Hunky Dory - Iceberg ASV

Roboboat 2025: Iceberg ASV

Madison Button, Grace Pearcey, Mackenzie Collins, Colin O'Driscoll, Emily Pike, Jocelyn Ralph, Katie Yanchus, Brooklyn Watkins, Michael Mccarthy, Robert Fudge

Iceberg ASV, Memorial University of Newfoundland, St. John's, Newfoundland and Labrador, Canada
Submitted: January 27, 2025

***Abstract*— This Technical Design Report outlines Iceberg ASV's approach to designing Hunky Dory for the 2025 Roboboat Competition. Hunky Dory was designed to improve reliability, functionality, and safety. Improvements from Iceberg ASV's previous ASV, included optimized hull size and deck space, improved remote kill switch control, and improved autonomous capabilities of the ASV with object detection and navigation.**

I. Competition Strategy

Iceberg Autonomous Surface Vehicle's (ASV) primary goal for the RoboBoat 2025 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 the common functionality from the Main Task Class along with specific task functionality, as seen in Figure 1. This structure reduced code complexity and duplication, unified the code base, and enabled quicker implementation and debugging of new tasks.

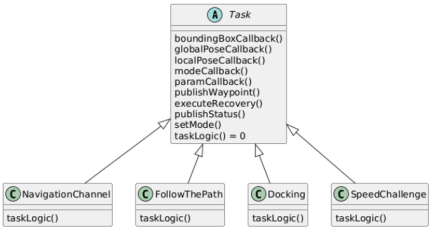

Figure 1: Task Class Tree

The task classes are implemented as lifecycle nodes, a ROS2 concept where nodes have four possible states [1], as shown 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.

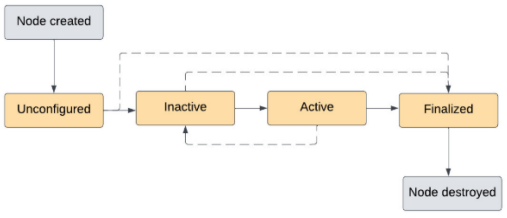

Figure 2: Lifecycle Node Flow Diagram

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 using a 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 3-5, 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 but instead 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 in between the red and green markers. If only a red or green marker is identified, the ASV sends a waypoint to the right of a red marker or, inversely, 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 uses 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 colors and will be trained on data during the competition to identify occupied docking bays. The boat will navigate toward the correct shape and color that is unoccupied, 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 heads to the next task's start position 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 heads to the next task's start position.

II. Design Strategy

A. Hulls

One of the most significant updates to the 2025 vessel design was the reduction in hull size and changes to their manufacturing process. The vessel's length decreased from 48″ to 36″ to enhance maneuverability, thereby reducing the required turning radius and vessel weight, see comparison in Figure 3.

	RoboBoat, Feb 2024	RoboBoat, March 2025
Length	48 in	36 in
Beam (*of one hull*)	7.5 in	7 in
Depth	5 in	5 in

Figure 3: Hull dimension comparison

Last year, the hulls were hand-shaped from foam, resulting in asymmetry, which increased drag. This year, the team implemented a precise manufacturing method using a combination of 0.5-inch thick 3D-printed inserts and insulation foam layers. This approach ensures uniformity along the hulls while retaining the foam's buoyant properties. A fibreglass coating was also applied to the foam hulls to further enhance durability.

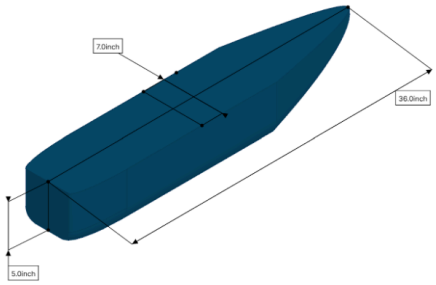

Figure 4: Hull design with dimensions

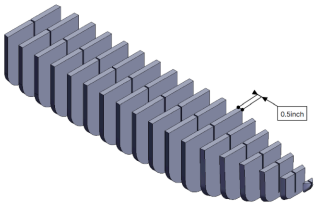

Figure 5: Printed insert model

B. Design: Propulsion

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.

