RoboBoat 2025: Technical Design Report

Autoboat, Cornell University
Ithaca, New York, United States

I. Abstract

For Roboboat 2025 the team's main focus is codebase development and validation through water testing and simulations, to improve performance in autonomous navigational tasks. An emphasis was placed on simplicity and efficiency, both for technical aspects and team timelines and milestones. Each competition task has a clear logical strategy associated with it, and each component of the software system was designed from the ground up to integrate easily to accomplish these tasks. Upgraded computer and sensor hardware has also been added to improve the accuracy and functionality of our boat. Mechanically and electrically Clifford remains largely unchanged, in order to give the software team a consistent and reliable hardware platform to work with. The few hardware changes that have been made are intended to improve ease of use, maneuverability, and robotics capabilities. This hardware design plan, combined with improved testing and simulation infrastructure, puts the team in a great position to score well in the autonomy challenge this year.

II. Competition Strategy

A. Approach

Our primary strategy for the RoboBoat 2025 competition is to perform all navigation tasks (tasks 1-4 and 6). We directed the majority of our focus towards these tasks as they make up the bulk of point awards and serve as an essential capability for any intelligent ASV.

B. System Overview

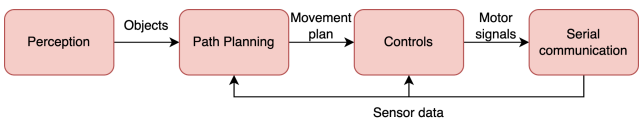


Figure 1: Overview of communication between software components. See Appendix D Figure 16 for a more comprehensive diagram of the ROS framework.

Autoboat's software system consists of four main components: Perception, Path Planning, Controls, and the Ground Station. The boat receives information about its surroundings, position, and orientation from the Perception system in conjunction with serial communication from the sensors. The Path Planning system uses this information to identify the task at hand and plan a path or sequence of movements to successfully accomplish the task. This plan is then given to the Controls system which determines how to move the motors to achieve the desired movement. The Ground Station system allows us to wirelessly control and monitor the boat. All of these systems are held together by our ROS2 Humble framework which facilitates parallelism and communication between each component. Importantly, it allows us to see (perception), think (path planning), and move (controls) simultaneously. This year we migrated from ROS to ROS2 to match our upgrade from the Jetson Xavier with Ubuntu 20.04 to the Jetson Orin with Ubuntu 22.04.

B. Task Strategies

Navigation Channel

For the Navigation Channel task, the boat identifies pairs of buoys (one red column buoy and one green column buoy) using the computer vision model. Once a pair of gate buoys are identified, the primary waypoints are selected using the midpoint mathematical calculation. If the second pair of buoys was not identified the first time, the same algorithm will repeat.

Mapping Migration Patterns (Follow the Path)

To complete the Follow the Path task, the boat identifies red and green buoy pairs forming gates, with the buoys sorted by distance. For each gate, a waypoint is created at the midpoint between the buoys. If only one buoy is visible, the boat pivots to locate the second buoy of the

pair. If it is unable to locate another buoy, the algorithm generates an offset waypoint of roughly one meter away from the buoy identified to ensure safe passage. Intermediate waypoints are injected into the path before it is sent to Controls. Throughout the path execution, Perception is continuously monitoring for obstacles (yellow buoys). If an obstacle is detected, the system recalculates a safe, collision-free path while maintaining progress toward the gate. The task completion is determined by the absence of visible gate buoys, indicating the end of the course. The system tracks the number of yellow buoys passed and performs a final rotation maneuver equal to this count.

Treacherous Waters (Docking)

To complete the Docking task, the boat pivots to identify the sign of the day. If the target sign is not in view, the boat will navigate to the other side of the dock by following a half-circular path with a radius equal to the vertical length of the dock. This will ensure that the boat does not hit the physical dock structure as it transitions to the other side of the dock. At the other side of the dock, the boat again pivots to identify the correct sign. Once the target sign is identified, the boat will navigate into the target dock by plotting a waypoint directly inside the dock of the target sign. The Controls algorithms then move the boat to the desired location.

Race Against Pollution (Speed Challenge)

For the Speed Challenge task, the boat first detects the red-green buoy pair, calculates its midpoint, and navigates to that waypoint using Controls algorithms. The boat then moves forward until it detects the blue buoy and then loops around it while avoiding black buoys by shifting waypoints either left or right of the black buoy. The black buoys (oil spills) are counted along the way. The boat then returns to the midpoint of the red and green buoy pair and spins to report the amount of oil spills detected.

III. Software Design Strategy

Perception

We use an object detection model to identify competition buoys and signs for autonomous tasks. Our model is trained on over 10,000 images of the competition environment using the You Only Look Once (YOLO) neural network. We also worked with the default YOLOv8L weights and trained for 400 total epochs. Taking a step back from newer models, we worked on testing each version number and model size that YOLO can offer. We tested each using 25 epochs and a batch of 3000 images. By comparing and understanding confusion matrices from each training, we concluded that YOLOv8L offered the most accurate readings of the different objects.

The computer vision model is capable of recognizing 19 different object classes that could show up during competition runs, such as “buoy-green” or “triangle-red.” One issue we faced was not having enough data for the variety of sign shape and color combinations. To target this problem, we decided to take a unique approach of altering existing images from last year’s competition to increase the number of images associated with each variation of shapes and colors in model training.

We continued to use the persistent memory algorithm designed in the last competition cycle, which uses the information from the previous frame to correct model predictions in the current frame. This additional step allows our system to be resistant to frames where the model fails to identify an object that was there before - a problem that was encountered in testing.

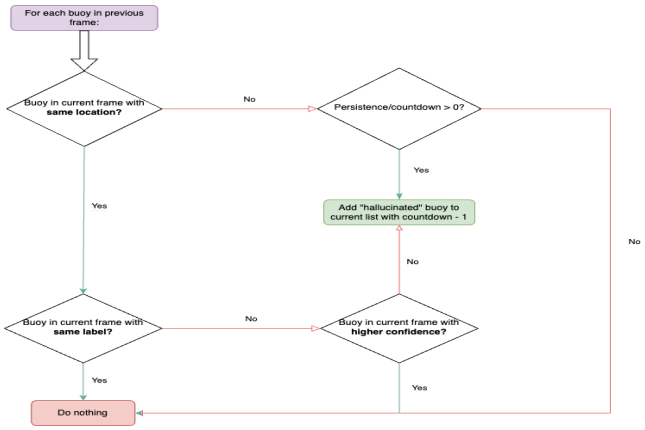


Figure 2: Persistent Memory Algorithm Flow

Path Planning

We use a similar approach for accomplishing all navigational tasks. First, the boat explores the

environment until it gains sufficient information regarding the task at hand. Then, it uses the information about its current state (GPS location, VN-300 heading) and environment (list of objects detected and their locations relative to the boat) to plan a path of GPS waypoints. After primary waypoints are selected we inject intermediate waypoints every meter or half meter along the path and smooth any harsh angles to produce more natural paths of movement.

Controls

To follow the waypoint path outlined by the path planning system, the control system employs a combination of Pure Pursuit and PD control. Pure Pursuit is a path tracking algorithm which maintains a "lookahead" point on the path some set distance away from the boat. As the boat moves, the point advances along the path, so the boat is always chasing it. Our implementation is inspired by the Purdue SIGBot's controller [5].

Pure Pursuit typically outputs an ideal linear and angular velocity for the vehicle. However, we found that controlling on velocity produces rough and shaky movements due to the sensitivity of our IMU, a sensor in the ZED 2i camera which produces velocity readings. This year we decided to transition to using heading as the variable we monitor since heading readings from the VN-300 are more reliable. Learn more about this method in [1]. The algorithm calculates the error in heading as the difference between the boat's current heading and its heading if it were pointed directly at the lookahead point. We then apply PD control to reduce this error. PD (proportional, derivative) control is a tunable equation which takes the heading error as input, multiplies the error and derivative of error by some constants, and outputs the offset which should be applied to the PWM signals sent to move the thrusters. This year we switched from a PID controller to a PD controller as we found that integral windup caused instability in the controls A larger error produces a greater offset causing the boat to turn faster and exhibit course correction.

