Patty Murphy - Iceberg ASV

Roboboat 2026: Iceberg ASV

Brooklyn Watkins, Jenna Blackmore, Katie Yanchus, Lia Dumersque Correa, Cameron Taveroff, Ty Freda, Paul Beaudoin, Harry Ngo

Iceberg ASV, Memorial University of Newfoundland, St. John's, Newfoundland and Labrador, Canada
Submitted: January 14, 2026

***Abstract*— This Technical Design Report presents the design of Patty Murphy, Iceberg ASV's autonomous surface vessel for the 2026 RoboBoat Competition. The platform emphasizes reliability, safety, and autonomous performance through a redesigned hull, improved power management using a custom thruster power board, and a modular ROS 2–based software architecture. Enhanced perception and navigation systems enable vision-based object detection and reactive waypoint navigation.**

I. Competition Strategy

Iceberg Autonomous Surface Vehicle's (ASV) primary goal for the RoboBoat 2026 competition was to enhance system reliability. This included designing a robust system capable of operating in water with minimal issues, as well as developing dependable perception and navigation systems to ensure consistent performance in navigating paths and avoiding obstacles.

A. Software Architecture

All navigation tasks require similar functionalities, such as obtaining the ASV's position, receiving the object detection results, accessing parameters, and sending waypoints. To reduce code duplication, any common functionality across tasks was consolidated into the Main Task Class.

Each class for the navigation tasks inherited common functionality from the Main Task Class, along with specific task functionality, as shown in Figure 1. This structure reduced code complexity and duplication, unified the code base, and enabled quicker implementation and debugging of new tasks.

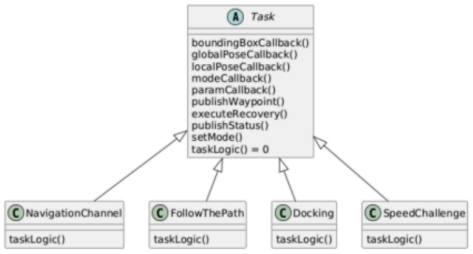

Figure 1: Task Class Tree

The task classes are implemented as lifecycle nodes, a ROS 2 concept where nodes have four possible states [1], as illustrated in Figure 2. A controller was created to manage the lifecycle of each task. The controller configures a Task Node by putting the node in the Inactive state. Once the node is initialized, the controller activates it and waits until it receives a finished signal before bringing the node to the finalized state, then destroying the node. All tasks go through the same process until the controller activates the final Return To Home task. Using lifecycle nodes ensures that unused nodes do not use computing resources and that only one node interacts with the flight controller at a time.

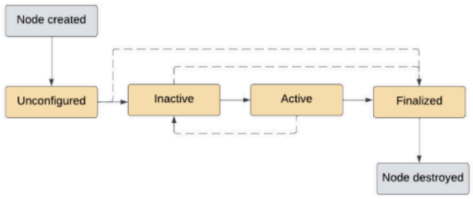

Figure 2: Lifecycle Node Flow Diagram

B. Path Planning and Navigation

ArduPilot firmware [2] and Pixhawk 6C flight controller [3] are used for navigation. This format has consistently provided the ASV with reliable waypoint navigation in local or global coordinate systems.

The new vision-only system operates by identifying and placing a bounding box around the target using its perception system, and then generating waypoints based on the target's horizontal bounding box position. The horizontal positions of bounding boxes are converted to angles, which are used to set a direction for waypoints as shown in Equation 1, where *fov* is the camera's field of view, *pixel* is the x pixel location of the center of the bounding box, and *resolution* is the x resolution of the camera.

$$a = \frac{\pi}{2} + \text{fov} * \left(\frac{1}{2} - \frac{\text{pixel}}{\text{resolution}} \right) \quad (1)$$

As bounding boxes are identified, waypoints are continuously generated and sent to the navigation system, as shown in Equations 2-5, where *a* is the waypoint angle, *d* is the waypoint distance, *h* is the ASV's heading and *xWP*, *yWP* are the resulting waypoint coordinates.