During the 2024 RoboBoat competition, cavitation issues occurred due to the location of the thrusters, reducing the overall thrust. In previous designs, the thrusters were positioned at the center of the bridge and mounted on 3D-printed shafts. These shafts, were not long enough to keep the thrusters fully submerged in choppy outdoor conditions and increased the risk of entanglement with buoys and debris. The new thruster configuration mounts the thrusters directly to the underside of the hulls, ensuring they remain submerged. This design incorporates steel plates embedded in the hulls with epoxy. Four bolts welded to the steel plates allow for easy attachment of the thrusters, ensuring a stronger and more reliable mounting system.

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

Figure 6: T200 Specifications

C. Design: Bridge

Significant changes were made to the bridge to reduce the vessel's weight and enhance modularity. In previous years, the vessel used a high-density polyethylene (HDPE) bridge, which added considerable weight and was cumbersome to transport. A new bridge was designed for 2025 using right-angle aluminum bars. This design can be fully disassembled into smaller components, making it travel-friendly. The aluminum construction is significantly lighter than the HDPE bridge, improving the vessel's thrust-to-weight ratio and overall performance.

D. Design: Electrical Integration

To streamline troubleshooting for the electrical and software subsystems, improving accessibility

to the electrical boxes during testing was a priority. In previous competitions, removing the electrical boxes from the bridge was a time-consuming process, causing interruptions during competition when quick troubleshooting was essential. This year, the boxes are secured to the aluminum bar bridge using spring-loaded clasps, allowing for rapid removal and reattachment. This approach is significantly more efficient than the previous method, which required removing the entire bridge and unscrewing the boxes to access them.

E. Electrical: Power Management

In previous competitions, the electrical system was powered using three 4-cell Li-Po batteries configured in parallel. This setup proved to be a constraint to the run-time length, averaging twenty minutes per testing session. It was determined that the short run-time was due to the thruster's high power consumption.

This year, two Li-Po batteries in parallel are used for the thrusters and one battery for all other components. The dedicated power source for the thrusters has drastically increased our run-time to an average of one hour of constant use. The increase in system run-time for the boat has allowed for more longer testing sessions, ensuring that software can be tested as much as possible in the water.

F. Electrical: Housing Layout

The ASV electrical housing has been a challenge for Iceberg ASV since first competing at RoboBoat. The continual upgrades to the boat's electrical, mechanical, and software components have made maintaining an organized and reliable electrical housing difficult. The housing system used in 2024 ultimately led to messy electrical wiring, loose terminal blocks, and inadequate electronic layout that combined to make troubleshooting and set-ups intimidating.

To combat the electrical house problems, Iceberg ASV implemented a steel mounting plate, cable tray, and improved DIN rails for the electronics inside the housing for the 2025 competition. Organized wiring for the electronics reduced troubleshooting time and wiring mistakes, contributing to easy set-up. See Figure 7 for the enhancements to the electrical housing.

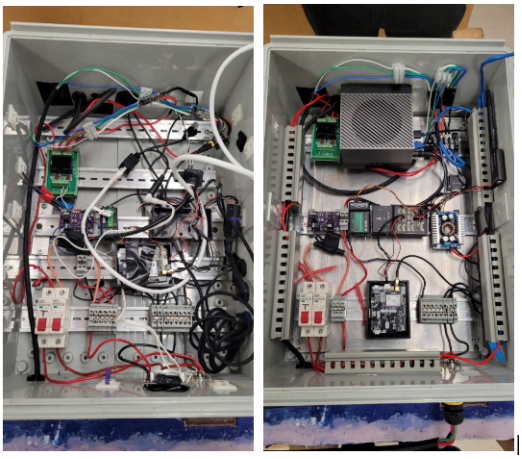

Figure 7: Before (left) and after (right) of electrical housing improvments

G. Electrical: PCB Re-spin

This year, Iceberg ASV improved upon the Pulse Width Modulation (PWM) Printed Circuit Board (PCB). The PWM-DC REV002, focused on improving reliability and ease of use for the 2025 competition. The goal of the PWM-DC REV002 is to control the remote emergency stop system. This is done by pulling a relay after a comparison between a set voltage and the average of the PWM signal. Alterations from the original design include the use of terminal blocks with screw terminals as opposed to the use of push-in terminals. To increase safety, a fuse was added to the 12V input for protection against shorts. Additionally, a status LED circuit was added to indicate board power and operation.

