Team Inspiration's 2026 RoboBoat Autonomous Surface Vehicles

Chase Chen (team lead), Cesar Barranco, Jayden Breesam, Keith Chen, Carina Dumitrascu, Jakob Roche, Parsa Sedighi

Abstract — Team Inspiration leveraged lessons learned from RoboBoat, RoboSub, and RobotX and other teams, including the SeaSentinel [1], in designing, building, and testing a new Autonomous Surface Vehicle (ASV) to address this season's new multi-vehicle requirement [2]. We upgraded our software framework from bare metal Python to Robot Operating System 2 (ROS2) [3]. Besides the availability of a large number of software drivers and robotics software libraries in ROS2, it provides tools for inter-process and vehicle-to-vehicle communication, freeing part of our software team to focus on higher-level software challenges. Both Crusader (our new ASV) and Barco Polo (our trusted ASV) (Fig. 1) benefit from upgraded microcontrollers for faster real-time control, with Crusader also receiving an improved power distribution system. Crusader has redesigned hulls that improve the ASV's maneuverability and stability, a more powerful embedded computer, and an additional Inertial Measurement Unit (IMU) fused with a Global Navigation Satellite System (GNSS) to provide position estimation, creating an Inertial Navigation Satellite System (INSS). Moreover, we improved our sensor fusion between our camera and LiDAR for better spatial detection.

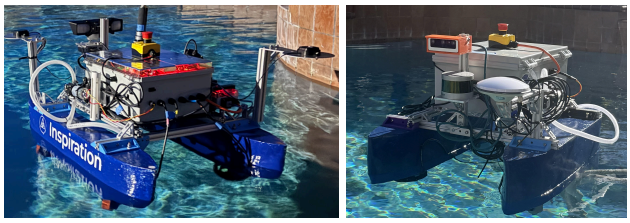


Fig. 1. Barco Polo (left), Crusader (right).

I. INTRODUCTION

Team Inspiration is a community team that participates in international robotic competitions, including all major RoboNation events, and STEM outreach. Notably, our team comprises 80% first-year RoboBoat students from middle schools and high schools in San Diego, San Diego Mesa College, University of California San Diego (UCSD), San Diego State University (SDSU), and remote members in Florida. Our diverse group of students enables overall growth and development, with members ranging from middle school to top-ranked universities.

To prepare team members for the RoboBoat 2026 competition, five small autonomous land vehicles, called MonkeyBot, were developed from the ground up, enabling members to learn fundamental robotics engineering concepts, including real-time control and coding in Python. Team members have been preparing for RoboBoat, since September 2025, including learning and working through the holidays. See more details in Appendix G.

II. COMPETITION STRATEGY

A. Overall Mission Strategy and Execution

We are developing our ASVs to divide the mission tasks using a map grid, the top half, or the bottom half. Both ASVs will pass through the gate, but Barco Polo will be the one to return to earn points. Barco Polo will go through the gate, navigation channel, debris clearance, and object delivery. Crusader will do the speed challenge and docking. When harbor alerts sound, Crusader will go to the corresponding area, if needed, Barco Polo can take over the mission depending on decision criteria in our algorithms, see details in II.B.7. Our overall strategy is to minimize time to complete all missions and return home, earning additional time bonus points.

The team improved the mechanical and electrical subsystems, updated the software framework of Barco Polo from RoboBoat 2025 [4], and built our second ASV to meet this

season's Inter-Vehicle Communication (IVC) requirement.

We based our core design for Crusader on the proven Barco Polo sub-system, reducing risk and time. Based on our RoboBoat acquired experience with fiberglass, we simplified manufacturing, maximizing time availability for electrical and software testing, and reduced the manufacturing time from 5 weeks to 4 weeks. Moreover, software development was accelerated using Barco Polo while Crusader was being built. Both ASVs were designed to have the capabilities to complete all the missions (Fig. 2).

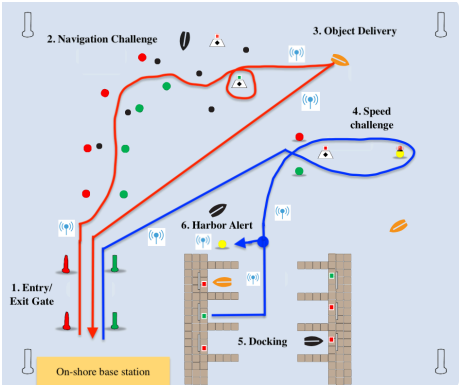


Fig. 2. Barco Polo's path for tasks 1-3 (red trajectory), Crusader's path for tasks 1, 4-6 (blue trajectory). Inter-Vehicle communication (antenna).

B. *Detailed Mission Plan*

We divided the mission tasks into a top and bottom half; the completion logic for each mission follows. Refer to Appendix C for the planned mission capability and its priorities.

1) Evacuation Route & Return (Entry): Both ASVs will begin in the GateEN (GEN) state, Barco Polo executing the entry first. The ASV will detect the first pair of pylons and record its gate-entry position, calculate the midpoint between the pylons, and then generate a waypoint 1 meter beyond the second pair of pylons.

While approaching the generated waypoint, it will continuously adjust its path using a smooth best-fit curve to align with the midpoint of the second pylon pair. After passing through the second pair of pylons, Barco Polo will transition to the NavChannel (NC) and communicate the completion of this portion to Crusader, signaling it to begin the mission. Crusader will complete this portion using the same logic and transition to the SpeedChallengeEN (SCEN) state upon completion.

2) Evacuation Route & Return (Exit): Upon confirmation that all missions have been completed, Barco Polo will transition to GateEX state (GEX) and navigate to the gate-exit waypoint, then navigate towards the gate-entry waypoint. Barco Polo will transmit another *GatePass* message upon crossing the pylons to indicate completion of the return portion of the mission.

3) Debris Clearance (Navigation Channel): Barco Polo will be operating in the NC state and actively searching for pairs of green and red buoys. Upon identifying the closest buoy pair, the ASV will compute the midpoint between them and navigate through sequential buoy pairs using logic similar to the one utilized in *Evacuation Route & Return.* As it navigates through each pair, it will avoid the black buoys and report their location via an *ObjectDetected* message. At the same time, the ASV will continuously refine the estimated position of the next set of buoys and will have prior knowledge of how many pair buoys there are, from a pre-loaded mission map manually generated by operators. After passing the final pair of green and red buoys, it will transition into the IndicatorDetection (ID) state.

In the ID state, Barco Polo will focus on detecting color indicators and black buoys, then report them using an *ObjectDetected* message. If a green indicator is detected, it will approach the indicator while keeping it centered in view until it is 2 (TBD, based on venue conditions) meters away. The ASV will then execute a circumnavigation maneuver. If the red color indicator buoy is detected, it will report an *ObjectDetected* message and avoid it similarly to a black buoy. Once this mission is completed, ASV will switch to ObjectDelivery (OD) state.

4) Emergency Response Sprint (Speed Challenge): Crusader will be in the SCEN state and searching for the *Speed Challenge* gate buoys. Upon detection of the gate buoys, Crusader will calculate the midpoint between them, record its pre-entry position, and generate a

waypoint 1 (TBD, based on venue conditions) meter beyond the midpoint.

Crusader will then transition to SpeedChallengeEX (SCEX) state and search for the color indicator buoy and the yellow buoy. It will then detect, determine the color of the color indicator, and report an *ObjectDetected* message. Then it will surge forward towards the generated waypoint and report the first *Speed Challenge GatePass* message. Once the yellow buoy is detected, Crusader will use the same logic for the green color indicator in the *Navigation Channel* to circumnavigate it. It will then generate waypoints utilizing the post-entry position and pre-entry position to navigate back out of the gates.

Crusader will transition to DockingStart (DS) state, and report a second Speed Challenge *GatePass* message upon completion.

5) Supply Drop (Object Delivery): Barco Polo will operate in the OD state and navigate to the target boat by generating a waypoint using the mapped target position. It will approach the waypoint until it is approximately 1.5 meters away from the target. It will then slowly circumnavigate the target by swaying and using computer vision to yaw. The ASV will utilize the detection bounding box's size and width-to-length ratio to verify that the ASV is approximately perpendicular to the target vessel.

Small sway adjustments are then applied to align the target with the center of the image. Once the target center is aligned with the camera's image center within a 5% error margin, the corresponding supply is delivered, and an *ObjectDelivery* message is reported.

6) Navigate the Marina (Docking): Crusader will be in the DS state and navigate to the approximate location of the dock using a pre-defined map. It will use LiDAR to find the approximate midpoint of the entrance of the dock, navigate towards it, and record its dock-entry position. It will then transition into Docking (D) state, communicate the transition of the state via IVC to Barco Polo, face north, and sway east to search for available slots.

If no available slot is found on the north side, the ASV will then face south and sway west to search for an open slot. Once an available slot is detected, it records its pre-docking position, centers itself with the slot, and surges forward to dock safely without hitting any walls. After being docked for 5 seconds, it will report the *Docking* message and then transition to the DockingEnd (DE) state. It will generate waypoints and navigate out by following the pre-docking position waypoint to undock, then exit the dock through the dock-entry waypoint.

7) *Harbor Alert:* The audio algorithms onboard the team's ASVs will continuously listen for the *Harbor Alert* tone. Crusader is the primary robot for this task; in the event that Crusader is actively executing a mission, Barco Polo will take over for execution of the mission. The ASVs will communicate to determine which will handle the mission via IVC. The ASV will plan its route using an A* path-finding algorithm detailed in [5]. This approach determines the shortest path to the target while avoiding obstacles on the way. The ASV will transition to HarborEmergency (HE) state for one blast and HarborMarina (HM) state for two blasts.

8) Inter-Vehicle Communication (IVC): Upon successful completion of each task, the ASV transmits the completion status to the other ASV. Transmission is indicated by 5 seconds of continuous green light, and acknowledgement of successful reception is indicated by the other ASV through 3 flashes of green light. Trans and receiver roles are played by both ASVs, as each keeps track of the other ASV's progress while also informing them of their own in real time. The interaction between the ASVs will be recorded at the ground terminal for reference. Failing to complete and report a task as successful prompts the other ASV to queue and attempt the task after its initial run.