Ground Station

Last year, we created our ground station by setting up a wireless network using Ubiquiti bullets, a router, and POE injectors to facilitate a connection between our onboard computer, the Nvidia Jetson Xavier, and onshore laptops. This connection allows for 3 essential features: SSH connection into the Jetson, transmitting GNSS correction source data, and receiving state information to be used as input to our monitoring and visualization platform. To facilitate the latter two features we dedicate nodes in the ROS framework for launching WebSocket connections to enable communication with programs run on onshore laptops. We have continued to use this same Ground Station set up but now with our Jetson Orin, which we have configured to work with our existing communication system.

We found that our visualization platform was fairly minimal, and periodically error-prone. This year we worked on entirely revamping our user interface. Last year, we showed a live local and global view of our boat and objects we detected. We have now added a dashboard for different metrics and performance graphs (i.e. velocity vs time) as well as both a live view and a past view, allowing for us to look back and analyze past data. These added functionalities have allowed us to better understand how the boat is processing data, saving valuable time during testing and competition.

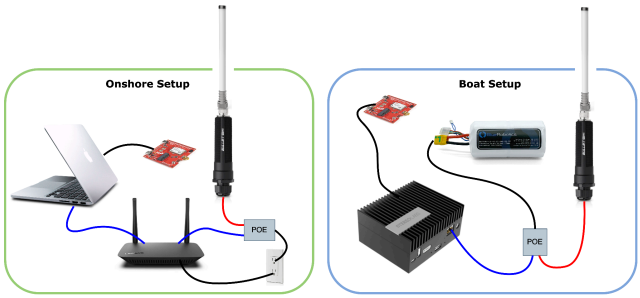


Figure 3: Communication system setup. Blue wires are LAN, red wires are POE, and black wires are power.

IV. Hardware Design Strategy

Over the past two years at Roboboat, our fiberglass-based trimaran design has proven to be reliably strong and stable - especially after the significant improvements that were introduced with Clifford. Therefore, it made sense to keep

Clifford as our primary boat, allowing the software team to focus on frequent water testing and codebase improvements with a consistent and reliable hardware platform. The hardware team mainly focused on small but impactful upgrades that would improve the boat's ease of use, and support the new sensors and robotics capabilities.

A. Mechanical System

Exterior Redesign

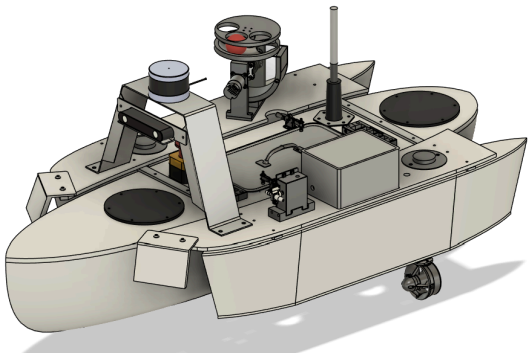


Figure 4: Top level boat assembly

The first change made to the boat's exterior was to reduce the width from 32.5in to 28in, making it easier to navigate between buoys without hitting them or getting them lodged between the main and side hulls - a major issue at competition last year. After hydrostatic simulation, 28in was chosen as the new width - retaining enough stability while giving the boat almost half a foot of extra space. To help with this, the mechanical team also designed blockers that are mounted between the hulls to deflect buoys and prevent the boat from getting stuck.

New bridge decks were designed to maximize usable space on top of the boat. This was done by extending the deck over the side hulls, making the entire width of the boat usable for component mounting. Additionally, a new camera mast was designed and optimized to withstand full-speed collisions.

A redesigned main hatch was also implemented. It has a larger footprint, allowing for easier access to the central electrical bay. It also has a custom designed sealing interface that is elevated above the deck level, ensuring that water doesn't pool by the opening and seep into the hull, another issue that we encountered last year at competition.

Electrical Bay Redesign

To save space, decrease clutter, and improve wire pathing, a modular electrical bay design was implemented. It features vertical PCB mounting on removable acrylic sheets, saving space and making it easier to troubleshoot each board individually.

The e-bay is split into 3 distinct sections. The components in each section were chosen based on their interactions with each other. The more wired connections there are between two components, the closer they should be to each other. This further reduces wiring complexity and clutter, as it makes most of the wires in the system short and easy to see. Furthermore, each section is removable, making it easier to troubleshoot electrical issues outside of the confining hull interior.

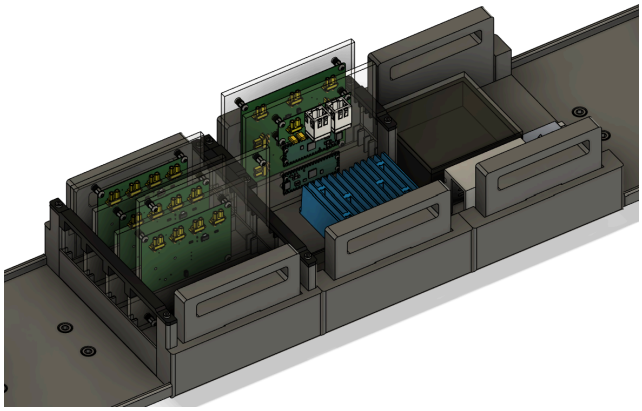


Figure 5: Electrical bay assembly

B. Electrical System

Power System

Electrical power on the boat is split between two isolated systems: a high power system and a low power system. The high power electronics are powered by a 20V battery capable of supplying peak currents near 200A - more than enough to support both motors at full power, the skeeball shooter, and the water gun all at once. The low power electronics are powered by a Blue Robotics 14.8V lithium-ion battery. The isolation between the two power systems makes it easier to troubleshoot issues, and protects the

more sensitive low power signals from noise or interference coming from the higher power lines.

Our high-power system runs through our custom kill-switch board, allowing us to manually cut power to all thrusters, motors, and robotics systems.

The low-power electronics first pass through a custom power-distribution board, which splits our battery into several outputs and steps down outputs to appropriate voltage levels for components operating at 5V and 12V.

Sensor and Computer Hardware

The boat's main computer is a Stereolabs Nvidia Jetson Orin NX, an upgrade from our Jetson Xavier last year. We use a ZED 2i stereo camera for object detection and depth sensing. A goal this year was to improve the accuracy of our localization, which was previously informed by a compass and GPS system using RTK. Experimentation showed that this system did not provide consistent, accurate results so we opted to integrate the RTK/GPS system with the VectorNav VN-300 INS system.

The VN-300 inertial navigation system also provides heading information. The GPS module is powered by a ZED-F9P Sparkfun Micromod GNSS carrier board coupled with an ESP32 processor. GPS information is sent to the VN-300, where it is inputted into built-in and custom filtering methods to stably determine the boat's position. The GPS and VN-300 communicate with the Xavier constantly, providing it with localization data. The GNSS boards were set up following [3], [4].

PCB Design

Power distribution, the kill switch, and POE injection are handled by compact PCBs. This year, 4 distinct boards were designed as opposed to a single motherboard like last year. Each board performs one of the specific tasks listed above. This makes the system more modular, making it easier to troubleshoot and replace any damaged components quickly. In addition, we have standardized connectors as much as possible, opting to use board-mounted XT-30 connectors for all wire-to-board connections except low voltage signals which use smaller wires. This makes ordering components more straightforward and reduces assembly time.

V. Testing Strategy

Our testing efforts this year have been split between land, simulation, and water testing.

A. Land & Simulation Testing

Mechanical Testing

Any component that was expected to be under significant load in the event of a crash was simulated in ANSYS Structural to determine the FOS. The simulations were verified with mesh independence studies as well as stress convergence tests. The stability of the boat was simulated in Orca. The new hatch design, as well as all of the waterproofing on the boat, was tested thoroughly by spraying water over the boat and checking for leaks before electrical integration.

Electrical Testing

Thorough electrical testing was essential to confirm the functionality of each individual PCB system and the functionality of the system as a whole. Verifying PCBs starts with a design that allows for ease of access of all essential traces. As our primary concern is voltage levels over a duration of time, each of our PCBs has test points for each intermediate or output value. Using these test points, each PCB was verified in isolation before being connected to the rest of the system, to protect our components from potential voltage/current spikes that could occur due to an unexpected issue. Before placing all of the PCBs and electrical components in the boat, the entire system was assembled and tested on a bench. This allows us to catch errors early, and troubleshoot these errors in a comfortable environment, as opposed to the space-limited hull interior.

