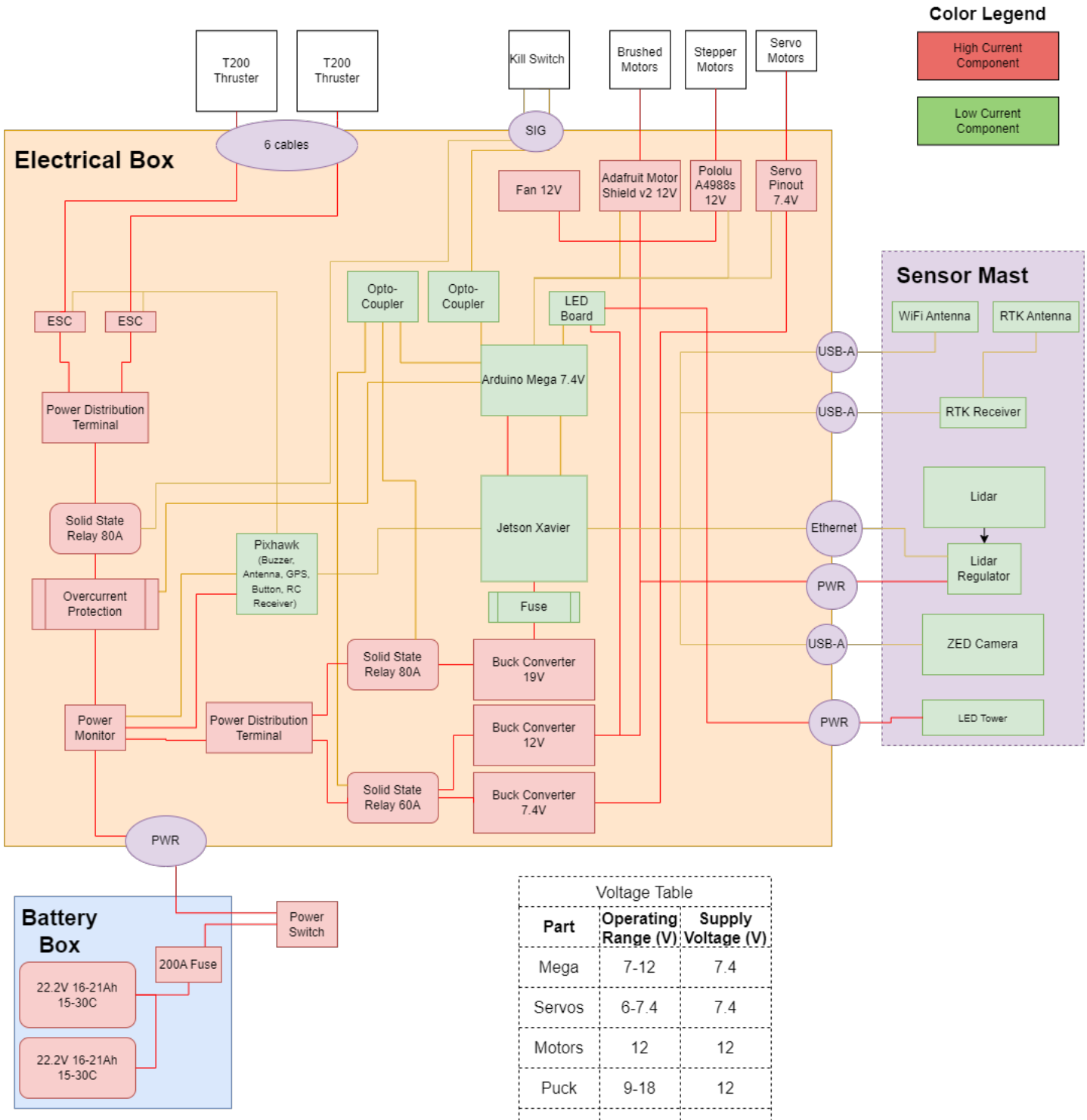


Fig. 14: Testing the ball launcher mechanism at the MIT Sea Grant Facility.