III. DESIGN STRATEGY

Software capabilities were developed in response to new mapping challenges, while the mechanical and electrical subsystems were further refined to improve reliability.

A. Mechanical Subsystem

Inspired by Barco Polo, a foam-core fiberglass wet-layup hull was refined with an epoxy fairing compound to achieve a smooth, hydrodynamically efficient finish. A catamaran configuration with holonomic vectored thrusters was repeated for Crusader for efficient position hold and maneuverability. To improve efficiency compared to previous construction, we UV-cured all three fiberglass layers simultaneously, promoting interlayer bonding. Additionally, we improved the placement of the fiberglass mesh on the outer layer, reducing the amount of filler compound needed for a smooth surface finish (Fig. 3).

Fig. 3. Applying fiberglass (left) and finished product (right).

A UCSD capstone team made improvements to the existing racquetball launcher, including a front-loading barrel for increased accuracy and a compound gear that compresses the spring further for a greater launch distance (Fig. 4).

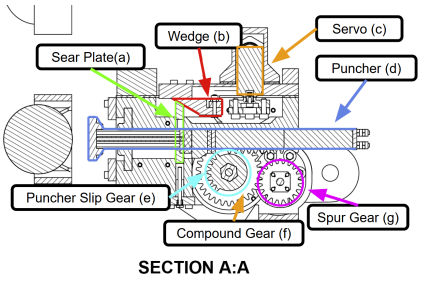


Fig. 4. Section view of parts required for launching.

In addition, the water gun for the *Rescue Delivery* task is mounted on both ASVs for redundancy. The pump intakes water, which is routed through a nozzle that tapers inward to increase the exit velocity, which increases the jet's range.

B. Electrical Subsystem

Both vehicles receive power to their electrical components from 18Ah 4S LiPo batteries. The safety system includes a physical kill switch and a remote control shutdown mechanism. An internal circuit breaker protects the overall system power, while the remote control kill switch and external button interface are limited to stop the propulsion circuit, ensuring that our embedded computer, a Jetson Orin Nano, and critical sensors remain powered for safe operation and diagnostics.

We upgraded our microcontroller from an Arduino to a Teensy 4.1 to improve the real-time control of the thrusters. The Teensy manages the thrusters and position estimation by fusing the GPS and IMU readings. Our redesign allows the embedded computer to focus on high-level control, mission logic, and decision-making. Refer to Appendix F for more details.

Our ASVs use a dual-input GNSS module to determine heading, longitude, and latitude, minimizing drifts and interferences normally found on consumer-grade electronic compasses. Crusader and Barco Polo are paired with a long-range stereo camera, OAK-D LR, whose stereo capabilities provide up to 30m of depth readings and enable obstacle avoidance through mapping. Additionally, IVC is achieved using two bullet antennas in a mesh network. In our previous competition season, we built custom mounts, but our challenge was limited space (Fig. 5A). Crusader features a larger electrical box for ease of maintenance and future expansion (Fig. 5B).

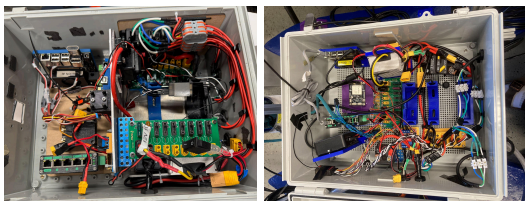


Fig. 5A & 5B. Barco Polo's electronics box (left) and Crusader's electronics box (right).

C. *Software Subsystem*

The team shifted from Python threading to ROS2 to leverage the extensive open-source tools for mapping development. The team also configures the environment within Docker, making it easier to set up the same environment with required dependencies in another vehicle, and provides version control.

Tests and data analysis helped us evaluate sensor accuracy, and ROS2 enables the team to easily collect sensor data using ROS Bags. This allows us to visualize our mapping and localization with ROS Visualization 2 (RViz2) and FoxGlove. Furthermore, we can now split the system into nodes that communicate via defined interfaces, allowing software members to test individual pieces of code before integrating them into the system. The software subsystem is divided into four main stacks: the Application Programming Interface (API) stack, the perception & mapping stack, the mission planner stack, and the navigation stack (Fig. 6).

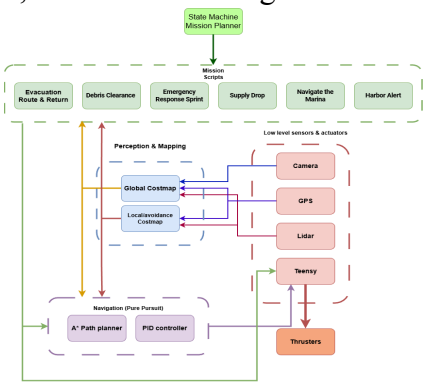


Fig. 6. High-level software stacks.

1) API Substack: The API stack is responsible for communicating with the hardware. Using Docker, we can easily configure the environment and use an open-source ROS2 driver for various sensors on our vehicle. With these existing drivers, we can focus on high-level sensor fusion, target mapping, and mission planning.

2) Perception & Mapping Substack: Leveraging the UCSD capstone team's LiDAR-camera fusion algorithm in [6], the substack involved an OAK-D camera and LiDAR. Using *depthai_ros2_driver* in [7], we are able to acquire object detection in 3D space relative to the camera. We are also planning to apply LiDAR-camera projection to give depth information to pixels and retrieve an object depth from LiDAR. To provide a more accurate estimate of object distance, we use a simple complementary filter to fuse depth estimates from OAK-D and LiDAR. The 3D positions are then passed to the Mapping substack, where local detections are transformed to a global map frame using ASV's odometry data calculated from the GNSS module.

As our ASVs traverse the course, we will utilize a Kalman Filter to give an improved estimate of the position of mission targets. When a new detection arrives, we determine the closest existing object around it. If the distance exceeds the threshold, we recognize that detection as a new object, and we update the existing position. For more details about the data pipeline for the mapper, see Appendix E.I.

3) Mission Planner Substack: With the new *Harbor Alert* mission this year, the team is developing a Finite State Machine along with ROS2 Action Server/Client to manage the high-level mission planning. ROS 2 Action Server/Client allows us to run the mission and receive real-time feedback. Each mission is represented as a discrete state with defined entry conditions, execution logic, and exit criteria. Feedback from the ROS 2 Action Service provides our ASVs with a clear understanding of their mission state (Fig. 7).

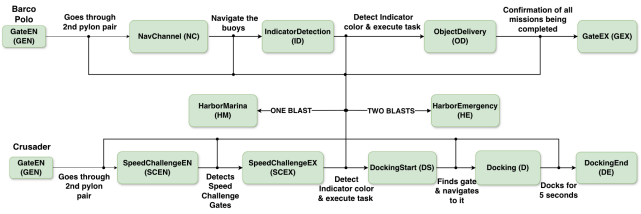


Fig. 7. Mission Script Substack. Determines mission state Barco Polo is in and how to transition between them.

4) Navigation Substack: The Navigation stack will utilize the global map produced from the Perception and Mapping stack. A local occupancy grid is generated from LiDAR data to compute collision-free paths to the current

objective using an A* path planner. The generated path is then followed by a pure pursuit controller.

IV. TESTING STRATEGY

Through a systems engineering approach, we optimize our testing strategy. After breaking down requirements into the smallest action items, we conduct unit/component tests on land to ensure rapid testing before integrating them into the system. Subsystem testing is conducted to test individual functionalities such as waypoint navigation. Systems testing involved multiple subsystems working together to complete missions in sequence. During each test, members are required to fill out a template test plan, recording the vehicle's state, test procedures, and their results. See more details in Appendix B.

A. Unit Testing

As we obtain new parts for integration into the ASVs, we unit-test them to verify their functionality. For instance, during the Crusaders' electrical system assembly, a digital voltmeter was used to measure voltage across a circuit breaker to confirm proper power transmission, which identified a faulty breaker that was inhibiting current flow.

B. Subsystem Testing

Harbor Alert Testing: To ensure that audio detection algorithms worked under various conditions, the team thoroughly tested the One-Class Support Vector Machine (SVM) [9] with generated audio. A preliminary synthetic test plan had a test dataset containing 2,000 artificially generated audio samples, 1,000 of which contained the desired tone, and the other 1,000 contained randomly generated noise with no significant tones. The tone detection architecture was tested to be up to 100% accurate in discerning both signal-containing and non-signal-containing tones, taking on average 160 microseconds per tone. Refer to Appendix E.IV for test data and more information.

C. System Testing

The team conducted a lake test in San Diego to approximate our ASV's experience on the competition site, and allowed us to test our system as a whole, ensuring we're running into conditions that appear in natural bodies of water. During the test, we analyzed our ASV's capability to execute missions.

D. Simulation/Remote Testing

Simulation enables rapid testing of mission logic without relying on physical water trials. This reduces development time and helps identify issues earlier. It also allows team members in Florida to run tests remotely without access to the vehicle in California. By testing in a virtual environment, the team can safely explore edge cases and refine behaviors before deploying them on the real ASV. More details in Appendix E.III.

V. ACKNOWLEDGEMENTS

Team Inspiration would like to thank the supporters below for monetary and equipment donations:

Diamond: Advancing Science, Technology and Art, the Chen family, Robonation/ONR, and Gilman Charitable Fund.

Platinum: Nvidia.

Gold: L3Harris, Qualcomm, and tinyVision.ai.

Silver: Blue Trail Engineering, Blue Robotics, and Chris Freeman.

Bronze: ePlastics, Manna's Martial Arts, and Sunrez.

We are grateful to our mentors Venkat Rangan, Alex Szeto, Jack Silberman, and Teresa To, who helped organize the team, continuously reviewed our work, provided technical suggestions, and guided team planning.

We would like to thank our alumni mentors Khushee Goel, Colin Szeto, Mabel Szeto, Eesh Vij, and Lindsay Wright for using their experience to weigh in on design decisions made by the team. Team Inspiration would also like to thank the Office of Naval Research and RoboNation for the opportunity to compete in a prestigious event such as RoboBoat.

Finally, we would like to thank our parents for their support, without which none of our accomplishments would have been possible.