Software Testing

We primarily tested our computer vision model through integration testing on our Jetson Orin. After training the model, we deployed it onto the boat's computer, staged buoys around the boat's environment, and used a combination of command line output and visualization output to smoke-test our model.

For testing specifically, we utilized our existing model of 16,000 images which contained images in an array of environments. We worked on creating and expanding a dataset that contains only the various testing environments we test in.

Given constraints on water testing due to weather and a lack of access to pools, we also built a Unity-based simulation framework to test path-planning code. The simulation connects to ROS and runs the path planning node assuming perfect controls and perception. It then displays the boat traveling the path in real time. The additional drag and drop map builder allows different testing scenarios to easily be created and shared.

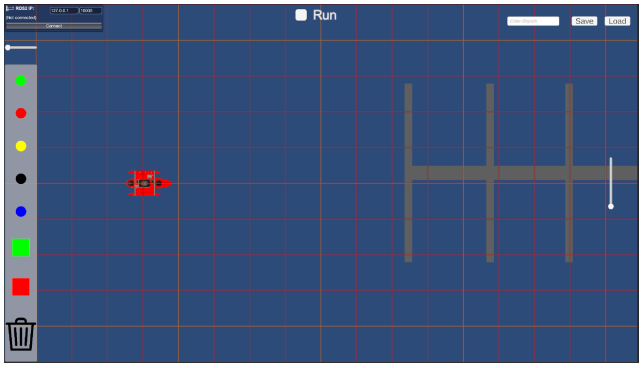


Figure 6: Simulations map builder interface.

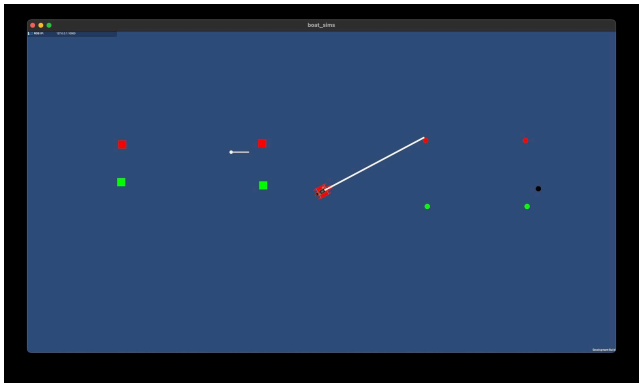


Figure 7: Unity simulation of an autonomous boat in action running a "hit buoy"algorithm. Colored squares represent column buoys, colored circles represent regular buoys, and the white line represents a planned path. See Appendix D Figure 17 for a full system diagram of our simulations integrated with ROS.

B. Water Testing

Our water testing includes outdoor testing at a private pond (provided by Cornell University) and indoor testing at local pool facilities. This testing consists of four stages: 1) safety testing to replicate the checks done at the onset of competition, 2) remote controlled buoyancy, speed, and stability testing to examine the maneuverability of the hulls, 3) autonomous control algorithm tuning (e.g. PD gains, pure pursuit lookahead distance), and 4) autonomous navigation testing. For indoor testing, a MarvelMind Starter Set Super-MP-3D ultrasonic positioning system was used to generate positional data for testing. To test autonomous navigation we incrementally build up the complexity of tasks, starting by attempting to navigate to a predetermined GPS waypoint, then navigating to a waypoint determined by the vision system, and finally autonomously planning a series of waypoints to accomplish a competition task.

VI. Acknowledgements

We extend our appreciation to everyone who lent their support to our team and helped enable us to innovate and grow as engineers. We thank the Cornell University College of Engineering for helping fund our team's travel and material expenses and providing access to facilities for boat construction and design. Thank you to our corporate sponsors: Saronic Technologies, ASML, Vectornav, and Orca3D. Our sincere thanks also go out to our individual supporters, including friends, family, and alumni of the AutoBoat team, whose generous donations have greatly contributed to our endeavors. The guidance and support provided by Dr. Hunter Adams, our faculty advisor, has also been instrumental in propelling our team towards success. We would also like to give a special thanks to the members of UM::Autonomy, RoboBoat staff including Nicholas Cassasa, Carloyn Judge, and Tanvir Reza, and all our fellow RoboBoat teams for their advice and support. Additionally, thank you to the folks at Hotel Ithaca and the Ithaca YMCA for allowing us to use their pool facilities. Lastly, we acknowledge the efforts and dedication of former AutoBoat team members whose significant contributions paved the way for our current achievements.

VII. References

[1] A. Sears-Collins, “PID Control Made Simple”. October, 2022. Automatic Addison. https://automaticaddison.com/pid-control-made-simple/

[2] N. Sariff and N. Buniyamin, "An Overview of Autonomous Mobile Robot Path Planning Algorithms," 2006 4th Student Conference on Research and Development, Shah Alam, Malaysia, 2006, pp. 183-188, doi: 10.1109/SCORED.2006.4339335.

[3] N. Seidle, “Setting up a Rover Base RTK System.” SparkFun Learn. https://learn.sparkfun.com/tutorials/setting-up-a-rover-base-rtk-system

[4] N. Seidle, “How to Build a DIY GNSS Reference Station.” SparkFun Learn. https://learn.sparkfun.com/tutorials/how-to-build-a-diy-gnss-reference-station

[5] S, Xiang, “Basic Pure Pursuit.” Last Modified May, 2023. Purdue SIGBots Wiki. https://wiki.purduesigbots.com/software/control-algorithms/basic-pure-pursuit

VIII. Appendix A: Component List

Component Name	Vendor	Model/Type	Specs	Custom / Purchased	Cost	Year of Purchase
ASV Hull Form/Platform	Self Developed	N/A	60in x 30in x 28in	Custom	$3000	2023
Propulsion	Blue Robotics	T200	Up to 5 kg Thrust / Each link	Purchased	$400	2023
Low Power Battery	Blue Robotics	Lithium-ion Battery 14.8V, 15.6Ah	14.8V, 15.6Ah Max draw 60A Max Burst 132A link	Purchased	$660	2023
High Power Battery	GoBilda	20V, 6Ah Li-ion Battery with LED Capacity Meter	20V, 6Ah, Max draw 200A, Max burst ~300A	Purchased	$40	2024
Motor Controls	Blue Robotics	Bidirectional ESC	30 A max 7-26 V Runs on BLHeli_S link	Purchased	$72	2023
Onboard Computer	Stereolabs	Jetson Zed Box Orin NX	Developer Kit: Jetpack 6.0 with CUDA 12.2 GPU: 1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores CPU: 8-core Arm® Cortex®-A78AE v8.2 64-bit Memory: 16 GB 128-bit LPDDR5 DRAM	Purchased	$1299	2024
Remote Controller	Radio master	TX12 Mark II radio controller	ELRS, Mode 1, FCC, 16 channels, 2.4 GHz	Purchased	$100	2024
Receiver	Radio master	ER8 receiver	ELRS, 8 PWM channels, 2.4 GHz	Purchased	$35	2024
Radios	Ubiquiti	UISP airMAX Bullet AC Dual-Band IP67	PtMP, PtP BaseStation radio, 5 GHz frequency band, weatherproof IP67 rate link	Purchased	$260	2023
Radio Antennae	Elecbee	Omni-Direction 2.4G Wifi Fiberglass Antenna	2.4GHz WIFI Antenna, Male, straight, standard, Threaded, 5dbi link	Purchased	$38	2023
GPS	Sparkfun	MicroMod GNSS ZED-F9P	25z Navigation Rate 0.01m Horizontal Positional Accuracy with RTK 2x USB-C Connectors link	Purchased	$325	2023

