RoboBoat 2026: Technical Design Report Georgia Tech Marine Robotics Group

Nicholas Lai, Marcus Agun, Niharika Anand, Sean Breton, Yanda Chen, Paul Dubrulle, Jason Hsiao, Srikar Kususmanchi, Ariana Litchford, Em Paul, Shrey Patel, Auryn Yamamura, Aaron Wu, and Sean Fish
Georgia Institute of Technology
Atlanta, GA, USA

***Abstract*—The tasks of the RoboBoat 2026 competition provide a challenging set of benchmarks for unmanned surface vehicles (USV), requiring autonomous systems to be designed with robustness against environmental noise. The tasks are closely aligned with the upcoming multi-domain RobotX 2026 challenge, motivating this year's entry to RoboBoat from the Marine Robotics Group at the Georgia Institute of Technology. The entry for this year leverages the BlueBoat Extended platform developed at the Aerospace Systems Design Laboratory (ASDL) as part of the Unmanned Collaborative Research Testbed (UCRT), a multi-agent field robotics testbed. The reconfigurable platform has been augmented with competition-specific safety systems including an emergency stop mechanism. The autonomy system leverages the LiDAR and dual depth cameras onboard the platform, utilized in conjunction with recent advancements in segmentation methods for adaptive object detection, a behavior-tree-based decision-making framework for rapid task composition, and ArduPilot-provided navigation and control. The team includes students trained through the Stinger Tug program, an interdisciplinary educational platform that provides hands-on experience in mechanical design, electronics integration, and ROS 2 software development. In this report, we describe our technical design as well as our competition strategy, including training, task-specific tactics, modular design philosophy, and testing methodology.**

Fig. 1. An ASDL BlueBoat Extended (left) and Stinger Tug training vehicle (right) encounter each other at a lake test for the Marine Robotics Group.

I. Introduction

The evolution of autonomous maritime systems increasingly requires a "System-of-Systems" approach to manage the complexities of multi-domain operations. The Georgia Institute of Technology's RoboBoat 2026 entry is built on a Blue Robotics BlueBoat provided by RoboNation and retrofitted into the Unmanned Collaborative Research Testbed (UCRT) architecture developed at the Aerospace Systems Design Laboratory (ASDL) [1, 2]. This BlueBoat Extended platform is designed to operate seamlessly across RoboBoat 2026 competition tasks and future RobotX Maritime Challenge deployments.

The UCRT framework applies DoDAF principles [3] to establish standardized operational views and interface definitions, enabling support for a wide variety of missions. These principles are reflected by the implementation of a backseat control architecture [4], wherein the frontseat (ArduPilot) manages safety-critical low-level control while the backseat (ROS 2) handles mission-specific autonomy [2]. Connected via MAVROS interfaces, this separation has dramatically accelerated development by decoupling competition-specific behaviors from vehicle control. Perception algorithms, navigation behaviors, and coordination logic can be modified without touching the underlying control infrastructure—critical for rapid iteration between RoboBoat navigation tasks and RobotX multi-domain coordination.

Our long-term strategy for competition development is to treat competitions as missions in the UCRT framework, with RoboBoat serving as a structured environment to develop baseline capabilities that transfer directly to the RobotX mission. Autonomy software for single-USV challenges—navigation channel traversal, obstacle avoidance, docking—integrates immediately with UAV and UUV coordination behaviors through standardized ROS 2 interfaces, requiring no architectural redesign.

This strategy is based on having a team with a strong educational foundation: the Marine Robotics Group's Stinger Tug training program [5] provides new team members with interdisciplinary experience in mechanical fabrication, PCB assembly, and ROS 2 development (Fig. 1). Students build small-scale educational USVs, gaining hands-on competency in a similar software environment (ROS 2, Gazebo simulation, Git version control) with similar hardware interfaces (GPS/IMU fusion, ESC control, sensor integration) to those on the competition platform. This training pipeline ensures that students can immediately contribute to RoboBoat development, prioritizing "Intelligence-on-the-Edge" perception and

planning over basic platform integration.

We utilize the UCRT framework for the capabilities it provides for adapting to new missions. Key technical capabilities of the testbed include:

1) **Rapid Sim-to-Real Deployment:** Containerized software enables seamless Gazebo-to-hardware transitions with declarative configuration for sensor packages and behaviors [2].
2) **Foundation Model Hardware:** The NVIDIA Jetson Orin Nano provides the necessary compute for a Segment Anything Model (SAM) implementation to provide zero-shot semantic segmentation of novel maritime obstacles, as well as other models such as YOLO and CLIP [6, 7, 8].
3) **Heterogeneous Sensing:** Dual OAK-D stereo cameras and Livox Mid-360 LiDAR provide 360-degree depth mapping, validated across multiple configurations in field tests [1].
4) **Cross-Domain Architecture:** ROS 2 interfaces support UUV and UAV asset management, establishing the foundation for RobotX coordination tasks.
5) **Resilient Communication:** Redundant wireless infrastructure using 2.4 GHz WiFi 802.11n for high-rate logging and 900 MHz WiFi 802.11ah HaLow mesh for command-and-control, ensuring telemetry continuity in non-line-of-sight conditions[9].