REFERENCES

[1] SimLE SeaSentinel Team, “RoboBoat 2025: Technical Design Report SimLE SeaSentinel Team” 8 January 2025. [Online] Available: https://robonation.org/app/uploads/sites/3/2025/02/TDR_SeaSentinel_RB2025.pdf. Accessed Jan. 19, 2026.

[2] RoboNation, “RoboBoat 2026 Team Handbook,” 12 December 2025. [Online] Available: Introduction & Table of Contents | RoboBoat 2026 Team Handbook. Accessed Jan. 19, 2026.

[3] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, “Robot Operating System 2: Design, architecture, and uses in the wild, “Science Robotics vol. 7, May 2022. [Online]. Available: https://arxiv.org/pdf/2211.07752.pdf. Accessed Jan. 19, 2026.

[4] K. Chen, et al., “The Design of Team Inspiration’s 2025 RoboBoat ASV”, Team Inspiration, San Diego, California, United States, 2024. [Online]. Available: https://robonation.org/app/uploads/sites/3/2025/02/TDR_TeamInspiration_RB2025-compressed.pdf Accessed: Jan. 19, 2026.

[5] S. D. Lai, *A* Pathfinding Algorithm Visualizer*, 2025. [Online]. Available: https://scottdlai.github.io/a-star-pathfinding/. Accessed: Jan. 1, 2026.

[6] S. Yoo, Y. Saravanan, S. Bian, M. Khattar, and B. Jeong, *RoboBoat: Camera-LiDAR Fusion for Autonomous Boat Navigation*, 2025. [Online]. Available: https://plesiosh.github.io/RoboBoat_SP25/. Accessed: Jan. 19, 2026.

[7] Luxonis, “DepthAI ROS Driver,” *DepthAI v3 Documentation*, 2025. [Online]. Available: https://docs.luxonis.com/software-v3/depthai/ros/driver/. Accessed: Jan. 12, 2026.

[8] J. DeBari, “DTMF Tones and Signaling Explained,” *www.onsip.com*. https://www.onsip.com/voip-resources/voip-fundamentals/dtmf-tones-and-signaling-explained. Accessed Jan. 6, 2026.

[9] J. Roche, “Using A One-Class SVM To Optimize Transit Detection,” *The Open Journal of Astrophysics*, vol. 7, Jul. 2024, doi: https://doi.org/10.33232/001c.121826. Accessed Dec. 9 2025.

[10] “2.5: Noise Modeling - White, Pink, and Brown Noise, Pops and Crackles,” *Engineering LibreTexts*, May 19, 2020. https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_%28Woolf%29/02%3A_Modeling_Basics/2.05%3A_Noise_modeling-_more_detailed_information_on_noise_modeling-_white_pink_and_brown_noise_pops_and_crackles Accessed Jan. 11, 2026.

Appendix A: Component List

Component	Vendor	Model/Type	Specs	Custom/ Purchased	Cost	Year of Purchase
ASV Hull Form/Platform (Frame, Waterproof Housing)	80/20 Inc.	1010 aluminum extrusion	Weight: 25 oz Size:115 in	Purchased and machined in the lab	$234.73	2019
	Lowe's	FOAMULAR NGX F-250 Unfaced Polystyrene Board Insulation	R-10, 2-in x 4-ft x 8-ft	Purchased	$51.98	2023
	Home Depot	Maple Plywood, ⅛ in. series	Thickness: ⅛"	Purchased	$23.99	2024
	Fibre Glast Developments Corp.	1.5 Oz Chopped Strand Mat	Quantity: 2 Length: 5 yards Width: 38 In	Purchased	$93.90	2023
	Sunrez Composites	7315 Polyester Resin	Quantity: 3 gallons	Donated	$320	2023
	Fibre Glast Developments Corp.	2 oz Fiberglass Fabric	Quantity: 2 Length: 3 yards Width: 38 In	Purchased	$49.90	2023
	McMaster-Carr	1016N215 Cable Entry Panels	Supported cable diameter: 0.22"	Purchased	$25.95	2023
	MAKERELE	Cord Grip Cable Glands Kit	1/4", 3/8", 1/2", 3/4", 1", 1-1/4" https://www.amazon.com/gp/product/B08R86BHBC/ref=ppx_yo_dt_b	Purchased	$24.99	2023

			_search_asin_title ?ie=UTF8&th=1			
Water Cannon	Bayite	12V DC Fresh Water Pressure Diaphragm Pump	https://www.amazon.com/gp/product/B01N75ZIXF/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1	Purchased	$20.99	2024
	Eastrans	3/8" ID x 10 Ft High-Pressure Braided Clear PVC Vinyl Tubing	https://www.amazon.com/gp/product/B07MTYMW13/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1	Purchased	$12.99	2024
	Everbilt	1/2 - 1-1/4 in. Stainless Steel Hose Clamp	https://www.homedepot.com/p/Everbilt-1-2-1-1-4-in-Stainless-Steel-Hose-Clamp-6712595/202309385	Purchased	$2.18	2024
Racquetball Launcher	AndyMark	NeveRest Classic 60 Gearmotor	https://www.andymark.com/products/neverest-classic-60-gearmotor	Purchased	$34	2024
Propulsion	Blue Robotics	T200 thrusters	Full Throttle FWD/REV Thrust @ Maximum (20 V): 6.7/5.05 kg f https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/	Purchased	$200 apiece $230 apiece	2019, 2025
	ELEGOO	Elegoo UNO R3	https://a.co/d/c9i5BFC	Purchased	$15.99	2024
Motor	Blue	Basic ESC	30A brushless	Purchased	$36	2019

Control	Robotics		ESC https://blueroboticscom/store/thrusters/speed-controllers/besc30-r3/		apiece $40 apiece	2025
	Teensy	Teensy 4.1	https://www.sparkfun.com/teensy-4-1.html	Purchased	$31.50	2025
Power System (Battery, Converter, Regulator)	Blue Robotics	Lithium-Ion Battery	4S 14.8V 18 Ah	Purchased	$425	2019
	Circuit Breaker	200 Amp Circuit Breaker with Manual Reset 12V	https://a.co/d/h648Kfr	Purchased	$29.99	2025
	Push-Button Switch	Emergency Stop Enclosed Push-Button Switch	https://www.mcmaster.com/6785k21/	Purchased	$69.69	2025
	ElecDirect	200A Manual Reset Type 3 Surface Mount	https://www.elecdirect.com/fuse-holders-circuit-breakers/circuit-breakers/200a-high-amp-manual-reset-type-3-surface-mount-12-42-volts	Purchased	$29.95	2023
	Pololu	RC Switch with Omron Relay	https://www.pololu.com/product/2804/specs	Purchased	$12.95	2023
	AEDIKO	DC 5V Relay Module 2 Channel Relay Board with Optocoupler Support High or Low Level	https://a.co/d/5N9Z85c	Purchased	$8.99	2025

		Trigger				
	iRhapsody	4 Pin High Current Relay	120A, 12V	Purchased	$11.99	2023
	McMaster-Carr	7060K16 Insulated Quick-Disconnect Terminals	https://www.mcmaster.com/7060K16/	Purchased	$16.36 per 100 pieces	2024
	McMaster-Carr	7113K612 Ring Terminals	https://www.mcmaster.com/7113K612/	Purchased	$13.74 per 50 pieces	2024
	McMaster-Carr	69145K218 Spade Terminals	https://www.mcmaster.com/69145k218/	Purchased	$14.74 per 50 pieces	2024
	TKDMR	Battery Cable Ends Ring Terminals Connectors Tubing	https://a.co/d/5omEFDm	Purchased	$13.59	2025
Actuator Control	HUAREW	PCA9685 16 Channel PWM Servo Driver Board 12-bit IIC Interface Module Compatible with Arduino and Raspberry Pi	https://a.co/d/3V12mrF	Purchased	$12.95 per 2 pieces	2025
DC-DC Converter	Cllena	Buck/Boost Converter	DC 8V-40V to 12V 10A	Purchased	$29.99	2023

	Matek Systems	Buck/Boost Converter	4A/5-12V and 4A/5V	Legacy		2021
	DROK	Buck converter	12v to 5v	Purchased	$9.99	2025
PoE Switch	Linovision	5 Ports DC12-48V Input Full Gigabit with Voltage Booster	5 Ports Full Gigabit POE Switch with DC12V ~ DC48V Input and Voltage Booster,Total IEEE802.3at POE Power Budget 120W | LINOVISION US Store	Purchased	$109	2023 2025
CPU	Nvidia	NVIDIA Jetson Xavier NX	https://developer.nvidia.com/embedded/learn/get-started-jetson-xavier-nx-devkit	Purchased	$399	2021
CPU	Nvidia	NVIDIA Jetson Orin Nano	https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/	Purchased	$249	2025
Teleoperation	Ubiquiti	airMAX Bullet AC IP67	Communication: GbE, PoE UISP airMAX Bullet AC Dual-Band IP67 Radio - Ubiquiti Store United States	Purchased	$129.00 apiece	2023
	Alfa	AOA-2458-79AF Antenna	https://www.alfa.com.tw/products/aoa-2458-79af?variant=36473962332232	Purchased	$24.46 apiece	2023

	Ubiquiti	Instant Outdoor PoE Converter	https://store.ui.com/us/en/category/all-accessory-tech/products/instant-802-3af-outdoor-gigabit-poe-converter	Purchased	$21.00	2023
	RadioMaster	ER6 ExpressLRS PWM receiver	https://www.tekrc.eu/shop/receivers/expresslrs/radiomaster-er6-expresslrs-pwm-receiver-6-channels/	Purchased	$31.60	2023
	Flysky	FS-iA6B Receiver Radiomaster ER6	https://www.amazon.com/FS-iA6B-Receiver-6-Channel-Compatible-Transmitter/dp/B09STCQCCF/ https://a.co/d/aUiDuj5	Purchased	$18.49 $39.99	2023 2025
	Pololu	4-Channel RC Servo Multiplexer	https://www.pololu.com/product/2806	Purchased	$12.95	2023
Perception	Livox	Mid-360	Specs - Mid-360 LiDAR Sensor - Livox (livoxtech.com)	Purchased	$749	2023
	Velodyne	VLP-16	https://ouster.com/products/hardware/vlp-16	Legacy	Legacy	2022
Inertial Measurement Unit (IMU)	Adafruit	BNO085	https://www.ceva-ip.com/wp-content/uploads/BNO080_085-Product-Brief.pdf	Purchased	$24.95	2024