GPS Processor	Sparkfun	MicroMod ESP32 Processor	Dual-core Tensilica LX6 microprocessor 240MHz clock 520kB internal SRAM Integraterd Transceiver link	Purchased	$16.95	2023
GPS Antenna	Sparkfun	MagmaX2 - AA.200	Magnetic Mount IP67 Covers GPS L1/L2 and more link	Purchased	$83.50	2023
Camera & IMU	Stereolabs	ZED 2i Stereo Camera	120 FOV, built-in IMU, Barometer, Magnetometer, depth sensing, positional tracking, object detection, IP66-rated enclosure link	Purchased	$499	2021
Microcontroller	Atmel	ATMEGA328P-PU	Core: AVR Program Memory Size: 32kB Data RAM Size: 2 kB Package / Case: PDIP-28 Max Clock Frequency: 20 MHz Supply Voltage - Min: 1.8 V Supply Voltage - Max: 5.5 V	Purchased	$2.89	2023
Circular Waterproof Boat Hatch	Pactrade Marine	Circular Hatch	- 6” Diameter - Foam Gasket link	Purchased	$32	2023
Propulsion Mounts	Self Developed	3D printed Brackets	- Modular Interface	Custom	$50	2023
Split Multi Cord Grips	McMaster	Surface-mount 10 inserts	- Modular interface with link	Purchased	$80	2023
Algorithms (Motion Controllers)	N/A	Pure Pursuit algorithm, PID controller	N/A	Custom	N/A	N/A
Vision	N/A	YOLOv8L	N/A	Custom	N/A	N/A
Localization and Mapping	N/A	ZED 2i Stereo Camera, custom sensor fusion	N/A	Custom	N/A	N/A
Autonomy	N/A	Specialized waypoint selection	N/A	Custom	N/A	N/A
Programming Languages	N/A	Python 3, Arduino/C++	N/A	Custom	N/A	N/A
Programming Packages and Open Source	N/A	ROS2 Humble, Numpy, PyTorch, Websockets, ZED	N/A	Custom	N/A	N/A

Software		SDK, Docker, GZweb, Google Collaboratory, Roboflow				
Simulation Software	N/A	ANSYS, Altium, Rhino3D, Orca3D, Unity	N/A	Custom	N/A	N/A

IX. Appendix B: Pre-Analysis and Simulation

Hydrostatics

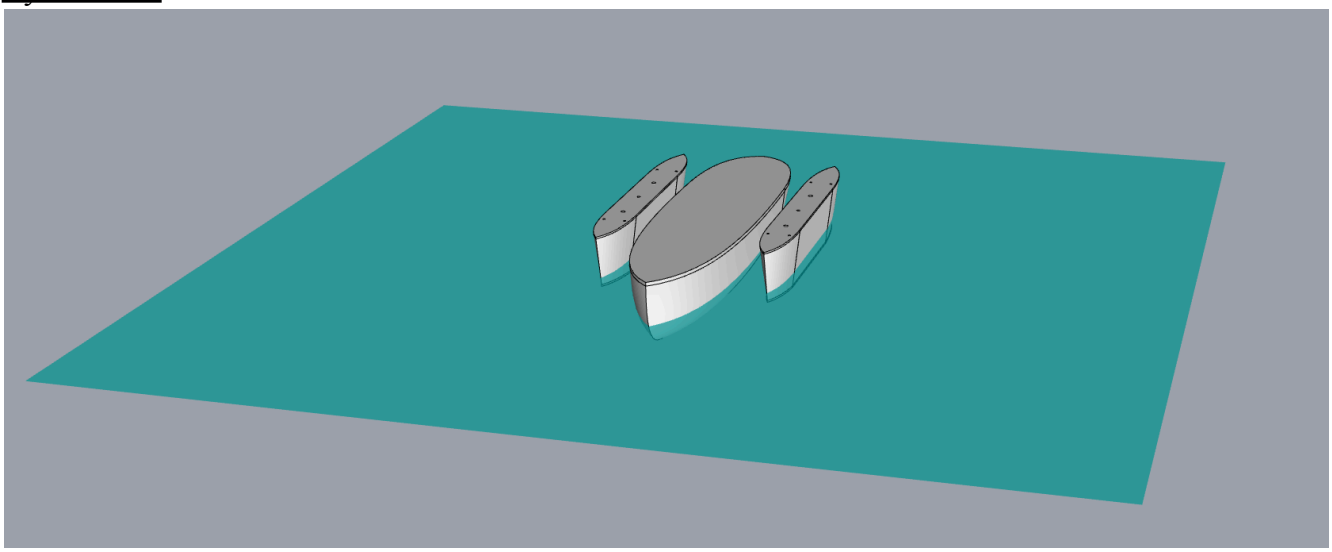


Figure 8: Render of Clifford with simulated waterline

For hydrostatics, a mass budget for the entire boat was completed. Components were then put into a Weight and Cost table in Orca, and the total center of mass was calculated. The final trim is -0.1 degrees and the heel is 0 degrees.

Stability

The width of the amas was decreased this year to allow for more error in avoiding the buoys. This, combined with a higher center of mass (from the camera mast and skeeball), results in overall much worse stability than in previous years. However, with a peak righting arm of 5in and a wide range of stability, the boat is still stable - about as stable as our previous boat George from Roboboat 2023.

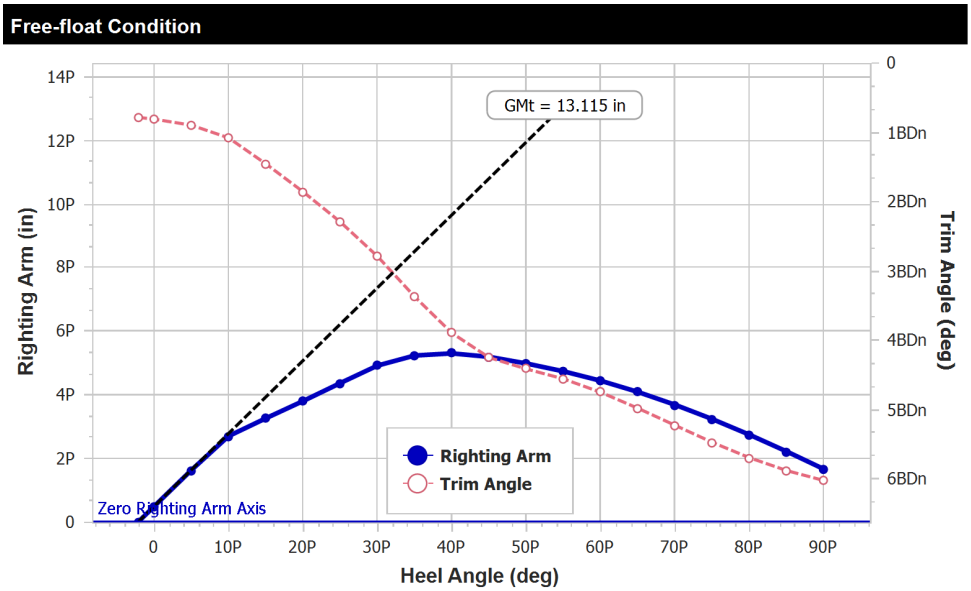


Figure 9: Updated stability curve

Hull Design Motivation

The main hull is designed to maximize the length of the waterline (LWL) to help maximize the hull speed and minimize the Froude number. The hull speed does not dictate the absolute maximum speed the boat is able to travel at, but rather the speed at which the boat will start to ride its own bow wave. This speed should not be exceeded to help minimize trim angle to ensure the camera stays as level as possible while navigating autonomously. With an overall length of 60in and a LWL of 57in, the LWL is 95% of the overall length. This corresponds to the main hull acting as a displacement hull at operating speeds, as shown by a Froude number of $Fn = 0.39$ which is just within the boundary of a displacement hull ($Fn < 0.4$). As this Froude number was calculated using the hull speed, it is important to consider the expected speed during autonomous runs to be around $u_{hull} = 0.5m/s$. The Froude number at this speed is $Fn = 0.13$, which shows the design goal was achieved and the main hull will act as a displacement hull to maximize stability.