$$x_{\text{rel}} = d\cos(a) \quad (2)$$

$$y_{\text{rel}} = d\sin(a) \quad (3)$$

$$x_{\text{WP}} = x_{\text{ASV}} + x_{\text{rel}}\cos(h) - y_{\text{rel}}\sin(h) \quad (4)$$

$$y_{\text{WP}} = y_{\text{ASV}} + x_{\text{rel}}\sin(h) + y_{\text{rel}}\cos(h) \quad (5)$$

The ASV does not need to reach these waypoints; it just needs to head in the waypoint's direction until a new waypoint is sent. The path planning system does not perform global path planning; instead, it reacts to each bounding box as soon as it receives it.

C. Task Packages

1) *Navigation Channel:* The boat searches for red and green markers. If both a red and green marker are found, the ASV sends a waypoint between the two markers. If only a red or green marker is identified, the ASV sends a waypoint to the right of a red marker or, conversely, the left of a green marker. If the boat detects multiple red or green markers simultaneously, the ASV reacts to the closest marker in view. The ASV sends waypoints until no more buoys are found, signalling the end of the task.

2) *Follow the Path:* The ASV uses the same logic as the Navigation Channel to complete Follow the Path, but additionally searches for yellow buoys and navigates around them. To count the yellow buoys, the perception system monitors the size of each yellow buoy, which increases as the boat moves towards it. If the buoy disappears or becomes significantly smaller (a new buoy is identified), the ASV has passed a buoy, increasing the buoy count. Once no more buoys are identified, the boat has completed the task, and the buoy count is displayed on the shore computer's screen.

3) *Speed Challenge:* The ASV enters the first set of holding bay gates using the same logic as the Navigation Channel, but it enters station-keeping mode once the boat identifies the second set of gates. It stores its position inside the holding bay. Once it detects a green buoy, it navigates to the right of the blue buoy until it can no longer perceive the buoy, then drives to the left to navigate around it and finally returns to the stored position inside the holding bay. The boat employs the same strategy in "Follow the Path" to count and report the black buoys.

4) *Docking:* The perception system is trained to identify the docking bay shapes and colours, and will be trained on data during the

competition to identify occupied docking bays. The boat will navigate toward the unoccupied area with the correct shape and colour, and the LiDAR will be used to identify when the boat is inside the bay before reversing out.

D. Task to Task Navigation and Recovery

After the ASV completes a task, it moves to the start position of the next task until it can identify task markers and drive towards them. If the ASV gets lost during a task (cannot identify any targets within a set period), it returns to the actual start position recorded earlier and re-attempts the task. If it gets lost a second time, it abandons the task and proceeds to the start position of the next task.

II. Design Strategy

A. Design: Hulls

For 2026 new hulls were designed and constructed. The main goals for the new hull are the following. First the hulls must be compatible with the 2025 bridge deck. This means that the attachment points for the bridge deck must be of the same type and location as on the 2025 hulls. Second, the hulls must maintain the same maneuverability of the previous hulls. To achieve this the hull must be of a similar dimensions to the 2025 hulls. Third, one issue with the previous hulls was their tendency to pitch during acceleration due to the pitch moment applied by the thrusters. The vessel must have adequate pitch stability to ensure that the vessel will not pitch excessively or pitchpole.

The following construction method was used. First a 3D print plug of the hull was printed out of PETG using an extremely low 3.5% infill. Due to the size limitations of the available Bamboo X1C the hull needed to be printed in 5 sections then glued together with epoxy. The 3D printed plug was then wrapped in fiberglass using an epoxy resin wet layup. For this method the fiberglass provides the strength of the hull while the inner plug provides the shape and water tightness. A cubic infill was chosen as it is a closed cell so if water penetrated the fiberglass into the plug it would not fill with water.