Section II outlines our competition strategy for RoboBoat and task-specific tactics. Section III details the design strategy emphasizing modularity and reuse. Section IV describes our testing methodology

II. Competition Strategy

The competition strategy for this year can be summarized as developing high-level core behaviors and action primitives which can be easily composed into task plans, with a primary focus on buoy gate traversal tasks.

The Georgia Tech RoboBoat 2026 entry is motivated by the recent alignment of RoboBoat and RobotX surface vehicle requirements and the recommendation of the Blue Robotics BlueBoat platform. This revealed an opportunity to leverage our access to an existing modified BlueBoat platform (Fig. 3) and utilize RoboBoat as an environment to develop core capabilities that are simultaneously applicable to RobotX and larger-scale maritime and multi-robot field experimentation. This announcement was made later in the RoboBoat competition season, and our strategy reflects this condensed development time frame.

There are three main factors that influence our competition strategy: task scope, testing, and time. Task scope is a major factor for success, with too conservative of a scope leaving unrealized potential and too broad of scope resulting in poor overall performance. Testing is important for reliability, but time spent testing is time taken away from development, which can sometimes be more valuable. Time is our main constraint for RoboBoat, especially with our late decision to participate.

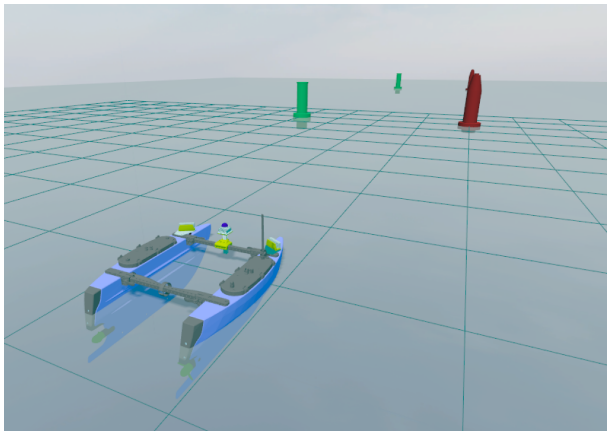

Fig. 2. Simulated BlueBoat Extended and Entry & Exit Gates Task in Gazebo (Evacutation Route & Return).

Rapidly increasing task scope in our recent experience results in architectures which quickly become complex and hard to scale. Examples of this can include large state machines in the best cases and spaghetti code in the worst cases. Testing is also important for ensuring reliability as code complexity increases, but tests alone cannot complete tasks. Given the time constraints for this competition, we decided the optimal trade to make was to reduce task scope in order to enable the development of a scalable code base while also enabling testing on the few tasks we prioritize. Considering our longer term goals at RobotX, this allows us to build up our capabilities while reducing our code complexity with intentional design in the long run.

As a result, the team's strategy is to build higher level functions relying on existing lower level capabilities. We prioritize reliability, generality, and architectural soundness over maximizing task coverage. We emphasize task composition, with tasks being handled with sets of behaviors and action primitives which are reusable across scenarios, but focus on a select few tasks during development of these core functions.

A. Task Decomposition and Requirements

Given the BlueBoat Extended platform's perception and navigation capabilities, we decompose several RoboBoat 2026 tasks to identify the required common core functions for core, advanced, and disruptive capability classifications.

Task 1 – Evacuation Route & Return: This mandatory qualification task (Fig. 2) requires autonomous navigation through two pairs of red and green gate buoys, beginning at least 6 feet before the first gate and exiting through the same gates at run completion. In order to fulfill this task, the perception system must be capable of identifying buoys and buoy gates, the decision-making system must be able to find the gates and determining a desired path, and the control system must be able to execute the path.

Core Requirements:

- Buoy detection and localization
- Buoy gate identification and localization

- Buoy discovery
- Collision avoidance
- Go to pose / Go through poses

Task 2 – Debris Clearance: This task simulates emergency lane navigation and debris field operations. The ASV transits a navigation channel defined by multiple red/green gate pairs, enters a debris field containing black obstacle buoys and color indicator buoys, interacts with indicators (red = avoid and report; green = circle and report), and returns through the channel.

Core Requirements:

- Buoy detection and localization
- Buoy gate identification and localization
- Buoy discovery
- Collision avoidance
- Go to pose / Go through poses

Advanced Requirements:

- Indicator classification
- Object location reporting
- Buoy circling

Disruptive Requirements:

- Robust buoy classification

Task 3 – Emergency Response Sprint: This speed challenge requires the ASV to pass through red/green gate buoys, circle a yellow buoy in the direction indicated by a color indicator (red = counter-clockwise, green = clockwise), and exit through the gates as rapidly as possible.