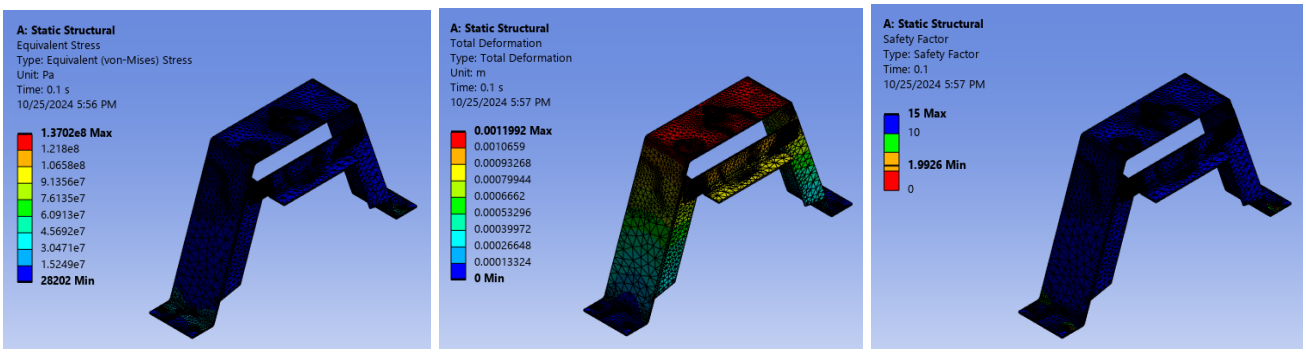
Hull speed: $u_{hull} = \sqrt{\frac{LWL \times g}{2\pi}} = 1.48m/s$

Froude number: $Fn = \frac{u_{hull}}{\sqrt{g \times LWL}} = 0.39$

Finite element analysis for loaded components

CV Mast:

The computer vision mast was simulated in Ansys static structural with a stepped, time-varying tabular shock load based on the maximum acceleration that the boat would experience during a full speed collision (determined to be 98.1 m/s^2). The minimum factor of safety was 2, which is acceptable.



Figures 10a, 10b, 10c: FEA analysis for CV mast

Ama beams:

Last year's ama beams were unnecessarily strong for the forces they experienced. In this year's redesign, smaller cross-section beams were chosen to save space. They were simulated in Ansys similarly to the CV mast. Stress converged to 15 MPa, yielding a FOS of roughly 16.

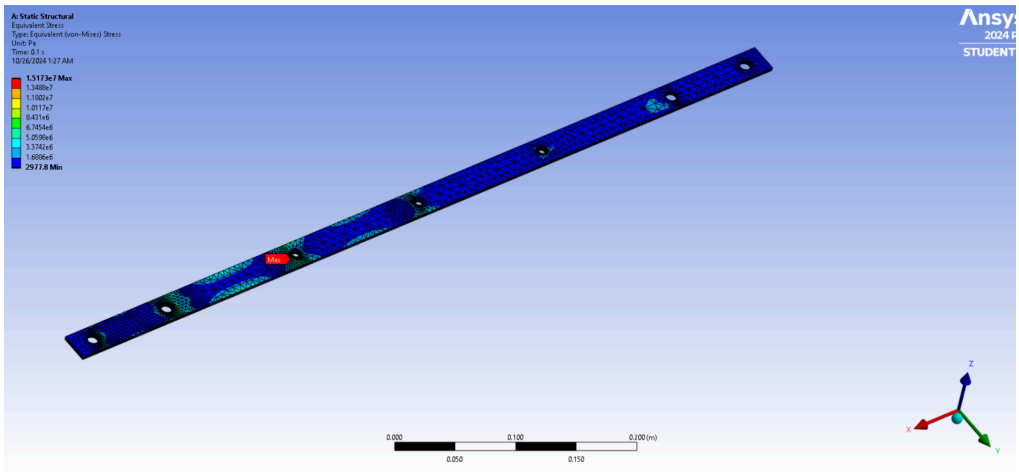


Figure 11: FEA simulation for the ama beams

Bridge deck

The bridge decks were simulated to verify that they don't deflect too much under load from the components on top of the boat. Point masses of each component were applied to the deck model with zero displacement boundary conditions applied at the inside edges of the ama beams. There was minimal deflection under the maximum loading condition.

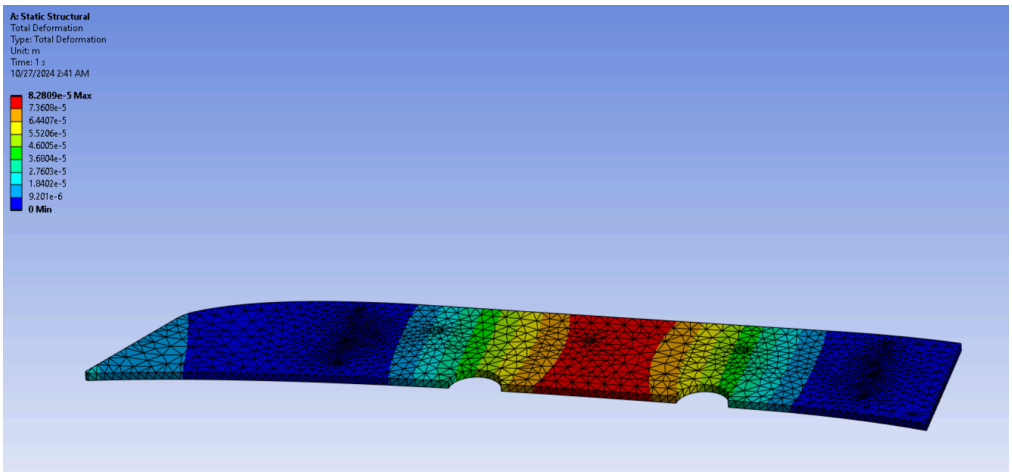


Figure 12: FEA simulation for the bridge decks

New main hatch

The main hatch was simulated to ensure that it doesn't fail under loading from the clamps that secure the cover down on top of the waterproofing gasket. The force required to adequately compress the gasket was determined based on o-ring material loading tables. This force was equally split between each gasket and applied in Ansys as a distributed force over a surface area equivalent to the surface area of each clamp.

The first design used 18 gauge 304 stainless steel as the cover material, and 4 clamps compressing it. The deformation is small - a fraction of a millimeter - but the stress converges to 202 MPa, which is pretty much exactly the same as the yield strength of steel. This gives us a factor of safety around 1, which is unacceptable.

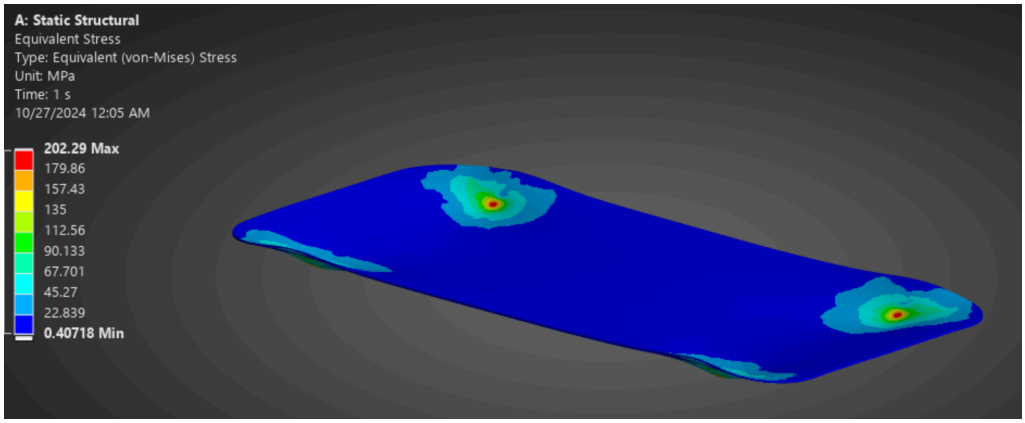


Figure 13a: First FEA simulation run for the main hatch

The second iteration increased the sheet metal gauge to 16, and increased the clamp count to 8, distributing the force more. The deformation remained almost the same, but the stress converged to around 60 MPa, yielding a FOS of around 3.4 which is acceptable.