Camera	Luxonis	OAK-D LR PoE	Resolution: 2.3MP (1920x1200) Frame Rate: 60 FPS max rate Focus: M12 FF (45cm - ∞) https://shop.luxonis.com/products/oak-d-lr	Purchased	$350.00 (50% discount)	2024 2025
Localization and Mapping	Beitian	UM982 GNSS Module Model BT-982K1	https://store.beitian.com/products/beitian-built-in-zed-f9p-navigation-surveying-positioning-precision-agriculture-centimeter-level-rtk-gnss-module	Purchased	$200.88	2023
	Beitien	High Precision Provides Stability And Reliability External Dish	https://store.beitian.com/products/beitian-high-gain-high-precision-gnss-antenna-provide-stability-and-reliability-gnss-signal-for-positioning-applications-bt-800s?variant=50615537369375	Purchased	$40.88 apiece	2025
	U-blox	ANN-MB series L1/L2 GNSS antennae (2x)\	https://content.u-blox.com/sites/default/files/documents/ANN-MB_DataSheet_UBX-18049862.pdf	Purchased	$60.00 apiece	2023

Vision	Custom	Open Computer Vision	Color isolation, binary thresholding, contour approximation, erosion and dilation, area thresholding, and Contrast Limited Adaptive Histogram Equalization (CLAHE)		Free/ Open Source	2019
	Ultralytics	YOLOv8	Real-time object detection and inference https://docs.ultralytics.com/models/yolov8/		Free/Open Source	2022
Autonomy	In-house	Custom (Python)	Object detection integration, mission planning, waypoint navigation		Free	2024
Open-Source Software	Open-Source (N/A)	OpenCV, Python, C++, Linux	Computer Vision, Inter-process communication, programming, and computer operating systems	Custom	Free	2024

Appendix B: Test Plan & Results

Testing followed a structured, consistent approach aligned with our ASVs. Each requirement was verified through unit, sub-system, and system testing, ensuring that all components functioned and integrated.

Test Name	Requirements	Test Description	Planned Test Date	People Involved
Motor Mixing	ASV shall be able to surge, sway, and yaw	1> Deploy ASV in the pool 2> Connect the computer to Teensy and send a small magnitude thruster command 3> Members hold the ASV in place and observe the direction of the current.	01/01/2026	Chase, Parsa, Jayden
LiDAR Testing	ASV shall be able to Map Mission Elements	1> Power up the ASV 2> Measure the distance away from an object through LiDAR and measuring tape, and compare the difference	01/20/2026	Jayden, Chase
Water Pump Test	ASV shall be able to shoot water	1> Deploy ASV in the pool 2> Run water pump node 3> See if the ASV can turn the water pump on and off.	01/20/2026	Parsa, Jayden, Cesar
Racquetball Integration Test	ASV shall be able to shoot racquetball and water	1> finish assembling the racquetball 2> mount on Crusader 3> drill into the box to receive power from 5V pdb 4> water test	01/21/2026	Ivan, Ruth, Emma
GPS Testing	ASV shall be able to localize itself	1> Put ASV on a cart 2> Record GPS data using ROS bag 3> Validate GPS accuracy using Google Maps	01/24/2026	Chase, Cesar, Parsa
RoboCommand Test	ASV shall be able to report heartbeat and mission element messages to the RoboNation Server	1> Power on the ASV 2> Set up the TeamInspirationField router, connect the member's computer (running as a server) to the router through Ethernet 3> Run Robocommand script in ASV's Jetson 4> Receive heartbeat on the member's computer size	01/25/2026	Chase, Parsa, Jayden
Control Stability Test	ASV shall be able to navigate from point A to point B with object avoidance	1> Deploy ASV in the pool 2> Laydown PVC pipes in the pool to measure the distance traveled 2> Send waypoint 3 meters ahead of the ASV and ask the ASV to follow 3> Observe if the ASV	01/25/2026	Chase, Parsa, Jayden

Test Name	Requirements	Test Description	Planned Test Date	People Involved
		overshoots 4> Repeat the test with 5 different waypoints in all directions		
YOLO Model Performance Test	ASV shall be able to Map Mission Elements	1> Follow documentation 2> Set up Jetson to view through OakD and run the YOLO model to test confidence	01/26/2026	Chase, Parsa, Jayden
Microphone Testing	ASV shall be able to pick up the frequencies from 400 Hz to 1000 Hz.	1> Bench test the mic to pick up audio from within the lab 2> Next, see how far the mic picks up audio and see how much accuracy drops depending on distance 3>integrate onto ASV 4>pick up mp3 samples	01/26/2026	Cesar
IMU Testing	ASV shall be able to localize itself	1> Deploy ASV in the pool 2> Drive ASV through remote control and go approximately 5 meters 3> Record IMU reading through ROS Bag 4> Approximate the velocity by integrating the acceleration data and compare it to the real average velocity gain from GPS	01/26/2026	Chase, Cesar
Teensy 4.1 Testing	ASV shall be able to use a microcontroller to operate the thrusters, water pump, GPS module, and IMU	1> Unit test individual components in the lab and record voltage readings before and after 2> Take ASV for a pool test to test the autonomy of each component	01/26/2026	Cesar, Parsa
Waypoint Navigation Test	ASV shall be able to navigate from point A to point B with object avoidance	1> Deploy ASV in the pool 2> Deploy buoys in the pool 3> Change waypoints in the .json file 4> Follow a minimum of 5 waypoints	01/30/2026	Chase, Parsa, Jayden
Mapping Test	ASV shall be able to Map Mission Elements	1> Deploy ASV in the pool 2> Deploy red-green buoys in the pool similar to the competition setup 3> Drive the ASV through Remote Control and collect Data through ROS Bag 4> Visually inspect if the map matches the physical setup in the pool	01/30/2026	Chase, Parsa, Jayden
Path Planner Test	ASV shall be able to navigate from point A to point B with object	1> Deploy ASV in the pool 2> Deploy buoys in the pool.	02/10/2026	Chase, Jayden

Test Name	Requirements	Test Description	Planned Test Date	People Involved
	avoidance	3> Run A* Mission Planner Node and give a waypoint to the other end of the pool 4> Record data, see if the ASV can follow the generated path 5> See if the Path Planner updates faster than 2 Hz for real-time object avoidance 6> See if the ASV avoids obstacles in the pool.		
Mission State Test	ASV shall be able to transition between states	1> Deploy ASV in the pool 2> Set up the evacuation route and navigation channel mission in the pool 3> Load the state machine to the ASV and start the mission 4> See if ASV reports transition between states after completion of the evacuation route.	02/10/2026	Chase, Jayden

<table>
<tr><td colspan="2" rowspan="2">Miramar Lake Test: Square Movement & Perception Data Collection</td><td colspan="2">Approval Authority</td></tr>
<tr><td>[Names of Members]</td><td>[Description of members’ position]</td></tr>
<tr><td colspan="2"></td><td>Chase
Parsa
Jayden
Cesar
Carina
Troy
Ruth
Emma
Ivan
Cecil</td><td>ASV operator
Documentation and recording
Base station control & data
Mission element deployer
Mission element deployer
Logistics
Video recording footage
Communication
Video recording footage
Logistics</td></tr>
<tr><td>Date: 01/04/2026</td><td colspan="3">Mission Title: Navigation Channel, Object Delivery</td></tr>
<tr><td>Test #: [Sequential Order]</td><td>Location: Miramar Lake, San Diego, CA.</td><td colspan="2">Risk: [Low, Medium, High]</td></tr>
<tr><td>Software Version: [Baseline, Preliminary, Version 0.0]</td><td>Hardware Sensors Mounted:
LiDAR, Oak-D, Bullet AC Antenna, GNSS receivers.</td><td colspan="2">Hardware Sensors Used:
LiDAR, Oak-D, Bullet AC Antenna, GNSS receivers</td></tr>
<tr><td colspan="4">Scope:
<u>Primary:</u> Square movement
<u>Secondary:</u> Perception data collection
<u>Tertiary:</u> Racquetball actuation</td></tr>
</table>

Roles		Walkie Talkies/ Cell Phones		Times		
				Event	**Time**	**Actual**
Test Conductor	Chase	**Ground**	Parsa	**Packup**	10:00	
Boat Launcher	Coach Alex & Cesar	**Kayak**	Cesar	**Go to the Lake**	11:20	
Ground Control	Jayden	**Lifeguard**	Alex	**Test**	11:40	
Data Collector	Jayden	**Deck**	Emma	**Cleanup**	14:40	
Photographer	Ruth & Emma			**Leave Lake**	15:10	
Safety Checker/QA	Chase			**Put Away**	16:00	
Person in water	Cesar/Carina					
Status						
Boat	Cesar	GO/NO GO				
Ground	Chase	GO/NO GO				
Pool Deck	Emma	GO/NO GO				
Attendance	**Test Notes**					

Resources Needed: Lake test packing list - Google Sheets
Environment: Freshwater lake, afternoon time with mission elements in place (target boats, buoys, light indicators).
Expected Results: Reverification of square movement, perception data collected.
Prerequisites:

- Software: Set up base station on shore, Router, personal hotspot, antenna, and be able to ssh into Barco Polo
- Mechanical: Power generator/ extension cord/table & chairs
- Ground: Set up mission elements.

Test procedure:
Start collecting sensor data through remote control

- Set up mission elements based on the entry and exit missions.
- Have a bunch of elements around for Yolo data image collection.
- Collect Sensor data.
- Ensure mission elements are in place with the mechanical team.
- Initialize required ROS 2 nodes.

Risk Management:

- Wire management.
- Fastened GNSS module and antenna mount.
- Battery Check per request or every 20 minutes.
- Well-tied and anchored mission elements.