Core Requirements:

- Buoy detection and localization
- Buoy gate identification and localization
- Buoy discovery
- Collision avoidance
- Go to pose / Go through poses

Advanced Requirements:

- Indicator classification

Disruptive Requirements:

- Object color reporting

Task 5 – Navigate the Marina: This docking task requires the ASV to enter a marina environment and autonomously dock in the lowest-numbered available slip (1-3), indicated by green color indicators.

Core Requirements:

- Dock detection and localization
- Collision avoidance
- Go to pose / Go through poses

Advanced Requirements:

- Indicator classification

Disruptive Requirements:

- Number classification

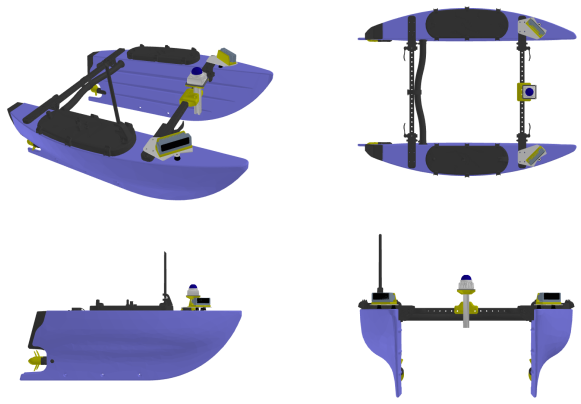

Fig. 3. Multiple views of the ASDL BlueBoat Extended platform rendered from the robot description package.

B. Cross-Task Capability Reuse

Across the tasks we have selected, they share many core requirements. These requirements, rather than the tasks themselves, will be the main development focus leading into the competition. These core requirements can generally be classified as either behavioral pipelines which constantly run throughout tasks processing information or actions which enable motions for accomplishing specific goals. These can be reused across tasks, ideally with only minor parameter adjustments being required. This design reduces integration complexity, accelerates testing iteration, and establishes a capability baseline directly transferable to RobotX multi-domain operations.

Based on the identified requirements, the capabilities which we will prioritize are:

- Buoy detection and localization
- Buoy gate identification and localization
- Buoy discover
- Go to pose / Go through poses
- Collision avoidance

III. Design Strategy

Our design strategy leverages the BlueBoat Extended platform (Fig. 3) developed for the UCRT project [1, 2]. The platform's intentional reconfigurability—featuring standardized bulkhead connectors, modular electronics bays, and extensible power management—enables rapid capability upgrades without compromising the baseline architecture. This approach preserves compatibility with broader UCRT research objectives while satisfying competition constraints, supporting multi-year development cycles and international transport logistics.

Due to the existence of the stable experimentation platform, we focus on the higher level software, designing perception and decision-making systems which rely on the existing control interfaces. Our hardware changes are intentionally minimized for competition compliance, both due to our strategic focus on higher level software as well as the need to return the

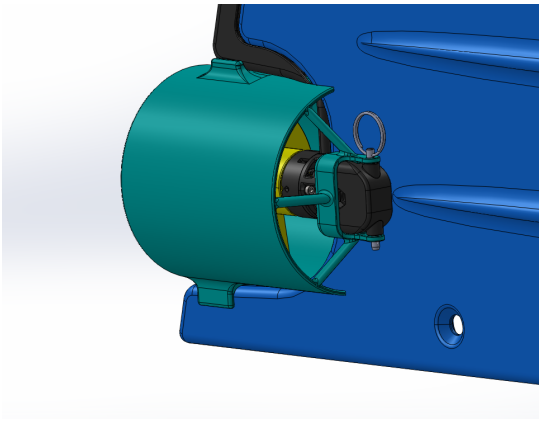

Fig. 4. Propeller shroud pictured on aft interior of BlueBoat hull.

vehicle to its original condition following the competition. We applied minimal competition-specific modifications to comply with RoboBoat 2026 safety requirements.

A. Hardware Modifications

1) Emergency Stop Integration: The paddle-accessible emergency stop system represents a pilot integration for fleet-wide deployment across UCRT platforms, with RoboBoat 2026 serving as the field validation environment. Mechanical integration leverages the BlueBoat's existing infrastructure: the exterior E-stop button mounts to aluminum crossbeams using standard M6 hardware compatible with the platform's T-slot mounting grid, while electrical connections route through pre-fitted bulkhead pass-throughs originally designed for sensor and power distribution expansion.

The E-stop circuit employs latching relays to selectively disconnect two of the three thruster motor phases at the ESC output, representing the minimum electrical intervention required to fully disable propulsion. This topology maintains compute and communication systems for diagnostic telemetry during emergency shutdown, enabling post-incident analysis and remote system status monitoring without requiring physical access to restart the vehicle's electronics. Custom electronics integration mounts on acrylic bay assemblies designed to fit within the BlueBoat's port and starboard hatch enclosures, providing structured component mounting while maintaining tool-free access for field maintenance.