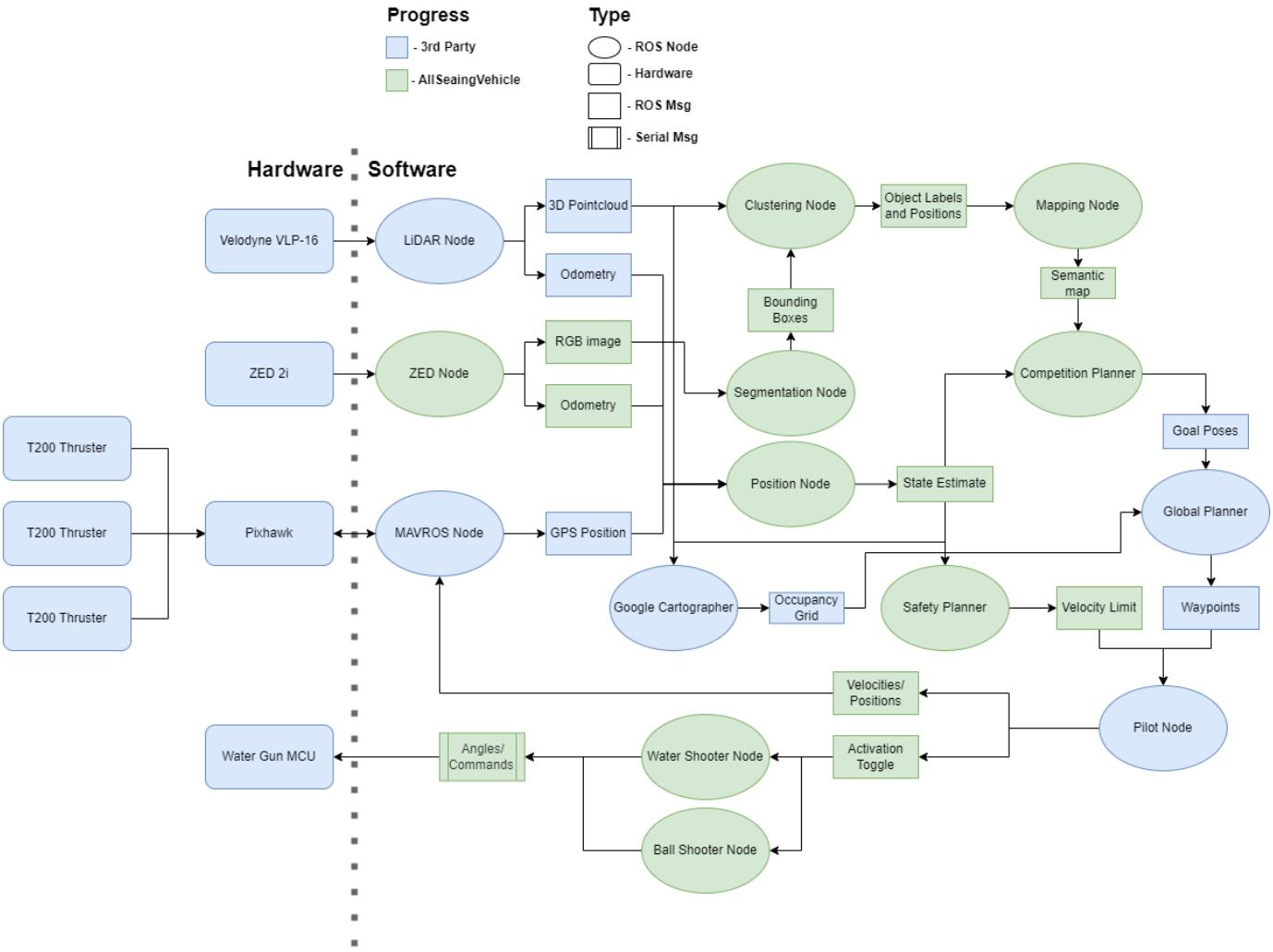
APPENDIX B: ELECTRONICS DIAGRAM



Voltage Table		
Part	**Operating Range (V)**	**Supply Voltage (V)**
Mega	7-12	7.4
Servos	6-7.4	7.4
Motors	12	12
Puck	9-18	12
Xavier	9-20	19
Stepper	12	12
LED Tower	7-24	12
Fans	12	12

Electronics diagram providing an overview of the electrical system for the entire boat.

APPENDIX C: SOFTWARE DIAGRAM



Visualization of ROS node graph during normal operations of competition vehicle.

APPENDIX D: COMPONENT SPECIFICATIONS

Components	Vendor	Model/Type	Specs	Custom/Purchased	Cost	Year of Purchase
ASV Hull Adhesive	West Marine	Two Part Epoxy Solution	West System 205-B, 105-B	Donated	$170.00	2021
ASV Hull Filling	Owens Corning	FOAMULAR Unfaced Foam Board Insulation	3-in x 4-ft x 8-ft	Donated	$45.00	2021
ASV Hull Wrap	McMaster-Carr	Fiberglass Wrap	50-in x 15-ft x 0.035-in	Donated	$60.00	2022
ASV Hull Undercarriage Reinforcement	McMaster-Carr	Carbon Fiber Wrap	Unidirectional Weave, 50-in x 36-in x 0.014-in	Donated	$67.49	2021
ASV Hull Platform	McMaster-Carr	Marine Grade Pressure-treated Plywood	1/4-in x 2-ft x 4-ft	Donated	$88.80	2022
ASV hull connection support beam	McMaster-Carr	Carbon Fiber Square tubing	0.4in wall thickness 1in x 1in 32in length	Donated	$664	2023
Propulsion	Blue Robotics	T200 Thruster and Basic ESC	Full Throttle FWD/REV Thrust @ Maximum (20 V): 6.7 / 5.05 kg f Operating Voltage: 7-20V Full Throttle Current @ Maximum (20 V): 32A Full Throttle Power @ Maximum (20 V): 645W	Borrowed	$236.00	2021
Battery (22.2V)	Pulse Ultra	LiPo Battery	15C 16000mAh 22.2V	Borrowed	$250.00	2017
Camera	Stereo Labs	ZED 2i	120 FOV, Built-in IMU barometer, magnetometer, positional tracking, spatial object detection, neural depth sensing	Purchased	$499	2021
Computer	Nvidia	Nvidia Jetson Xavier Developer Kit	21 TOPS, 32 GB Memory	Borrowed	$1900	2022
LiDAR	Velodyne	VLP-16	100m range, 360 degrees	Borrowed	$3300	2022
GPS RTK System	Emlid	Reach M2	Multi-band, Baseline up to 100 km in PPK	Donated	$599	2022
Microcontroller	Arduino	Arduino Mega 2560	54 digital input/output pins, 16 analog inputs, 4 UARTs , a 16 MHz crystal oscillator	Purchased	$48	2023
Motor controller	Pixhawk	Pixhawk 2.4.8	32-bit ARM Cortex M4 core with FPU. 168 Mhz/256 KB RAM/2 MB Flash	Donated	$190	2023