Time IN water: 11:45
Time OUT of Water: 14:55

Voltage	16.3 V	15.7V	15.1V	14.8 V	Battery Replaced 16.6V	15.9 V	15.4V	Voltage OUT: 15.2V
Time	11:43	12:37	13:05	13:20	13:20	13:39	14:00	14:55

	Validation Step Description	**Expected Result**	**Reference to Data**	**Pass/Fail**
1	**Testing roboboat_2026/api/servos/ball_launcher_node.py**	The chamber is rotated, the spring is	- Refer to the reflection below.	- FAIL, the chamber of the racquetball was rotated successfully; however, the

	Ran Ball Launcher sevicreRos2 service call /	released, and the Racquetball is shot out towards the target.		spring was not released
2	**Testing the water pump: ros2 service call /toggle_water_pump std_srvs/srv/Trigger "{}"**	The water pump shoots water.	- Refer to the reflection below.	Pass
3	**Ros bag 1: Everything all at once, as seen in the picture. Remote control the boat while we record the camera stream.** **Recording** **Lidar** **Camera** **GPS**		0104_yolo001	Pass, Recorded data
4	**Ros bag 2: This time, focused more on the black target boat** **Recording** **Lidar** **Camera** **GPS**		0104_yolo002	Pass, Recorded data
5	**Ros bag 3: Long range and closer** **Recording** **Lidar** **Camera** **GPS**		0104_yolo003	Pass, Recorded data

6	**Ros bag 4: Focusing on the green color indicator on top of the kayak, moving to the target boat, failed to shoot racquetball at the black + target.** **Recording** **Lidar** **Camera** **GPS**		0104_yolo004	Pass, Recorded data
7	**Ros bag 5: Focusing Everything as seen in the picture** **Recording** **Lidar** **Camera** **GPS**		0104_yolo005	Pass, Recorded data
8	**Waypoint Test 1** Test out the waypoint - Chase At least 30 m range Minimum 5 waypoints for 3 runs Record sensor data along the way Cesar will collect the orange/black boat and color indicators.	Test 1: See above	No Data Collected	Test 1: **Fail,** The heading overshoots
9	Waypoint Test 2	Test 2: See	No Data Collected	Test 2:

		above		Partial Pass, The heading is not strong enough to correct itself. The odometry is good, able to come back to 0,0. The path on the way back is not predictable.
10	Waypoint Test 3	Changed the P controller to the Segment controller Test 3: See above	0104_waypoint003	Test 3: Overshoots a lot; the heading was not able to correct itself fast enough.
11	Waypoint Test 4:	Increase the surge power from 0.6 to 0.8 Test 4: 5 Waypoints added: Square movement WP1: 0, 0 WP2: 0, 10 WP3: -10,10 WP4: -10,0 WP5: 0,0	0104_waypoint004	Test 4: Pass Teensy Node crashed
12	Waypoint Test 5:	WP1: 0, 0 WP2: 0, 20 WP3: -20,20	0104_waypoint005	Test 5: The Boat keeps going in a circle

		WP4: -20,0 WP5: 0,0		
13	Test out how far the boat can go before losing communication in SSH.			Pass Able to surpass 100m via ssh without losing connection

Notes during testing:

- Connection through SSH is laggy
- The water pump worked continuously and shoots water when switching to Autonomous mode.
- ROS2 Service doesn't work as expected, unable to turn on and off as expected when running the navigation script
    - The Teensy board requires more software refinement
    - The Teensy board has an I/O error when running for a long time (Water pump).

Results and Reflections:

General:

- Make a more organized test plan. Have someone review the test plan
    - Assign the roles and logistics
- Brief everyone about the goal of the test
- Pack everything the night before

Mechanical

- Fix all mission elements
    - Light tower/indicator
    - Orange boat
- Racquetball
    - Unit test racquetball launcher on Barco Polo
- Microphone
    - Collect audio files and save it to an mp3 file?
    - Integrate to system

Software:

- The water pump service call needs to be fixed
- Unit test racquetball launcher
- Waypoint nav heading is off
    - Recalibrate each run
- ssh latency
- Have a launch file for the test

    - Test in the pool first before testing in the lake
- Practice setting up the router/ base station/ROS nodes, sensors
- Do a 100-meter Waypoint test to see if we can pass enter and exit
- Position hold while aiming for the yellow and black boat
    - A flotation device creates waves in a pool

TODOs:

Mechanical:

- 3D prints
- Color indicator
- Racquetball on Barco Polo

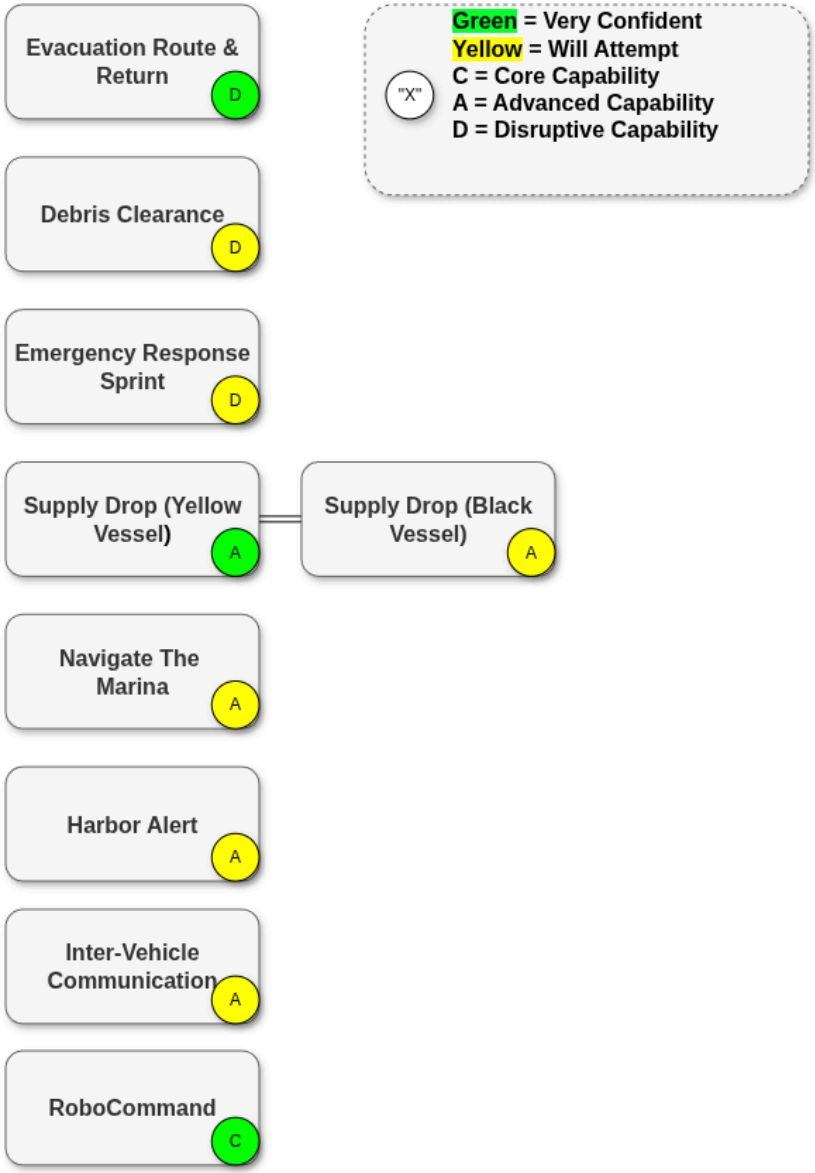
Software:

- Move all rosbags out from the Jetson, upload to Google Drive
    - Convert video from .db3 to mp4
    - Upload to Roboflow
- Check the ROS service for the racquetball launcher and the water pump
- Check error handling for Teensy Node
- Waypoint nav controller needs more tuning

Design documentation:

- Competition Strategy → Jayden

Appendix C: Mission Capability Chart



Appendix D: System Diagrams

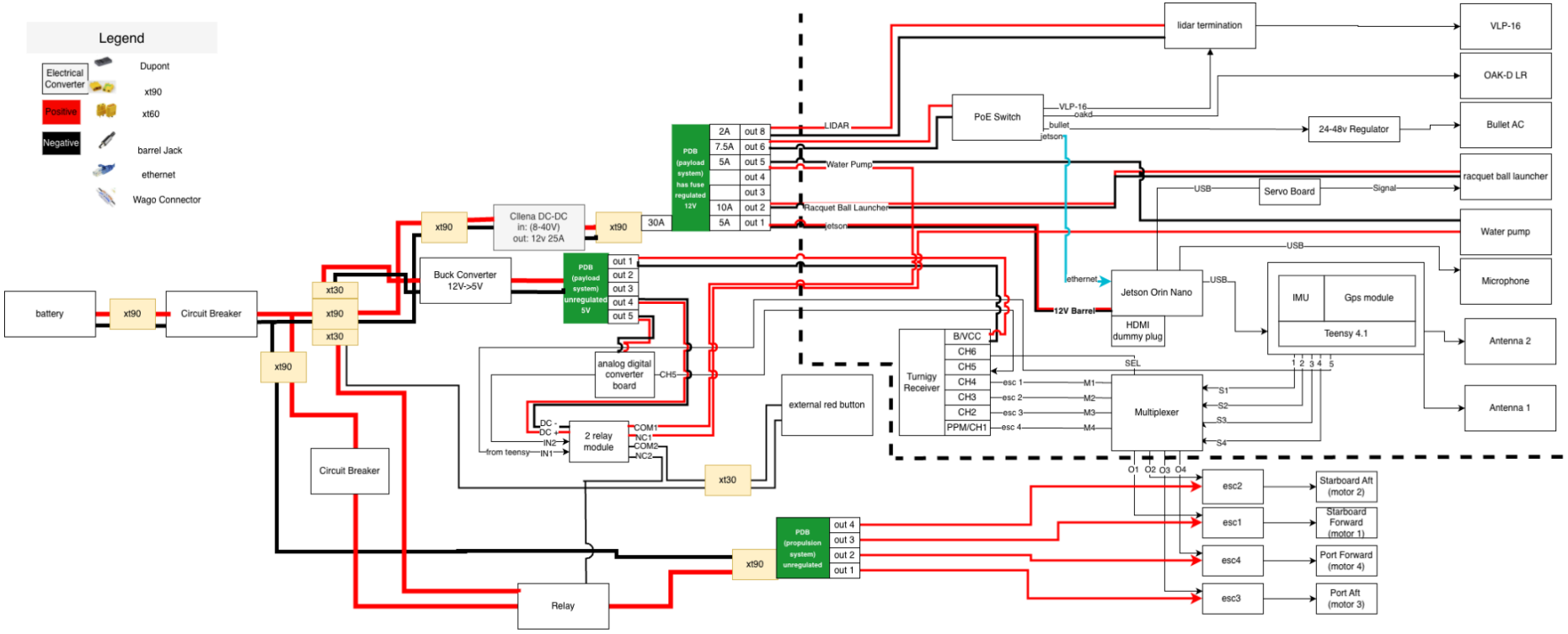


Fig. D.1: Full electrical schematic for Barco Polo.