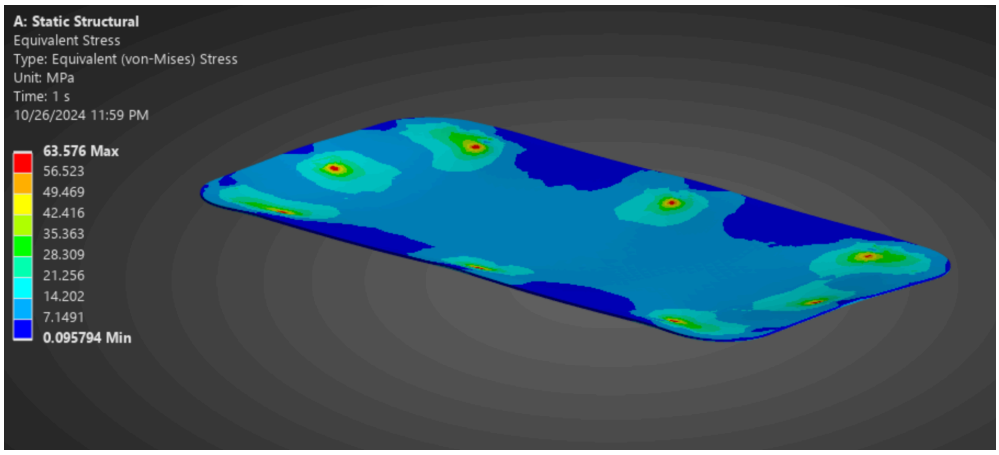


Figure 13b: Second FEA simulation run for the main hatch

X. Appendix C: Manufacturing Methodology

Center Hull, Frame & Deck

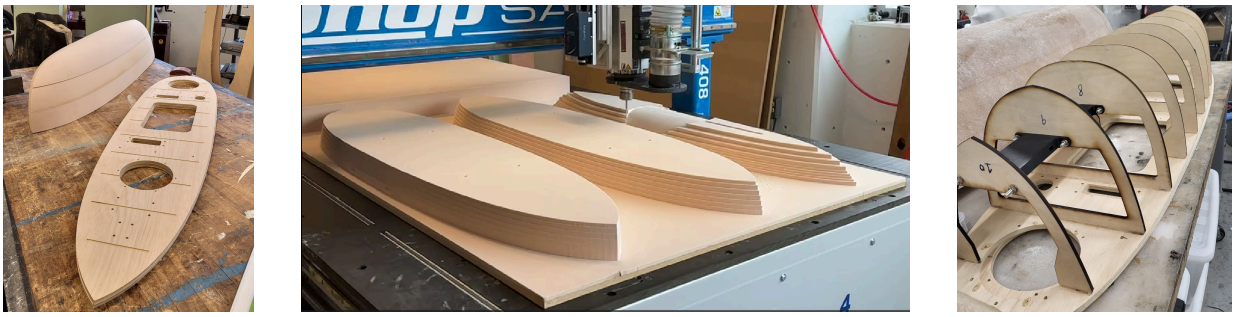


Figure 14: CNC foam mold and wooden deck (L), CNC mold fabrication process (C). Laser cut wood frame (R)

Initially, a male mold of the hull surface was created by assembling CNC cut sections of tooling board. Following the mold's completion, the hull was crafted using a hand layup of fiberglass mat and polyester resin. After the resin cured, the hull was separated from the mold. The structural elements of our frame were laser-cut from Baltic birch wood, assembled inside the fiberglass hull, and covered with our CNC cut wood deck. The junction where the deck met the fiberglass was sealed with fiberglass mat and resin. Epoxy fairing compound was applied to address major irregularities in the geometry. The entire structure was sanded for a smooth surface finish then topped with a barrier coat and paint to protect against the elements while boasting a flashy red appearance.

Figure 15: Fairing Compound Coat (L), Barrier Coat (C), Red Paint (R)

The amas were manufactured similarly to the main hull, with the distinction that the foam core was hand-cut to shape from low-density polystyrene. Additionally, the foam core was not removed from the composite part to enhance structural strength.

XI. Appendix D: Additional Figures

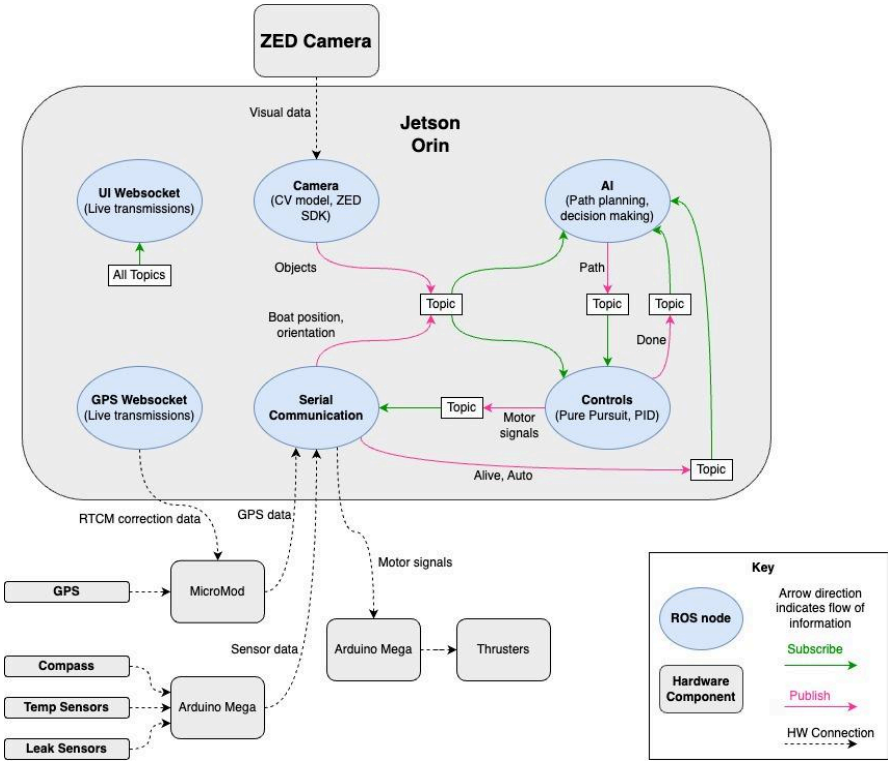


Figure 16: Publisher/subscriber node structure of the ROS framework, showing intercommunication between hardware and software systems, including sensors.

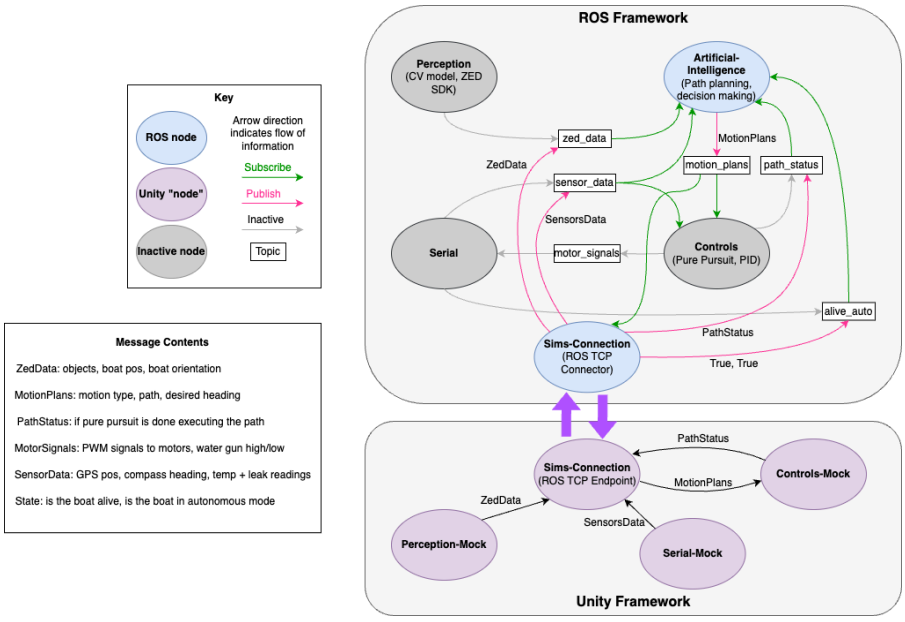


Figure 17: ROS framework contextualized by Unity simulations framework, with mocked message data sent across TCP.