III. Testing Strategy

A. Outdoor Testing

From September to December, technical teams worked in iterative cycles, developing and building in the evenings and conducting outdoor testing in a pond every one to two weeks.

1) *System Integration:* Testing is completed collaboratively between design, software, and electrical. This integrated approach allows for real-time feedback on areas requiring improvement or redesign, such as camera and sensor placement, platform stability for data collection, and overall system integration. These tests also provide critical evidence about the performance of supporting components—such as e-stop mounts, camera position, and WiFi bullet mounts—under in-water conditions and extended runtime scenarios.

2) *Overall Software Strategy:* After the 2024 competition, the ASV's computer was upgraded from an NVIDIA Jetson TX2 to an NVIDIA Jetson Orin [5]. In addition to faster processing speeds and better support, this also allowed the team to work in Ubuntu 22.04 [6], and upgrade from the ROS1 framework to ROS2 [7]. This allowed 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, as shown in Figure 8, 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 8: Fall Testing Course Set Up, St. John's, NL

Testing involved having the ASV complete five runs, varying a parameter, and then completing another five runs. Performance would be evaluated visually and through viewing recorded and live data. The significant amount of testing over the Fall has resulted in reliable autonomous performance and ensures 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, see Figure 9 and Figure 10. The tuning process involved putting the ASV through various maneuvers while monitoring real-time plots of the actual versus desired performance.

Figure 9: PID Tuning Parameters

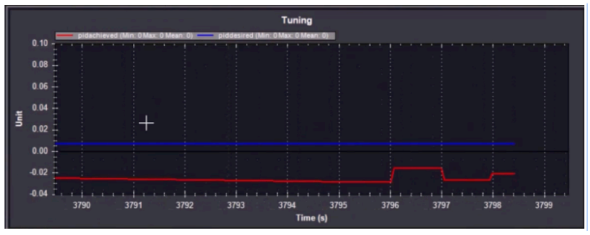

Figure 10: Real Time PID Feedback

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 (Figure 11) 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 11: 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 reduce 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 improved the overall object detection model quality.

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 previously. The team focused on in-water testing and just used Gazebo as a simple "proof of concept" verification tool.

2) *YOLO Detection Visualizer and 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 produced too much data to send over the telemetry system, so a YOLO Detection Visualizer was created to visualize objects identified by the perception system, as shown in Figure 12. This provided us with insight into what the ASV was reacting to in real-time.

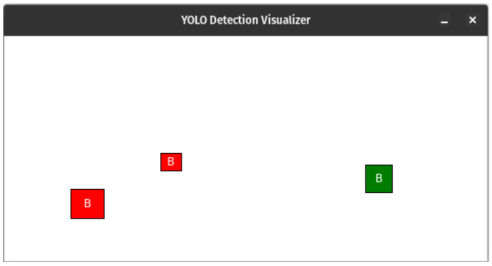

Figure 12: YOLO Detection Visualizer

Figure 13 displays the Debug GUI. Any data commonly viewed in the terminal was added to the list on the left of the GUI. This made the data much easier to read, leading to quicker debugging. Behavior and Search Statuses were also added, corresponding to the ASV state. The right side displays node statuses. Active nodes have a green dot, and inactive actions have a red dot. This helps identify if the proper nodes are being used or if a node has an unexpected shutdown.

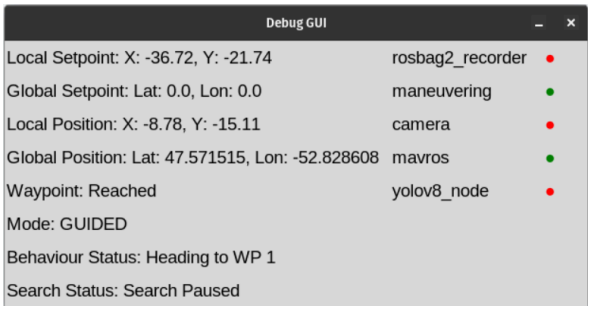

Figure 13: Debug GUI Example

IV. 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, Nasdaq Verafin, Kraken Robotics, Madsen Group, Colab Software, IEEE, C-CORE, PEGNL, Fastenal, Hakko, Central Law, and City of St John's.

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/autonomous-machines/embedded-systems/jetson-orin/

[6] Canonical, “Ubuntu 22.04 LTS (Jammy Jellyfish).” [Online]. Available: https://releases.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

,