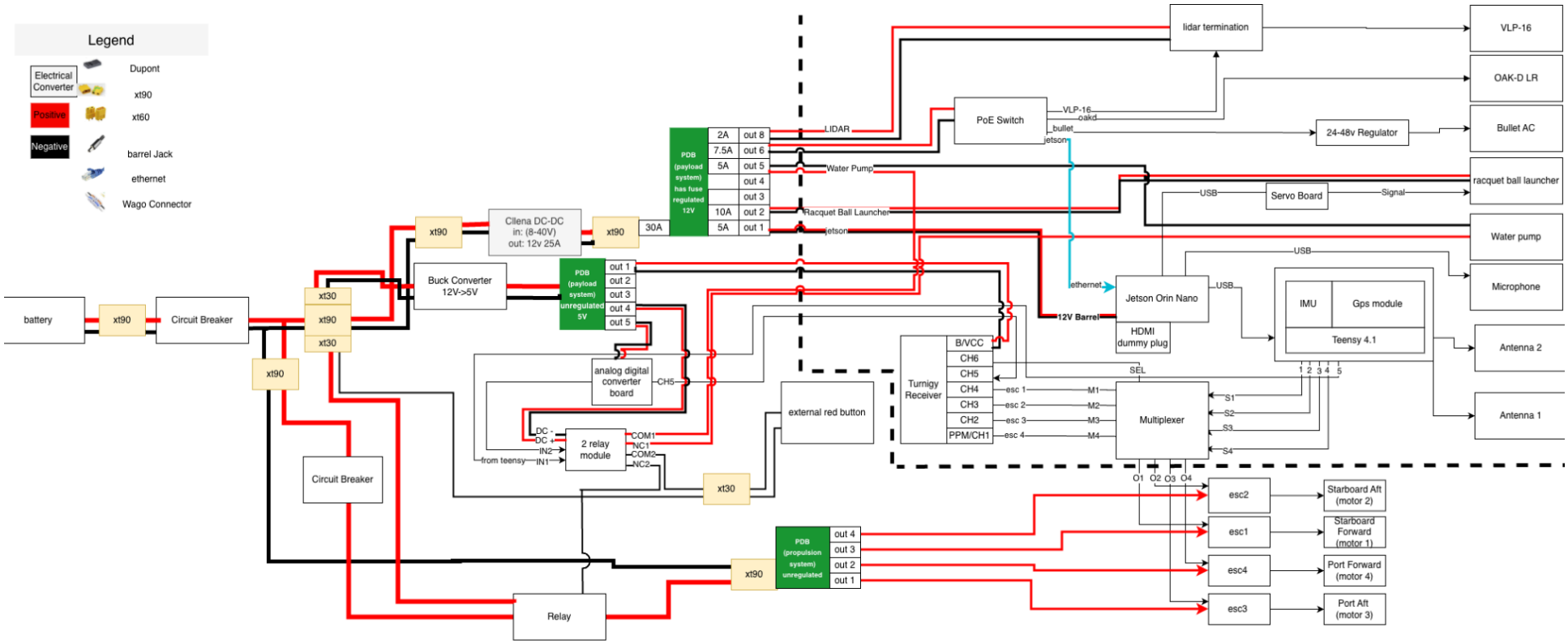


Fig. D.2: Full electrical schematic for Crusader.

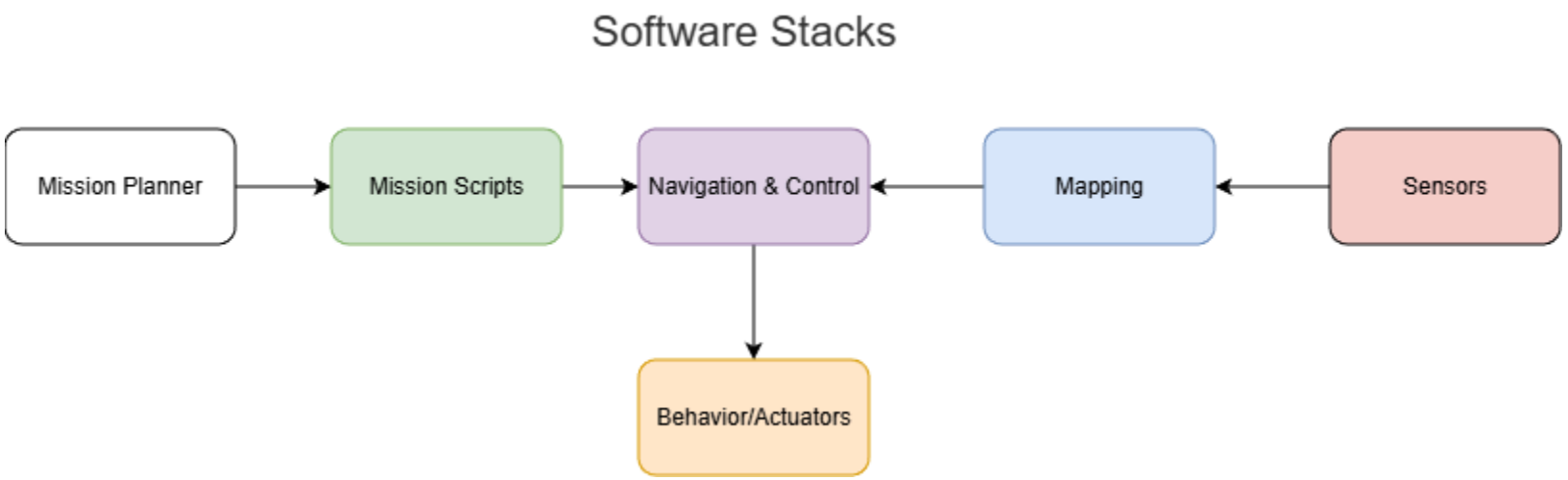


Fig. D.3: High-level software execution stack.

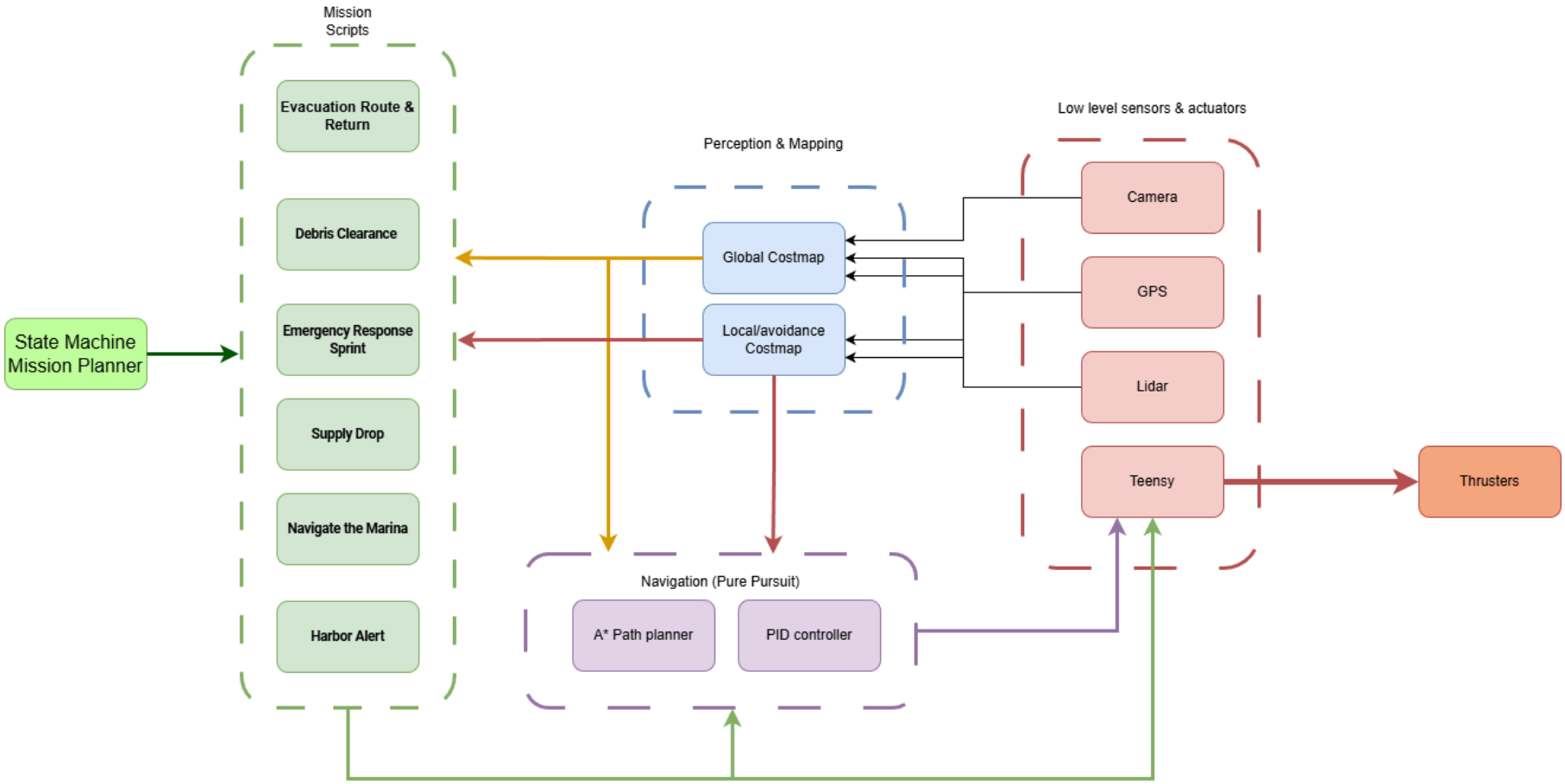


Fig. D.4: Low-level software architecture.