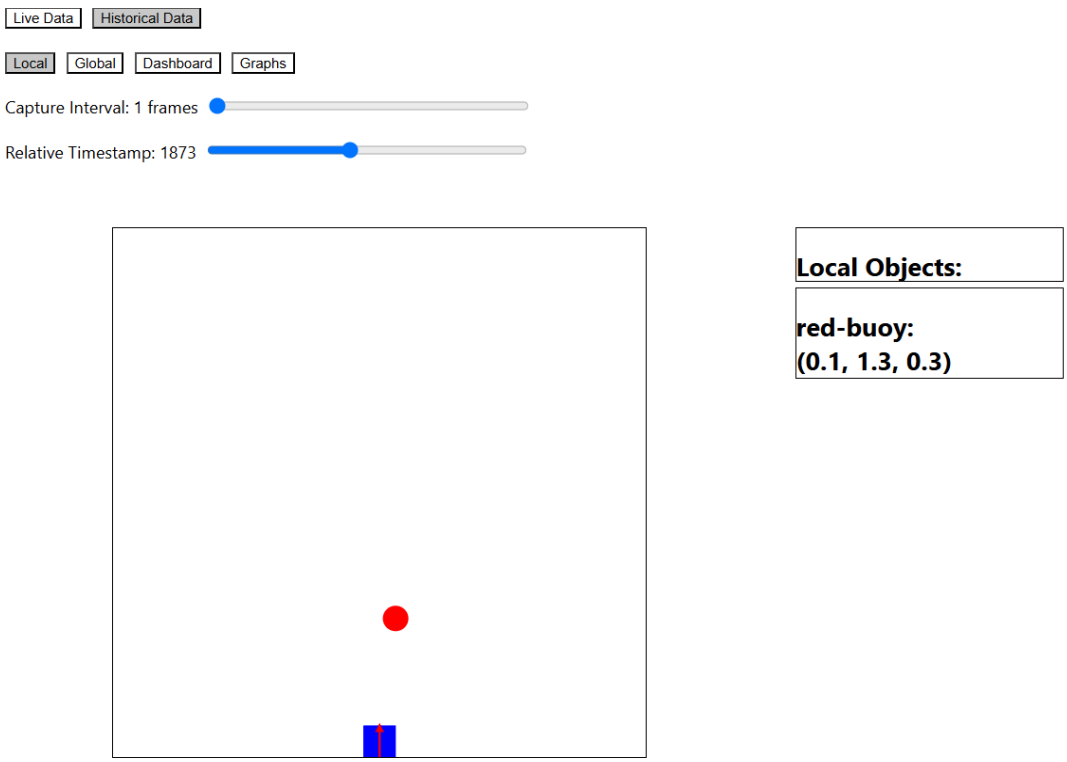


Figure 18: Our UI local view when in "historical mode"

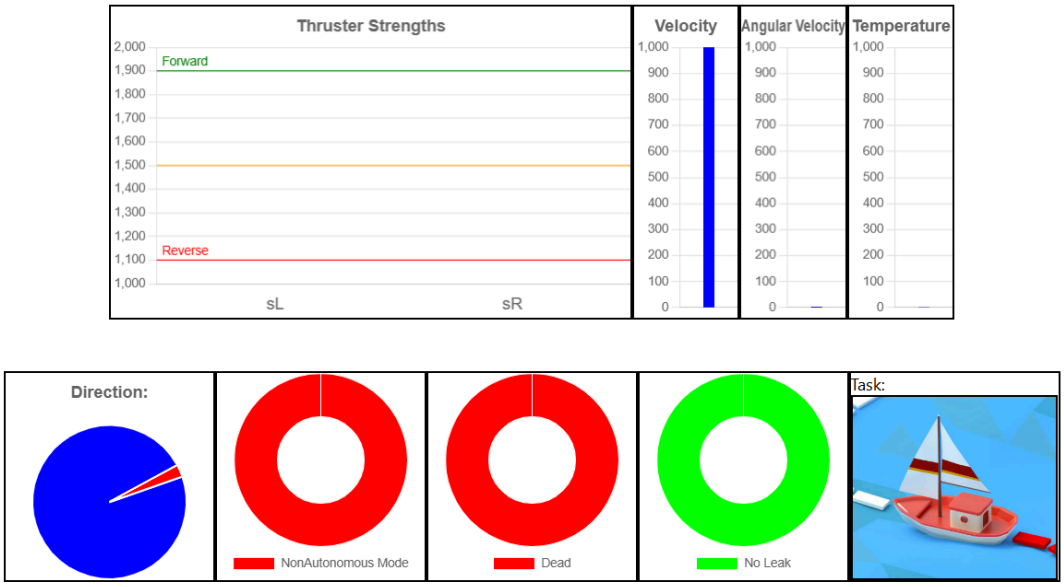


Figure 19: Our UI dashboard, showing metrics such as thruster strengths, velocity, and direction