The relay control logic integrates both local (physical button) and remote (900 MHz RC Link) activation paths, with latching behavior ensuring the E-stop state persists through power fluctuations or communication loss. All electrical modifications utilize the platform's existing bulkhead connectors, avoiding new hull penetrations and maintaining sealed integrity.

Along with the propulsion power relays, the system includes an independent E-stop status relay that provides an electrically isolated indication of the E-stop circuit state. The sensing relay output connects to a microcontroller, which reads it as a digital input to determine the E-stop status. It shares this information over serial in a protobuf packet along with an additional output to a light tower to communicate system state to nearby operators, supporting rapid troubleshooting during field testing.

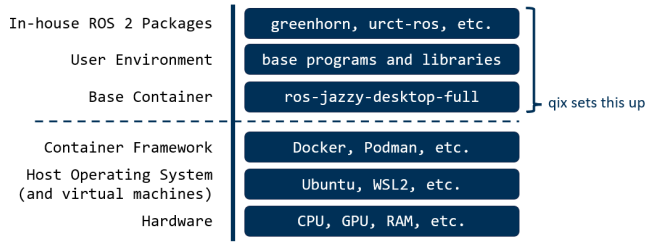

Fig. 5. Overview of the software stack.

Once verified reliable through competition field operations, this E-stop architecture will be standardized across all UCRT BlueBoat Extended platforms as a safety baseline, demonstrating the framework's capacity for minimally invasive fleet-wide upgrades. The dual activation mechanism establishes a safety template applicable to future multi-domain UCRT deployments where human proximity varies by operational context.

2) Propeller Shroud: The propeller shroud (Fig. 4) is intentionally designed for rapid mechanical reconfigurability via a low-profile mounting interface. A compact mounting bracket is permanently secured to the hull using the existing propulsion motor's mounting hardware, providing a robust anchor point without requiring additional hull structure. The shroud utilizes a sliding-fit geometry that meshes with the hull's contours to ensure a secure hold. Once aligned, a single quick-release pin locks the assembly in place, allowing for tool-less installation and removal. While the shroud provides propeller protection and improves robustness in confined or debris-rich environments, it may be removed during UCRT field operations to reduce the risk of marine growth accumulation and weed entanglement over extended deployments. For long-duration testing and multi-run experimentation, an unshrouded propeller simplifies inspection, cleaning, and recovery, improving overall system reliability. The ability to rapidly install or remove the shroud allows the platform to be tailored to short-duration competition runs or longer-term research deployments without requiring structural modifications, reinforcing the UCRT's modular, experiment-driver design philosophy.

B. Software Stack

The software stack, or the set of software tools and technologies which make up the base infrastructure for an application, is the bedrock upon which our robotics software is built. One major challenge encountered in previous years was the difficulty of initially setting up the software stack. Reducing the difficulty of reproducing the software environment was a major prong of our design strategy.

This year, significant effort has been invested into tooling which can semi-declaratively build our stack in a mostly

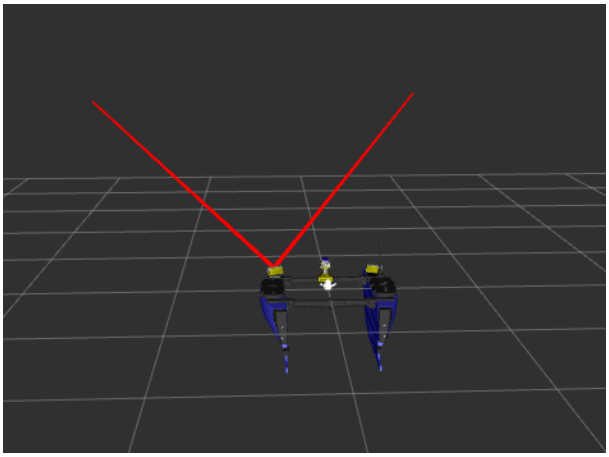

Fig. 6. Bearings of buoys based on segmentation-based perception are displayed in RViz.

reproducible manner. The Quickly Install X, or *qix*, build orchestration tool was created for this purpose. *qix* assumes an existing Linux base for the software stack with a containerization tool installed. Everything on top is built from a single toml file which describes all explicit system installations and automatically pulls source code from repositories.

The software stack built for this year is based on the 'ros-jazzy' container environment from the Docker official repository, with a set of base applications from the UCRT project, such as robot description and bringup packages. Our successor version of the 2024 RobotX application suite, Greenhorn 2, depends on the UCRT packages and provides our competition-specific capabilities. Greenhorn 2 consists of legacy RobotX code and also incorporates some code from the RoboSub Pontus suite, as we seek to deduplicate and reuse code as part of our overarching strategy towards RobotX.