Table 1: Target Design Specs

Length (inches)	42
Beam (inches)	7
Total Displacement (lbs)	35lbs
Per hull Displacement (lbs)	17.5lbs
LCB from bow	58%
Freeboard (inches)	4

The target displacement of the hulls was slightly increased from 33 to 35lbs as the new hull construction is slightly heavier. The length was increased compared to the previous hulls by 6” from 36 to 42 inches. This increase was made to increase the stability of the hull and decrease the wave drag. The increase was small to not sacrifice too much yaw maneuverability and the addition of rocker as opposed to a flat hull further decreases the yaw resistance. The max beam of 7” is carried all the way aft to add additional stability and make construction easier. The new bow is longer and finer than the previous hull increasing the efficiency of the hull. The hull freeboard was increased to 4” to ensure adequate clearance of the bridge deck to the water's surface.

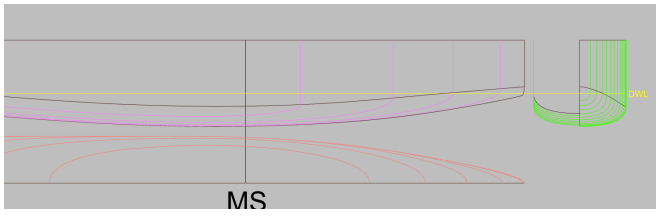

Figure 3: Lines Plan

One of the challenges faced by previous vessels was the fragility of the hull-bridge deck and hull motor attachment points. The two possible hull construction methods, XPS foam wrapped in fiber glass and very low infill 3D printed PETG wrapped in fiberglass. Both of these methods make embedded hardware difficult. The

fiberglass is too soft and thin to tap threads therefore the hardware must be embedded in the foam or PETG. Due to the necessity of low weight, both of these materials are fragile and any hardware embedded in them is at risk of being damaged during operation or by overtightening. This was learned the hard way with a previous hull that was destroyed when a hull-bridge deck bolt was overtightened, ripping the anchor out of the hull.

The solution chosen for this problem is to imbed wooden blocks into the hull. These blocks are glued into their hull plugs with epoxy covered with fiberglass along with the rest of the hull. Metal bolts are embedded into the top of the hull for the bridge deck and coupling nuts are embedded into the bottom so that the thrusters can be bolted in. Wood was chosen as it is light while still being significantly stronger than the low infill 3D printed hull. Wood also bonds well to epoxy allowing for secure attachment to the hull. By choosing wood the concentrated force on the metal hardware can be distributed to the rest of the hull. The aft bridge deck bolt is positioned vertically above the motor mounts and a single wooden block passing through the entire height of the hull attaches to both. By having the wood pass through the entire length of the hull the moment applied by the thrusters is transferred to the stronger fiberglass skin on both sides of the hull reducing the load on the weaker hull plug.

The propulsion system consists of two Blue Robotics T200 thrusters [4], which have been used since the team's first competition. Each iteration of the vessel has required a new propulsion design to address performance issues. This year, a focus was put on the thruster placement about the length of the vessel, seeking to minimize cavitation and optimize bow rise.

Thruster attachment difficulties led to catastrophic failure at the 2025 competition. This year, the team redesigned the thruster-to-hull attachment method. Previously welded vertical bolts were embedded in the hulls. Now, wooden blocks have been nestled between two of the 3D printed segments, and covered with fibreglass as per the rest of the body. This provides a location for the thrusters to be strongly attached to the hull, without sacrificing structural integrity.

Thruster Specifications	Thrust @ 20V	Current Draw @ Max Thrust	Power Draw @ Max Thrust
T200	5.05-6.7 kg	32 A	645 W

Figure 4: T200 Specifications

C. Design: Bridge

Testing confirmed that this change was effective and thus was not altered for the 2026 competition.

E. Electrical: Battery Power Management

Iceberg ASV has modified the power architecture since previous Roboboat competitions. In the past, the entire ASV electrical system was powered using three 4-cell Lithium-Polymer batteries configured in parallel. This setup proved to be a constraint to the run-time length, averaging twenty minutes per testing session.