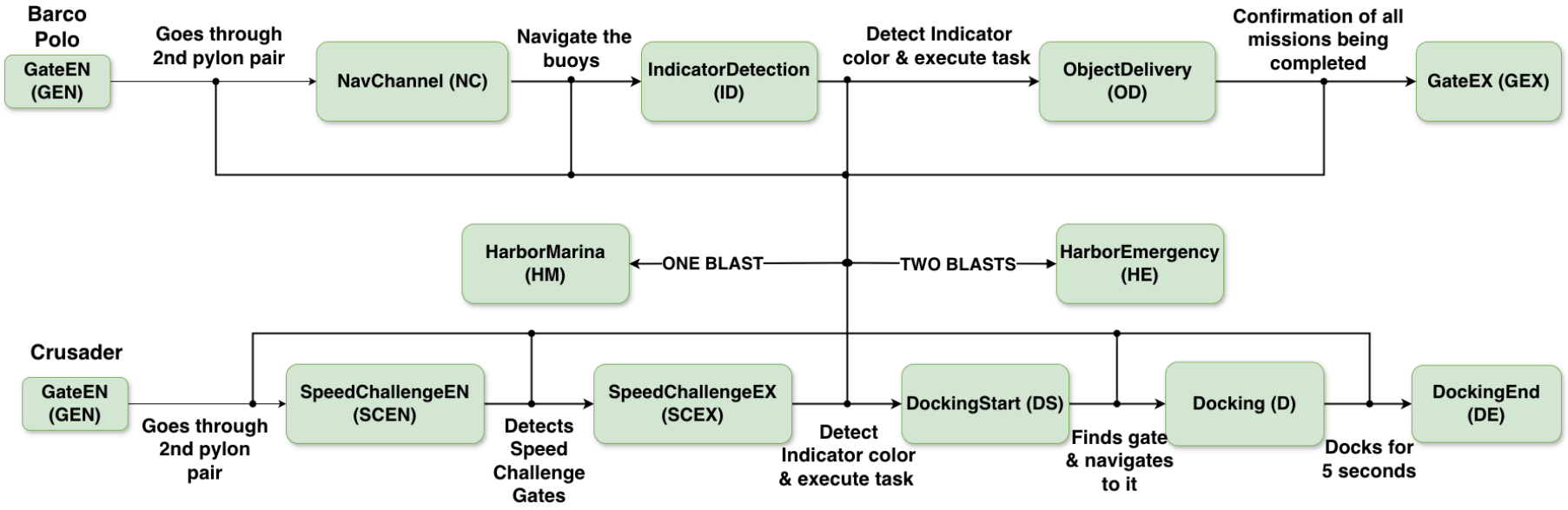


Fig. D.5. Mission Script Substack. Determines mission state of ASVs and how to switch between them.

Appendix E: Software

I. You Only Look Once (YOLO) MODEL TRAINING

In order to train our YOLO model to accurately detect mission elements, we collected images at the lake to ensure the training data closely resembled the real competition course. The YOLO model version we trained is a YOLOv8, which was chosen because of its object detection capabilities and ease of integration with the OAK-D LR stereo camera.

During lake testing, we collected datasets that captured a wide range of variations to improve the model's reliability in dynamic environments. Images of mission elements were taken from multiple angles to account for camera orientation during runs. We also collected data under varying light conditions, like overcast skies and sunny days. This captures different scenarios the ASV might run into, such as soft shadows or glare from the water surface.

In addition, varying the distance from where the mission elements' images are taken allows the model to learn how mission elements appear at different scales. These variations are important because lighting can change drastically due to the weather, ripples in the water, and reflections, and the scale of mission elements continuously changes with distance. All of these environments and static variations provide the YOLO model a robust dataset to train on and provide reliable detections.

Fig. E.1. Annotations for YOLO model

When it came to processing the data, it was about choosing the right images to include in the dataset. Due to our small team and the significant time required for manual image labeling, we decided on about 200 images for each element: yellow buoy, red buoy, green buoy, black buoy, yellow vessel, black vessel, and light indicators. These images were picked based on quality, visibility, and the likelihood of the ASVs seeing something similar to it in the competition.

We also introduced images that are hard negatives into the dataset; these images intentionally contain no mission elements and are used during training to help the YOLO model learn what does not constitute a valid detection. This reduces false positives and helps the model distinguish mission elements from the regular environment.

When it comes to labeling, we use a website called Roboflow, which streamlines the labeling process by providing an interface for drawing bounding boxes, naming classes, and exporting the dataset in a form that is compatible with YOLOv8. To ensure consistency in labeling and allow all members to help with labeling, we created a step-by-step labeling guide that documents the intricacies of Roboflow tools and shows the annotating workflow. This resource allowed team members unfamiliar with Roboflow or YOLO to still contribute and significantly reduced the time required to annotate the entire dataset, while maintaining labeling accuracy.

II. MAPPING PIPELINE

The ASVs utilize LiDAR and camera fusion to map their environment and enable autonomous operation. The change from Python threading to ROS2 gave us access to many open-source libraries that work in conjunction with the onboard LiDARs and OAK-D cameras. Prior to integration, we tested both embedded systems separately to ensure functionality using ROS2 bags to collect data and Foxglove to visualize the topics output.

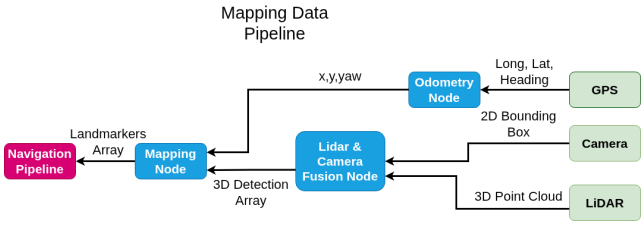


Fig. E.2. Mapping Data Pipeline

In order to generate a global map, we utilize GNSS position and heading and convert them to planar x, y, and yaw parameters using an odometry node. This provides global localization of the ASV and enables all sensor detections to be transformed into a consistent global reference frame.

The OAK-D LR camera outputs detections consisting of a class label (what that object is) and a 2D bounding box in image pixels, while the LiDAR outputs a list of 3D points. Since we're working with two different dimensions, 2D for the camera and 3D for the LiDAR, we must geometrically transform the data in order for it to be merged. This transformation projects LiDAR points into the camera frame, allowing depth information from the LiDAR to be associated with camera detections.

To account for potential sensor timing delays, the camera and LiDAR data streams must be synchronized. We utilize the ROS2 ApproximateTimeSynchronizer to pair camera detections with the closest in time LiDAR scan. This allows the data output to be in sync and ensures that the 2D bounding box from the camera is paired with the closest in time LiDAR scan, so the depth is accurate.

The LiDAR point cloud is first converted into an N×3 XYZ array. Each 3D LiDAR point is then projected onto the camera image using the known intrinsic and extrinsic calibration parameters. This results in pixel coordinates for each LiDAR point along with a corresponding depth value. Points that are outside the camera field of view (FOV) are discarded by bounding them only to the image dimensions.

For each camera detection, the bounding box defines a region of interest in the image. The node queries the projected LiDAR points that fall within this region and selects the closest depth measurement, corresponding to the nearest surface observed by the LiDAR. This results in an estimated distance of the object consistent with what the camera is seeing and yields a 3D detection in the form of (x,y, depth) relative to the ASV.

The mapper node takes these 3D detection positions and transforms them into a global position using the ASV's current odometry estimate. To prevent duplicate landmarks from repeated detections, newly observed landmarks are compared with existing ones and discarded if they fall within a predefined similarity threshold. Kalman filter smoothing is applied to each landmark to mitigate sensor noise, detection jitter, and pose estimation error. The Kalman filter allows each landmark's position to be refined over time, resulting in a more stable and consistent map. Finally, the mapper node publishes an array of all known positions of the landmarkers that is passed downstream to the navigation pipeline and used for autonomous decision making and path planning.

Ⅲ. SIMULATION

The team created an omnidirectional land robot in Gazebo to accurately simulate the holonomic drive of our ASVs, equipping it with the same sensors as the real vehicle and modeling sensor noise and range limitations based on vendor datasheets to ensure realistic performance. The simulation also integrates computer vision as a combined detection-and-actuation system, allowing the team to test mission logic under controlled conditions. Last year, strong sun glare caused unstable detection during the Navigation Channel task, making it difficult to determine whether failures were due to the vision system or the mission logic. By using simulation, we can isolate these factors and develop stronger mission logic assuming reliable detection, while also being able to introduce controlled detection errors to evaluate system robustness. This approach accelerates development, improves

reliability, and enables faster iteration before deploying to the physical vehicle.

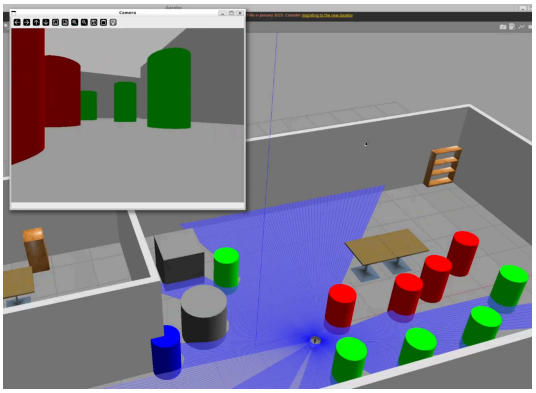


Fig. E.3. Navigation Channel simulation

IV. AUDIO DETECTION

Trial	Positive Accuracy	Negative Accuracy	Runtime (µs)
1	99.9%	100%	213
2	99.9%	100%	249
3	100%	100%	220
4	100%	100%	136
5	100%	100%	142

Table E.1. Accuracies and per-sample runtimes for the SVM system for the 1000 Hz, 1 blast tone. Each trial used 1,000 white and pink noise-injected samples for training, with different datasets generated for each trial. Different test data were used for each trial.