C. Software Architecture

The software architecture is represented as a number of behavioral pipelines which run in parallel. The pipelines are categorized in accordance with the Sense-Think-Act paradigm [10]. The *Sense* pipelines process information from sensors which are more useful for the *Think* pipelines, which provide desired actuation to the *Act* pipelines. The framing of the architecture is intended to emphasize the parallelization of these processes and improve the ease of understanding for new members and encourage modularity in design and development, as opposed to a directed graph representation.

Each pipeline is dedicated to a particular function, with well defined inputs and outputs. This simplifies the creation of tasks and the design of unit tests.

The strategy used in implementation is to build up a single core pipeline in each category, and bring up supplementary pipelines once the core pipelines are established.

Sense - The core pipeline for sensing is the *visual perception* pipeline, which takes an input of an image and outputs a segmented version of the image. The implementation of this

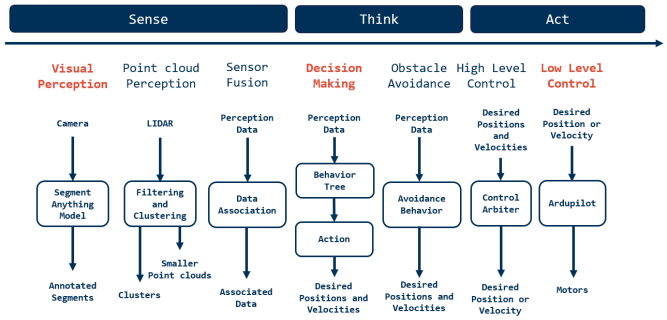

Fig. 7. Overview of the software architecture.

pipeline is centered on incorporating Segment Anything Models (SAM). We specifically utilize FastSAM for this [11]. The simulation environment provides segmentation directly from the camera, allowing for downstream development to occur in parallel to work on this core capability. Given functional segmentation and a camera model, we can determine bearings of objects for decision-making (Fig. 6).

The *point cloud perception* pipeline involves the use of filtering and clustering to better identify objects and for collision avoidance. The *fusion* pipeline fuses the visual perception and point cloud outputs.

Think - The core pipeline for thinking is the *decision making* pipeline, which consists of a behavior tree initialized at the start of a run. Behavior trees were chosen for ease of design and composition. We would like to easily adjust behaviors in competition, both to adapt decisions to changes in our base capabilities as well as to course conditions. Due to the compositional nature of behavior trees, we are also able to compose subtrees of task-specific behaviors for a full run of multiple tasks.

An *obstacle avoidance* pipeline may also be active in parallel to the decision making pipeline, in order to cover low level decisions that may be undesirable to include in the behavior tree.

Act - The *position and velocity control* is provided by the Ardupilot firmware running on the BlueBoat, which is accessible via MAVROS. This provides the majority of functionality for movement.

If multiple decision-making pipelines are used in parallel, a *high level control* arbiter must be active to fuse the decisions. In the past, a behavior-based architecture like the Distributed Architecture for Mobile Navigation has been used to great success [12].

IV. Testing Strategy

Our testing strategy is designed to maximize development velocity while minimizing risk to the physical BlueBoat platform, in alignment with our broader competition and research objectives. Following the strategic decision to focus on navigation-centric behaviors, the team leverages RoboBoat as a structured, repeatable environment to mature autonomy capabilities that scale directly to RobotX deployments. By emphasizing reliability, generality, and architectural soundness,

Fig. 8. Simulated version of the Supply Drop task in Gazebo. While it is not one of our primary tasks, simulations are created for as many tasks as possible.

testing is approached not as an isolated activity but as an integrated part of the development lifecycle.

A key component of this strategy is the use of containerized software deployment via our *qix* build orchestration tool. By packaging the full software stack, including perception, planning, and control modules, into reproducible containers, we are able to rapidly deploy updates to both simulated BlueBoat instances in Gazebo and physical vehicles in the field. This approach ensures that experimental modifications can be tested iteratively with minimal risk to hardware and provides a consistent baseline across multiple test runs.

A. Simulation Environment

The UCRT project includes the Lyoko simulation tools, which are an effort to improve the process of importing robot descriptions into simulation and enable similar processes in bringing up software both in simulation and real world environments.

In order to simplify the software suite, the simulator is configured to use Gazebo's graded buoyancy plugin, which allows for the simulation of autonomous surface vehicles. There are plans to reintegrate VRX-based wave plugins in the future as an option, to allow for multi-fidelity testing. The simulated worlds themselves are created for each task with a minimal set of entities for proving base viability per task without incurring additional computational cost. Buoys were imported into the simulation environment from older versions of VRX for some of the tasks [13].

The simulation of the vehicle itself also features options to support varied computational resources, with an option added to the UCRT's BlueBoat Extended description package to enable a segmentation camera in place of standard OAK-D images to avoid the need for a full segmentation pipeline.