This year, the setup consists of two Lithium-Polymer batteries in parallel for the thrusters and one battery for all other components. The decision to allocate two batteries for the thrusters is due to the high power consumption. Now that the thrusters have dedicated batteries solely for powering them, the run-time length for the entire system has increased, averaging one hour per testing session. The increase in system run-time for the boat has allowed for more effective testing periods.

F. Electrical: Thruster Power Board

This year, the Iceberg ASV team designed a custom Printed Circuit Board (PCB) to improve the vessel's power system management. The thruster power board accepts a 14.8 V supply from one of the lithium-polymer batteries used to power the ASV. The board incorporates four N-channel MOSFETs connected in parallel, enabling the safe and controlled operation of two

electronic speed controllers (ESCs) with a combined current draw of up to 50 A.

An onboard ESP32 microcontroller is used to generate control signals for switching the N-channel MOSFETs and regulating power delivery to the dual ESCs. The ESC output connectors are equipped with flyback diodes to protect the circuit from high-voltage transients caused by inductive loads. Overcurrent protection is implemented using a high-speed current-sense amplifier with an integrated comparator, which monitors current through a sense resistor.

III. Testing Strategy

A. Outdoor Testing

From September to December, IcebergASV worked in iterative cycles, developing and refining our designs through outdoor water testing every two weeks. This allowed the team to analyze the ASV's performance and response to obstacles and forced inputs. Many issues were discovered and resolved using this approach, including thruster sensitivity, PID tuning, and object detection.

1) *System Integration:* Testing is conducted collaboratively across the design, software, and electrical teams. This integrated approach enables real-time feedback on areas requiring refinement or redesign, including camera and sensor placement, platform stability during data collection, and overall system integration. In-water and extended runtime testing also provides critical insight into the performance and robustness of supporting components—such as emergency stop mounts, camera positioning, and Wi-Fi antenna mounts under real operating conditions.

2) *Software Strategy:* After previous competitions, the ASV's computer was upgraded from an NVIDIA Jetson TX2 to an NVIDIA Jetson Orin [5]. In addition to faster processing speeds and improved support, this also enabled the team to work in Ubuntu 22.04 [6] and upgrade from the ROS 1 framework to ROS 2 [7]. This allowed the code to be rewritten with organization and flexibility. While the migration to Ubuntu 22.04 and ROS2 was effective, it required a high volume of testing to ensure functionality was maintained and improved upon.

The software team debugged and verified program logic and tuned parameters in the water. The team set up a course, and varied individual parameters to optimize the ASV's performance. Some parameters tested include: the number of object detection frames to collect before generating a waypoint, the angle between the target and waypoint, and the frequency of sending waypoints.

Figure 5: Fall Testing Course Set Up, St. John's, NL

Testing involved having the ASV complete five runs, varying a parameter, and then repeating the process with another five runs. Performance would be evaluated both visually and through the viewing of recorded and live data. The significant amount of testing conducted over the Fall has resulted in reliable autonomous performance, ensuring that we can set up quickly for competition runs, debug issues easily, and work effectively as a team.

3) *Navigation:* The Pixhawk 6C [3] features several control systems that ensure the ASV navigates smoothly and accurately. In preparation for the competition, each control system was fine-tuned through on-water testing, with PID parameters adjusted based on real-time feedback. The tuning process involved putting the ASV through various maneuvers while monitoring

real-time plots of the actual versus desired performance.

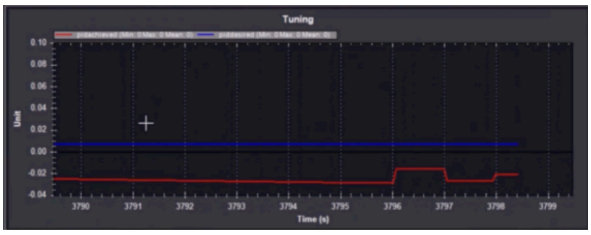

Figure 6: Real-Time PID Feedback

For instance, to tune the propulsion controllers, the ASV was RC-driven in straight lines with both gradual and rapid throttle changes. To fine-tune the GPS navigation controllers, GPS waypoint missions were set up to test how well the ASV adhered to its expected path. These waypoint missions were designed to simulate the types of maneuvers the ASV would encounter during the competition. For example, a waypoint mission shaped like a box was used to assess the ASV's ability to handle sharp 90-degree turns, while a figure-eight mission helped refine its ability to execute smooth, gradual turns.