The team favored an approach similar to dual-tone multi-frequency (DTMF) detection [8]. The ASV's onboard microphone is constantly active and recording the ambient environment, processing this through a tone-detection system. This system relies primarily on a combination of Fast Fourier Transforms (FFTs) and One-Class Support Vector Machines (SVMs). SVMs are less commonly used for tone detection than other methods like the Goertzel filter; however, the team decided to use it due to their ability to detect periodic brightness changes in noisy environments [9]. An FFT is applied to the audio data to isolate the frequency domain. The FFT-processed data is sliced into a small frequency window, then normalized and passed to an SVM, which was trained on a dataset of tone-containing samples, with added artificial noise to increase robustness. This noise consists mainly of white and pink types, reflecting the relevance of those types in the 600-1200 Hz frequency range [10]. A single SVM trained with one tone is deployed and discerns if the tone has been heard. High-level logic then determines if the tone is pattern A or B.

Appendix F: Teensy Integration

I. INTRODUCTION

Arduino Uno couldn't handle control messages from Jetson faster than 1.5 Hz due to its computational limitations. When we have sent messages faster than 1.5 Hz to Arduino Uno, the thrusters would simply not move, and the ESCs would begin to sound the alarm. Therefore, we pivot to Teensy 4.1 for faster control, GPS message parsing, and GPS & IMU sensor fusion.

II. UNIT TESTING

After understanding the requirements on the Teensy, we follow the systems engineering V principle, which breaks down requirements into detailed action items. We conducted unit testing on each functionality: Communication with the GPS module and IMU; Communication with Jetson; and Motor Mixing.

We tested the motor control loop frequency first because it's fundamental to our navigation capability. We wired the Teensy to the Multiplexer, allowing us to safely test the controls with a soft kill from the controller. After testing for loop rate ranging from 1 Hz to 80 Hz, we concluded that Teensy 4.1 with our loaded code can handle inputs coming in at least 50 Hz, which meets our minimum 10 Hz requirement for a real-time control loop (Fig. F.1.).

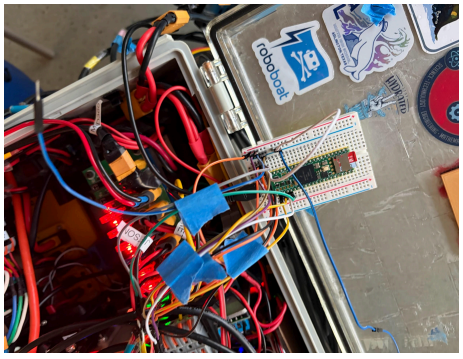


Fig. F.1. Test bench for motor control

We tested the ability to read from the Adafruit BNO085 9-axis IMU using the example code provided from the library (Fig. F.2).

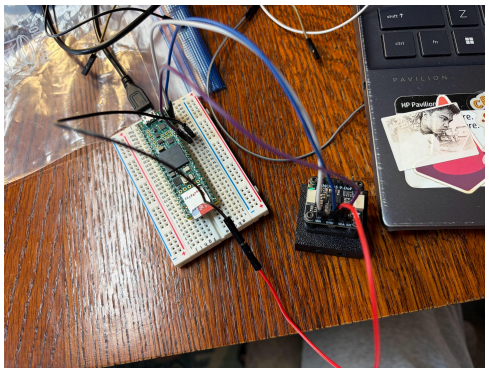


Fig. F.2. Physical setup

We created a test bench for our differential GPS by placing two GNSS receivers on the ends of a 1-meter aluminum bar, allowing us to take the receivers to open space to verify the system's accuracy and reliability after fusing GPS and IMU (Fig. F.3.).

Fig. F.3. GPS unit test setup

III. INTEGRATION

After verifying all functionality on a test bench, we designed our own electronics layout on perforated boards. Under the provision of making minimum changes to Barco Polo and the electronics difference between Barco Polo and Crusader, we designed two different perforated boards for the two boats. Barco Polo does not have an IMU, and the GPS module is directly connected to the Jetson. Therefore, the Teensy's responsibility is solely to receive motor commands and perform motor mixing.

We fused the IMU and GPS module to Teensy with a more compact and modular design to minimize exposed wires for sensor

communication stability. We did not use the Teensy to power the GPS module because the module requires 5V, while the Teensy only has a 3.3V output pin.

Appendix G: MonkeyBot

To train members to acquire relevant knowledge in mechanical, electrical, and software systems tailored to the RoboBoat 2026 mission, our mentor, Colin, designed an autonomous RC car (MonkeyBot) whose main function is to navigate between red and blue bricks (Fig. G.1.). A 12V NiMH battery powers the Raspberry Pi 3, the Arduino microcontroller, and the motor drivers via a terminal block. A separate battery powers the motors to eliminate voltage drops due to the large current draw from operating the motors. Computer vision enables color and object detection to identify colored bricks, sending information to our RPi, which controls our Arduino (Fig. F.3.). All members created the MonkeyBot and ran their own computer vision script, which taught them basic mechanical, electrical, and software skills that they now use in RoboBoat.

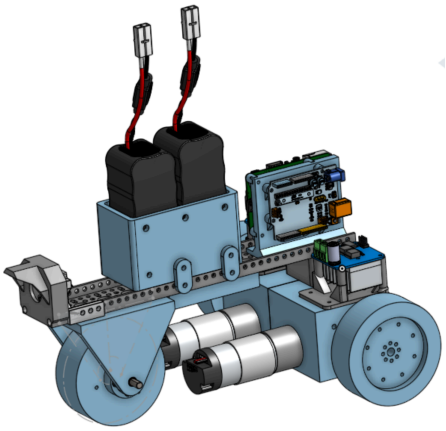


Fig. G.1. CAD model of the MonkeyBot

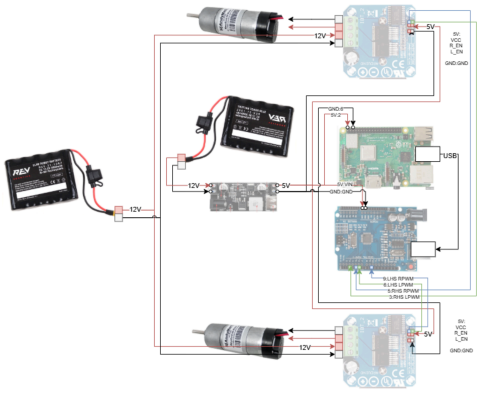


Fig. G.2. Electrical diagram of MonkeyBot

Appendix H: Systems Engineering Approach

I. SYSTEMS ENGINEERING V PROCESS

Team Inspiration has been implementing the Systems Engineering V process (Fig. I.1.) since our first RoboNation competition in 2019. As new team members join, the expectation of the engineering life cycle understanding has been raised to the industry standard. We are taught to become problem solvers. We execute requirement analysis and prepare design reviews based on previous competition performance. We conduct tests and development in a fast, repetitive manner. The end product goes through verification and validation to ensure it is built according to the plan. The process focuses heavily on self-awareness and reflection at every stage. The cycle is then repeated based on lessons learned.

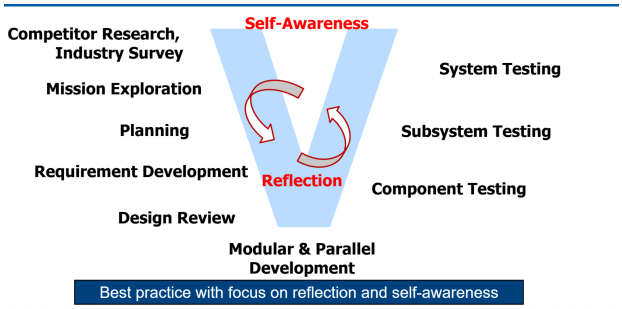


Fig. I.1. Systems Engineering V process

II. COMPETITION RESEARCH

After a competition, it is vital that our team reflects on the technical processes and designs of the top teams. We analyze how our design compares with others in order to learn more. In addition, competing ASVs since 2022 have given the team a basis of knowledge of how our current ASV performs.

III. MISSION EXPLORATION

All RoboBoat tasks are divided into three capability levels: core, advanced, and disruptive. Last year, our goal was to complete all tasks for the points. Our ASV completed the *Evacuation Route & Return, and Navigation Channel.* This year, our goal is to attempt disruptive capabilities for all tasks. With new capabilities of LiDAR-camera vision, better GPS, and a new Teensy microcontroller, we gain more confidence to complete the missions.

IV. PLANNING

The onboarding process began in September. All members worked on MonkeyBot to gain fundamental knowledge of mechanical, electrical, and software processes. Weekly meetings led to modest progress within the first three months, which we accounted for in the planned schedule. The last three months have focused on mechanical and software development and testing.

V. REQUIREMENT DEVELOPMENT

Systems requirements for our ASVs were derived from the RoboBoat competition missions and scoring guidelines. The requirements were then broken down into hardware, software, test, and operation. Based on the requirements, we built two ASVs that have already tested advanced IVC capabilities that will help us improve our score from last year. Our development for the ASV begins with assessing how it will complete the tasks at hand.

VI. SELF-AWARENESS

Most of our team started with no knowledge of RoboBoat. Therefore, we prioritize how to further develop the successful systems that we used in the previous year. Due to busy schedules and a long learning process, the team spent the initial weeks understanding the ASV before being able to make improvements on Barco Polo.

VII. DESIGN REVIEW

We completed multiple Design Reviews throughout the process. In the first month of the process, we constructed a Preliminary Design

Review discussing the schematic of last year's design, where we could easily edit electrical schematics, mechanical CAD design, and new software development.

VIII. MODULAR AND PARALLEL DEVELOPMENT

With the manufacturing of two ASVs, our team developed algorithms on Barco Polo during the construction of Crusader.

IX. TESTING

We prioritize testing as this is where many shortcomings come to fruition. Here, the V system is enabled as our team reworks through the failures and goes back to where systematic errors can be resolved. Our remote team members utilized simulations to develop object detection that local members were then able to water test.