These simulations do not currently have automated scoring functionality, but this is expected future work as we intend to adopt test-driven development practices as we approach RobotX.

B. Hardware-in-the-Loop (HITL) Development

HITL testing provides a controlled indoor environment to validate the full autonomy stack without requiring water access or GPS. Conducted in laboratory settings, these sessions verify that the software builds correctly on the target Jetson Orin Nano and that perception algorithms process real sensor streams from OAK-D cameras and Livox LiDAR.

The primary goal is perception validation against static test scenes, using printed buoy patterns, gate markers, and obstacle arrays. This isolates perception performance from navigation dynamics, enabling rapid iteration on object detection, segmentation, depth estimation, and inference parameters. Hardware-specific issues, such as USB bandwidth limitations or ROS 2 message queue backlogs, are exposed during HITL, allowing corrective measures before field deployment.

A secondary objective is software stack reproducibility. HITL ensures that the declarative *qix* build process correctly instantiates the complete environment on fresh hardware, catching dependency or configuration drift early.

Sessions follow structured protocols tracked in the team's Notion database[1], recording sensor setups, software versions, and anomalies. Metrics such as detection rates, false positives, and inference latency establish baselines for field testing. While HITL cannot replicate GPS navigation or water dynamics, it enables rapid, repeatable verification of perception and autonomy software on actual hardware, supporting efficient development and tuning of neural network models.

C. Continuous Field Testing

The George H. Sparks Reservoir at Sweetwater Creek State Park (Fig. 1) serves as a common testing site for the BlueBoats. Field tests at the reservoir verify the system performs as anticipated with environmental noise, including but not limited to waves, glare, and wind. By leveraging a common platform from the UCRT, the competition entry benefits benefits from the improvements made as a result of all previous tests. A database contains reports of all field tests as well as maintenance records. The BlueBoat Extended platforms have been fielded 13 times as of time of writing, with more testing planned through January and February.

By leveraging existing procedures and checklists, testing is made more reliable, and the competition-specific field tests, as field tests of the UCRT, will also contribute to further improvements in the deployment process.

ACKNOWLEDGEMENTS

The RoboBoat team thanks the Aerospace Systems Design Laboratory at the Georgia Institute of Technology for providing facility access, technical mentorship, and a competition platform for this year. We also thank the Georgia Tech Student Government Association for providing funding for development and travel. We extend our gratitude to Blue Robotics for the foundational BlueBoat platform and RoboNation for their generous equipment contributions.

The team also would like to thank all of the Marine Robotics Group's sponsors and supporters for their contributions to the overall organization over the past years.

[1]https://www.notion.so/gt-mrg/1eb4d4efb4c7802ca32dde7ae45887d6?v=1eb4d4efb4c7803ea2d5000c33cc5bf6

References

[1] T. Devlin et al. "Implementation of a Reconfigurable Unmanned Surface Vehicle Testbed for Multi-Agent Field Robotics". In: *Proc. OCEANS 2025 - Great Lakes*. 2025.

[2] J. L. Ortiz Solano et al. "A Systems Engineering Approach Enabling Multi-Domain Robotic Field Testing: the Unmanned Collaborative Research Testbed". In: *Proc. OCEANS 2025 - Great Lakes*. 2025.

[3] U.S. Department of Defense. *The DoD Architecture Framework (DoDAF) Version 2.02*. Change 1, 2015. Chief Information Officer. 2010.

[4] Donald P. Eickstedt and Scott R. Sideleau. "The backseat control architecture for autonomous robotic vehicles: A case study with the Iver2 AUV". en. In: *OCEANS 2009*. Biloxi, MS: IEEE, Oct. 2009, pp. 1–8. ISBN: 978-1-4244-4960-6. DOI: 10.23919/OCEANS.2009.5422212. URL: https://ieeexplore.ieee.org/document/5422212/ (visited on 02/18/2024).

[5] A. Chan et al. "Stinger Tug: Educational Robots in the Maritime Domain". In: *Proc. OCEANS 2025 - Great Lakes*. 2025.

[6] Alexander Kirillov et al. *Segment Anything*. arXiv:2304.02643 [cs]. Apr. 2023. DOI: 10.48550/arXiv.2304.02643. URL: http://arxiv.org/abs/2304.02643 (visited on 01/19/2026).

[7] J. Redmon et al. *YOLOv8 Object Detection Model*. Accessed: Jan. 8, 2026. 2023. URL: https://ultralytics.com/yolov8.

[8] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. arXiv:2103.00020. Feb. 2021. DOI: 10.48550/arXiv.2103.00020. URL: http://arxiv.org/abs/2103.00020 (visited on 10/30/2025).

[9] *IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation*. IEEE, 2017. DOI: 10.1109/IEEESTD.2017.7920364.