Figure 7: Example Path

4) *Object Detection Fine-Tuning:*

A script was created to automate image collection, to create a dataset for model training. To reduce the model's sensitivity to brightness and colour saturation, and to mitigate the impact of motion blur, the following transformations were applied.

- Saturation: +/- 25%
- Brightness: +/- 15%
- Blur: 1px

During in-water object detection testing, it was concluded that these changes enhanced the overall quality of the object detection model.

B. Testing Tools

1) *Gazebo Simulation:* The team used Gazebo [8], a physics simulator that can be used with ROS, to verify program logic. This year, the team conducted less testing in Gazebo compared to the previous years. The team focused on in-water testing and used Gazebo as a simple "proof of concept" verification tool.

2) *YOLO Detection Visualizer & Debug GUI:*

The team developed several debugging and visualization tools to aid with testing. Previously, we had no way of viewing what the ASV was doing in software logic other than to view ROS topic data in the terminal. This was time-consuming, hard to read, and limited. The camera generated excessive data for transmission over the telemetry system, so a YOLO Detection Visualizer was created to visualize objects identified by the perception system. This provided us with insight into what the ASV was reacting to in real-time.

A debug GUI was created to display data commonly viewed in the terminal. This made the data much easier to read, leading to more effective debugging. Behaviour and Search Statuses were also added for responding to the ASV state. The right side displays node statuses. Active nodes are denoted by a green dot, and inactive nodes are denoted by a red dot. This identifies if the proper nodes are in use or if a node has an unexpected shutdown.

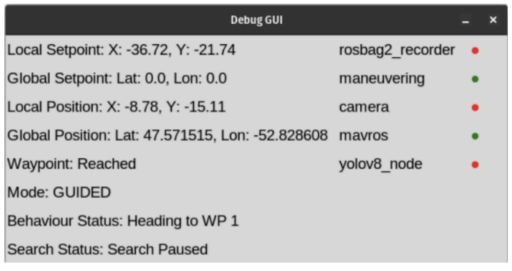

Figure 8: Debug GUI Example

Acknowledgements

Iceberg ASV would like to thank the many people who helped support the team's endeavours. John Walsh has encouraged and supported Iceberg ASV through mentorship, support, and resources from Memorial University's Student Design Hub. The team would also like to thank our sponsors: Genoa Design International, Solace Power, IEEE, COLAB, Angus Bruneau LIFE Awards, PEGNL and Memorial University Student Design Hub.

Bibliography

[1] Open Robotics, "ROS 2 Node Lifecycle." [Online]. Available: https://design.ros2.org/articles/node_lifecycle.html

[2] ArduPilot Community, "ArduPilot Firmware Documentation." [Online]. Available: https://firmware.ardupilot.org/

[3] Holybro, "Pixhawk 6C Hardware." [Online]. Available: https://holybro.com/products/pixhawk-6c?srsltid=AfmBOoqK9DrGG1WXXgcP1enQVosnebK4D2kMOArS6BMG9UTUPr-PwD4d

[4] B. Robotics, "T200 Thruster." [Online]. Available: https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/

[5] N. Corporation, "NVIDIA Jetson Orin Platform." [Online]. Available: https://www.nvidia.com/en-us/autonomousmachines/embedded-systems/jetson-orin/

[6] Canonical, "Ubuntu 22.04 LTS (Jammy Jellyfish)." [Online]. Available: https://.ubuntu.com/jammy/

[7] Open Source Community, "ROS 2 GitHub Repository." [Online]. Available: https://github.com/ros2

[8] Gazebo Sim Community, "Gazebo Sim." [Online]. Available: https://gazebosim.org/home