XII. Appendix E: Electrical Schematics

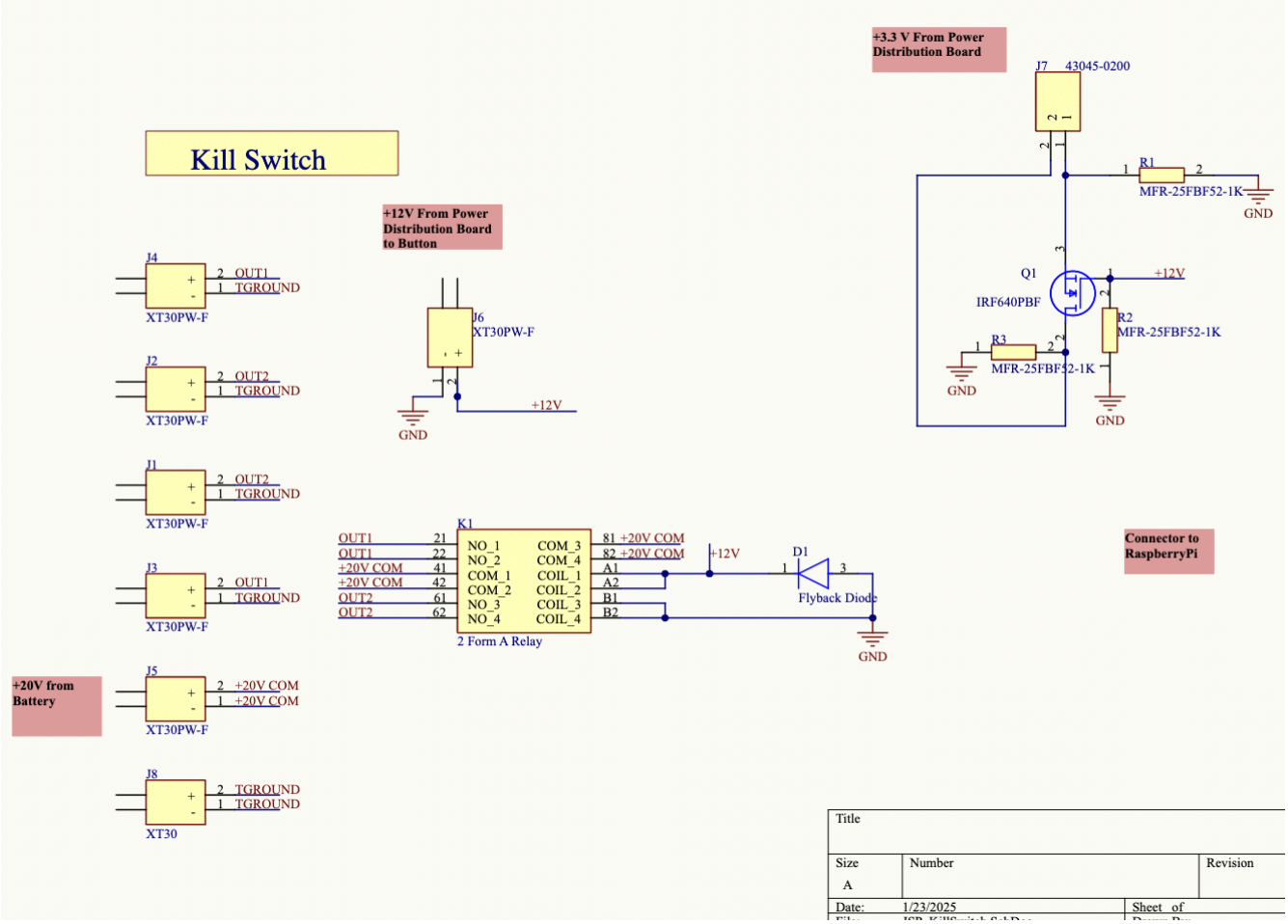


Figure 20: Kill switch board schematic

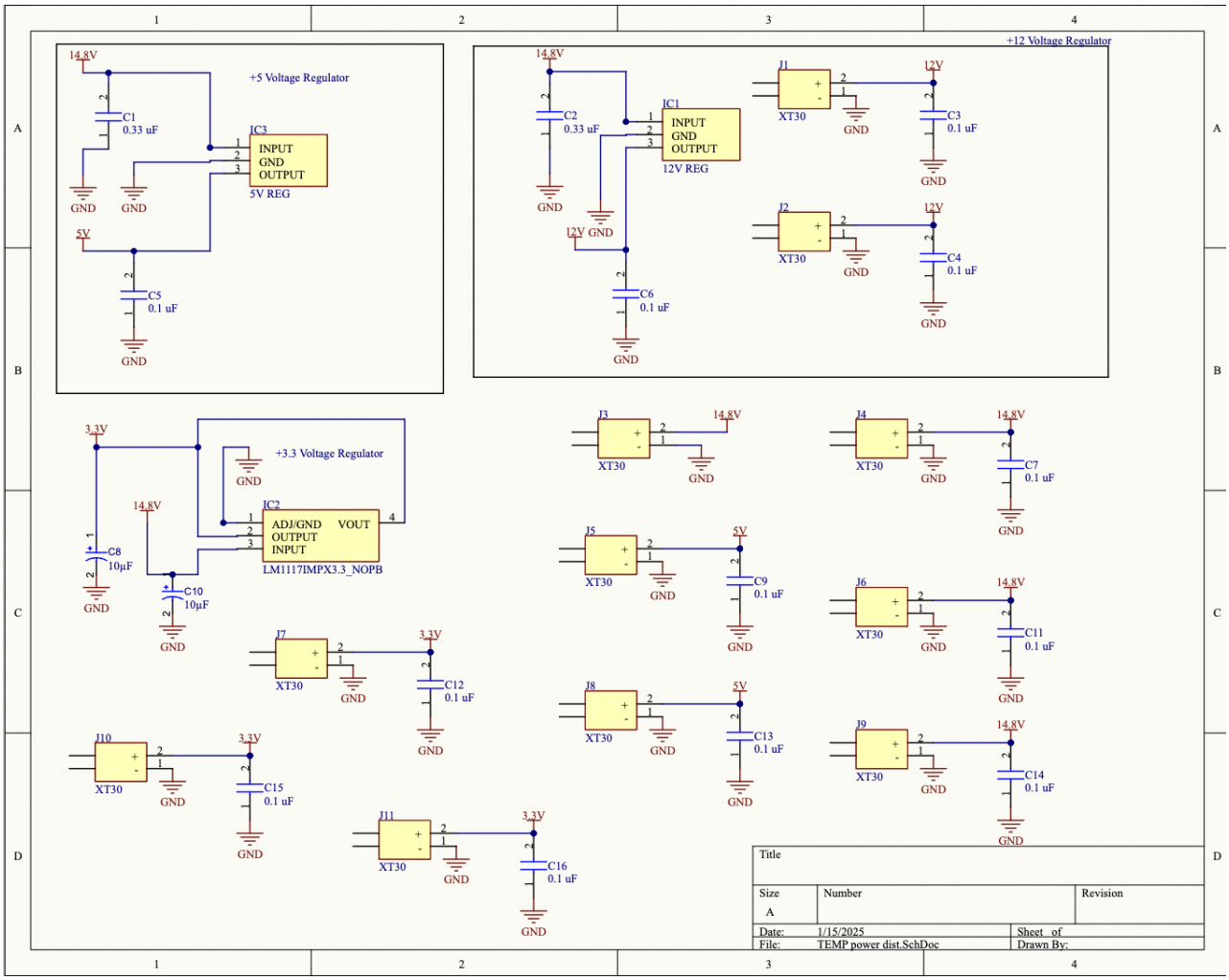


Figure 21: Power distribution board schematic

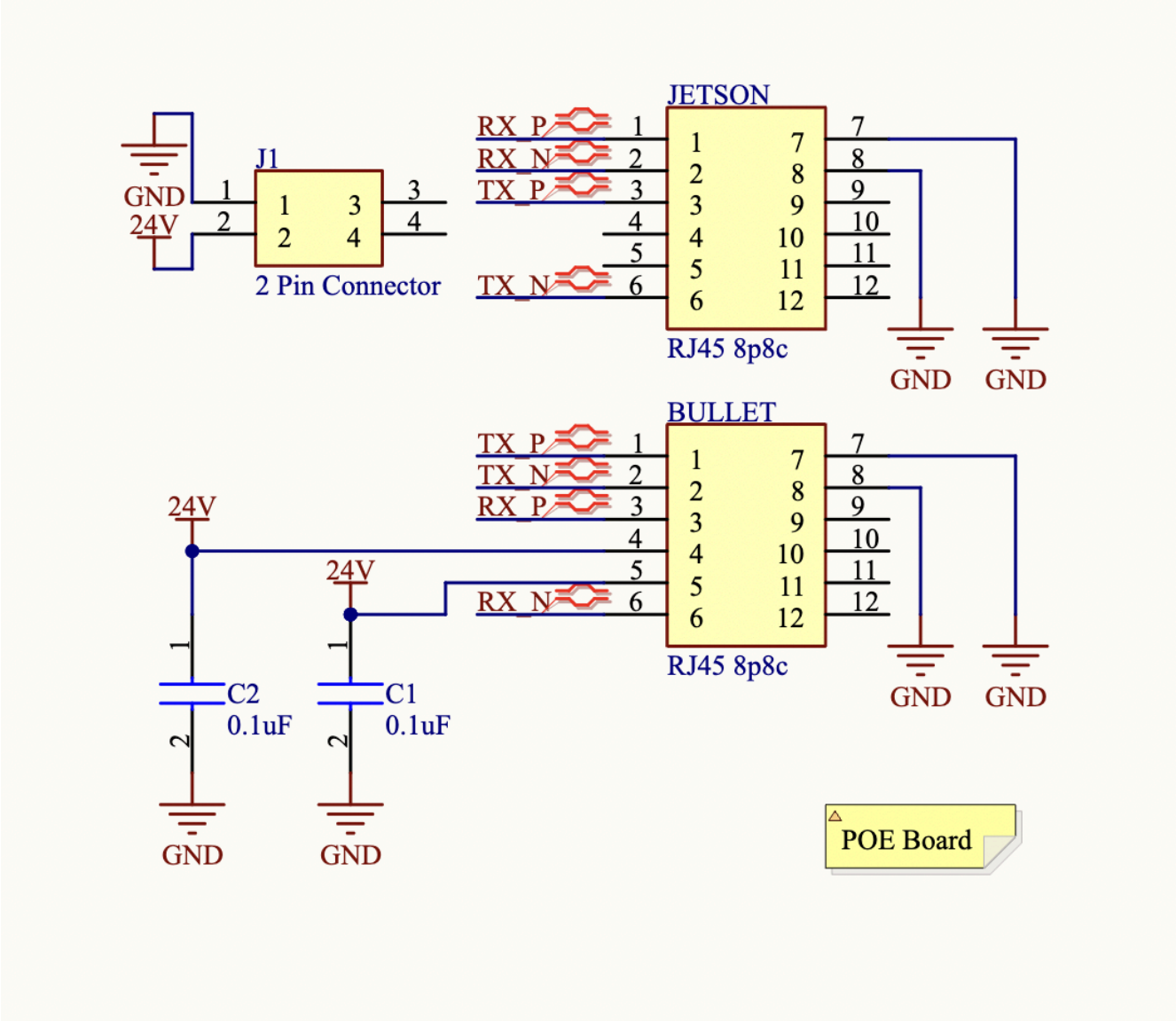


Figure 22: POE board schematic