[10] M. Siegel. "The sense-think-act paradigm revisited". In: *1st International Workshop on Robotic Sensing, 2003. ROSE' 03.* Orebo, Sweden: IEEE, 2003, p. 5. ISBN: 978-0-7803-8109-4. DOI: 10.1109/ROSE.2003.1218700. URL: http://ieeexplore.ieee.org/document/1218700/ (visited on 01/19/2026).

[11] Xu Zhao et al. *Fast Segment Anything*. arXiv:2306.12156 [cs]. June 2023. DOI: 10.48550/arXiv.2306.12156. URL: http://arxiv.org/abs/2306.12156 (visited on 01/19/2026).

[12] Samuel Seifert et al. "Why Yes, That is Hot Glue: A Simulation Based Approach to Roboboat 2016". In: ().

[13] Brian Bingham et al. "Toward Maritime Robotic Simulation in Gazebo". In: *OCEANS 2019 MTS/IEEE SEATTLE*. Seattle, WA, USA: IEEE, Oct. 2019, pp. 1–10. ISBN: 978-0-578-57618-3. DOI: 10.23919/OCEANS40490.2019.8962724. URL: https://ieeexplore.ieee.org/document/8962724/ (visited on 01/19/2026).

APPENDIX A: TEST SCHEDULE

The BlueBoat Extended platform has undergone 13 field deployments as part of the broader UCRT framework development, with comprehensive test reports and maintenance records documented in the team's Notion database. These prior tests validated the baseline platform architecture, sensor integration, and communication infrastructure that form the foundation of the RoboBoat 2026 entry. However, competition-specific testing—including task-oriented behaviors, emergency stop validation, and mission sequence rehearsals—follows an accelerated timeline driven by the January 14, 2026 submission deadline for the Technical Design Report, competition video, and team website.

The competition test schedule spans January 11-14, 2026 for pre-submission validation, followed by five weeks of post-submission testing and refinement leading to the February 18, 2026 competition date. The schedule balances simulation development, Hardware-in-the-Loop verification, and field testing at Sweetwater Creek while maintaining critical dependencies between simulation validation, component integration, and field deployment. By leveraging the mature UCRT platform and existing field procedures, this timeline focuses effort on competition-specific capability development rather than fundamental system integration.

TABLE I
ROBOBOAT 2026 TEST CAMPAIGN TIMELINE

Period	Dates	Phase	Primary Activities	Deliverables	Risks
Pre-Submission	Jan 11–14	Final Validation & Documentation	• Simulation-based task validation • Algorithm baseline verification • TDR finalization • Competition video production • Website completion	• TDR (Jan 14) • Video (Jan 14) • Website (Jan 14) • Baseline performance metrics	Documentation rush Limited iteration
Week 1	Jan 15–21	Post-Submission Field Test #1	• Full system checkout at Sweetwater • Task 1 gate navigation baseline • E-stop field validation • Communication range testing	• Field test report #1 • Performance baseline • Issue list	Weather delays MLK holiday
Week 2	Jan 22–28	Algorithm Iteration #1	• Address Field Test #1 findings • Perception algorithm tuning • Behavioral tree refinement • Simulation recalibration	• Updated configs • Behavioral tree v2 • Sim-to-field alignment notes	Integration delays
Week 3	Jan 29–Feb 4	Field Test #2 – Task Integration	• Task 1 & 2 validation • Debris detection testing • Multi-run reliability • Channel navigation	• Field test report #2 • Task success metrics • Failure analysis	Weather delays Low success rate
Week 4	Feb 5–11	Algorithm Iteration #2	• Final perception refinement • Navigation planner tuning • Emergency procedure validation • Full simulation regression	• Final configs • Deployment checklist • Version freeze	Late bugs Over-tuning
Week 5	Feb 12–18	Final Field Test & Competition	• Task 3 & 5 validation • Full mission rehearsal • Competition logistics • On-site practice	• Field test report #3 • Competition readiness • Performance data	Transport issues Competition conditions

A. Detailed Period-by-Period Breakdown

Pre-Submission Period (January 11–14): Final Validation & Documentation

Objectives: Complete submission requirements while establishing baseline algorithm performance metrics for post-submission development.

Simulation and Algorithm Validation Activities:

- Finalize Gazebo environments for Task 1 (entry/exit gates), Task 2 (navigation channel and debris field), Task 3 (speed challenge), and Task 5 (marina docking)
- Validate behavioral tree execution and task sequencing across all simulated scenarios
- Establish baseline trajectory tracking accuracy, gate passage success rate, and perception detection range
- Capture simulation and system demonstration footage for competition video production

Controlled-Environment Validation:

- Perception algorithm validation using static buoy targets to characterize SAM segmentation accuracy, color classification precision, and depth estimation consistency
- Inference latency measurement for perception, sensor fusion, and planning pipelines on the Jetson Orin Nano under nominal load
- Emergency stop functionality verification for both local and remote activation paths, including propulsion disable response time
- Software deployment reproducibility check using containerized builds on clean hardware images

Documentation Activities:

- Technical Design Report final review, formatting, and submission
- Competition video production, editing, and submission
- Team website content finalization and publication
- Reference and citation verification

Deliverables:

- Technical Design Report submitted by January 14, 2026
- Competition video submitted by January 14, 2026
- Team website live by January 14, 2026
- Baseline simulation and algorithm performance metrics

Risks: Documentation workload limiting iteration depth; compressed validation timeline; incomplete baseline data prior to submission.

Week 1 (January 15–21): Post-Submission Field Test #1

Objectives: Establish real-world performance baselines and identify algorithm deficiencies under operational conditions.

Field Test Activities (Sweetwater Creek):

- Full system power-on and sensor initialization verification
- GPS/IMU fusion performance assessment in open-water conditions
- Communication range and reliability testing over HaLow and 2.4 GHz links
- Manual override and emergency stop validation during live operation
- Task 1 autonomous gate navigation baseline testing
- Environmental condition logging (wind, waves, lighting, glare)

Deliverables:

- Field test report #1 with ROS 2 bag data and telemetry logs
- Initial task success rates and navigation accuracy metrics
- Communication performance characterization
- Prioritized issue list for algorithm refinement

Risks: Weather delays; reduced team availability due to MLK Day; hardware transport or setup issues.

Week 2 (January 22–28): Algorithm Iteration #1

Objectives: Address Field Test #1 findings through focused perception, planning, and behavior refinement.

Analysis and Development Activities:

- ROS 2 bag replay and failure mode analysis
- Perception performance review (false positives, missed detections, range limits)
- Localization drift quantification and filter parameter tuning
- Communication dropout analysis and mitigation
- Behavioral tree logic refinement and replanning trigger adjustment

Simulation Updates:

- Calibrate simulated sensor noise and latency models using field data
- Validate updated perception and navigation parameters in Gazebo
- Generate targeted test scenarios reflecting observed field failures

Deliverables:

- Updated perception and navigation configuration files
- Behavioral tree version 2.0
- Simulation-to-field alignment documentation
- Closed issue tracker items with verification notes

Risks: Integration delays; limited regression testing time; interaction effects between updated subsystems.

Week 3 (January 29–February 4): Field Test #2 – Task Integration

Objectives: Demonstrate reliable Task 1 and Task 2 execution with integrated perception, planning, and control.

Field Test Activities (Sweetwater Creek):

- Task 1 entry and exit gate navigation with repeated autonomous runs
- Task 2 navigation channel traversal and debris avoidance
- Color indicator detection and task-specific response validation
- Multi-run reliability testing to assess repeatability
- Emergency stop validation during autonomous operation

Success Criteria:

- Task 1 gate passage success rate $\geq 80\%$
- Task 2 debris detection rate $> 70\%$ at 10 m; color classification accuracy $> 90\%$
- End-to-end perception-to-planning latency < 200 ms
- Zero collisions during autonomous operation
- Emergency stop response time < 2 s

Deliverables:

- Field test report #2 with task-level performance metrics
- Updated perception datasets for continued tuning
- Failure analysis and recommended mitigations
- Demonstration video footage

Risks: Weather disruptions; hardware failures; insufficient task success rates requiring additional iteration.

Week 4 (February 5–11): Algorithm Iteration #2

Objectives: Finalize perception and navigation algorithms and prepare for competition deployment.

Refinement Activities:

- Final perception tuning for competition lighting and water conditions
- Navigation planner optimization (velocity profiles, clearance margins, replanning logic)
- Task-specific behavior refinement for speed challenge and docking
- Emergency procedure validation and recovery workflow testing

Testing and Validation:

- Full simulation regression testing across all tasks
- Edge-case testing (partial occlusion, glare, sensor dropout)
- Software deployment verification on backup compute hardware
- Competition checklist dry run

Deliverables:

- Final perception and navigation configurations
- Competition deployment checklist
- Version-tagged software repository freeze
- Backup hardware configuration documentation

Risks: Late-discovered bugs; insufficient convergence of tuning parameters; limited time for edge-case validation.

Week 5 (February 12–18): Final Field Test & Competition

Objectives: Validate remaining tasks and execute RoboBoat 2026 competition runs.

Field Test Activities (Early Week):

- Task 3 speed challenge validation
- Task 5 marina navigation and docking approach testing
- Full mission sequence rehearsal
- Stress testing with consecutive autonomous runs

Competition Activities:

- Transport and setup at competition site
- Safety inspection and practice runs
- Official competition runs and scoring
- Post-run data collection

Deliverables:

- Final field test report #3

- Competition performance data and scoring results
- Post-competition analysis and lessons learned

Risks: Weather constraints; transport delays; competition environment differences (lighting, water conditions, GPS availability); last-minute integration failures.