MIT Arcturus

August 2023

1 Introduction

Arcturus is the Massachusetts Institute of Technology's autonomous marine robotics team. Our mission is to give undergraduates hands-on experience leading teams and applying their knowledge by building and programming autonomous vehicles, share our love of robots with the community through outreach events, and create technology that we believe will positively impact the environment and coastal communities.

This technical report explains the design of our autonomous surface vehicle (ASV) built for the Njord Challenge 2023, *Ship Happens*, seen in Figure 1. Our design strategy was to create a robust simulation platform, a stable yet maneuverable hull design, and a comprehensive electronic support system. The particularly innovative elements of our design which we are proud of include:

- Azimuth thruster pods which allow our thrusters to rotate in place relative to the hulls
- Safety modules for remote and on-boat emergency stops
- Fully-defined transform tree for the robot

Figure 1: *Ship Happens*, our autonomous surface vessel (ASV).

2 Vessel design

2.1 Hull design

Ship Happens was originally developed as a platform for autonomous oyster bag flipping through MIT Sea Grant, but in the years since its conception, its hulls have been reused as a flexible platform for a variety of research and competition applications [1].

At 1.75m long, it is compact enough to fit inside an SUV but large enough to house necessary electronics and testing equipment one might need for a specialized mission. For stability, we selected a catamaran design. We used a 1.5m bridge made of marine plywood which spans nearly the entire length of the hulls to maximize the area of the deck and give more flexibility for component placement. The hulls themselves were custom-made. First, six layers of foam board are Computer Numerical Control (CNC) milled such that they create the general hull shape when stacked on top of one another, as seen in Figures 2 and 3. The foam layers were glued together, with a single wooden beam running down the center line of each hull to hold the hulls up while the epoxy was applied along the entire exterior. Lastly, a top layer of plywood was added to provide a mounting point for 3-inch steel studs to screw in and mount the deck.

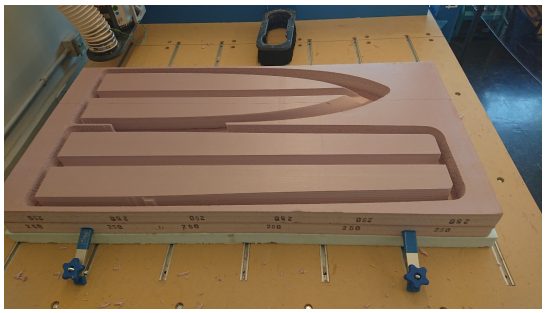


Figure 2: Foam board is milled out to shape.

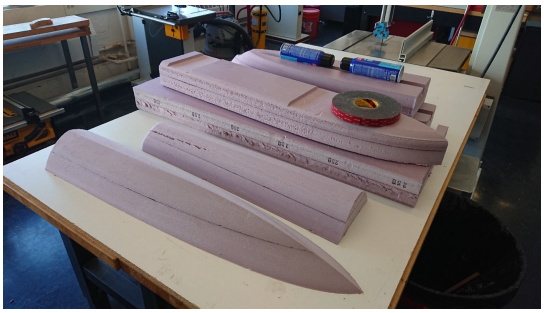


Figure 3: Pieces are stacked together to create a hull shape.

After the foam layers were stacked, they had to be waterproofed. The foam was coated in layers of fiberglass soaked in epoxy. On the outer surfaces, strips of fiberglass were laid in random directions and smoothed out with a squeegee. For sharp corners, it was easiest to soak the first few layers of fiberglass strips in epoxy and then lay them over the foam using welding rods as chopsticks (Figure 4). Once the first layer was dry and adhered to the foam, strips of fiberglass were laid down and epoxy was painted on the surface (Figure 5). As necessary, excess fiberglass was cut off to prevent it from sticking off the surface. Furthermore, bubbles between layers were pushed out to ensure smooth adhesion. After each layer dried, the hulls were sanded so that the surface was smooth and followed the intended shape without sharp edges. This entire process took weeks as each layer takes several hours to dry.

Figure 4: Using welding rods to place strips.

Figure 5: Painting on epoxy onto fiberglass.

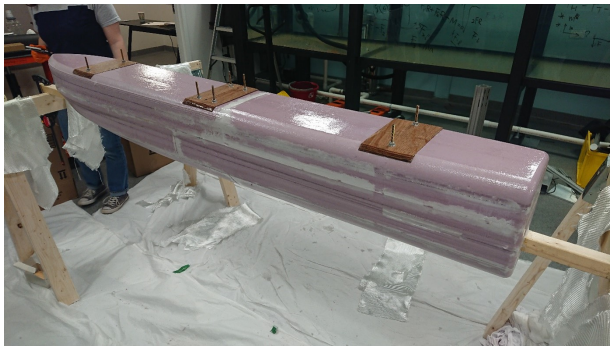


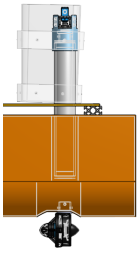
Figure 6: The completed hulls after drying.

This approach of creating our own composites had several benefits. Firstly, by using a rigid foam board on the interior, we can ensure that the boat would continue floating even if punctured. Furthermore, this allows for total customization in hull shape, allowing us to integrate our unique thruster pod design. It was also very cost-effective, costing less than $500 when similarly sized inflatables typically cost $2K to $5K. In total, each hull weighs 9kg and can together hold a maximum load of around 135kg.

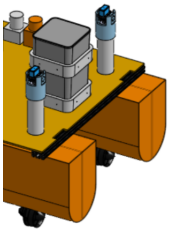
2.2 Propulsion

To increase our maneuverability and prevent us from drifting into obstacles, we developed an azimuth thruster pod design that allows us to rotate our Blue Robotics T200 Thrusters in place (Figure 7). Each T200 thruster is attached to a servo motor through a PVC pipe so that we can reorient the thruster relative to the hull. The system is mounted inside a larger PVC pipe to create a telescoping mechanism, allowing us to compress the thruster into the hull during transport and deploy the thrusters in the water. By mounting the thruster this way, the force is redirected into the bearings, preventing shearing of the mount. The PVC extends up through the hull and above the deck to maximize the space between the thruster and the upper bearing, thus minimizing the force on the bearings caused by the cantilever.

(a) Cross-section of thruster assembly.



(b) Thruster assembly integrated into the hulls.



(c) Thruster assembly integrated into the vehicle.

Figure 7: Azimuth thruster subsystem

2.3 Batteries

Ship Happens runs on two 22.2 Volt, 22 Amp-hour, 6 cell, lithium polymer batteries. Each battery is fitted with its own 100A 6-cell lithium polymer battery management system (BMS) with overcharge voltage, over-discharge voltage, current, and temperature protections (for more specifics refer to Appendix A). One battery provides power for the thrusters, while the other powers the autonomy stack, controlling everything else on the vehicle. This includes the Nvidia Jetson AGX Xavier computer, the custom signaling circuit boards, and the sensors.

We chose to power both of these circuits with twin 22.2V LiPo batteries since both our circuits have certain components that require the 22.2V power supply voltage – specifically the computer and the thrusters. To simplify maintenance, charging, and general battery logistics, we wanted to choose one battery model with specifications that could satisfy the requirements of both.

Our computer requires a stable 19V to operate optimally, so we either needed to step up a smaller voltage or step the voltage we had down. The thrusters could also run at lower voltages. Given the bigger size of our boat, we were not as worried about the weight of the battery and were more concerned with optimizing for thrust. Thus we decided it was a better idea to work with bigger batteries that were familiar and reliable. Choosing a smaller battery might have been more efficient were it not for the steady current and voltage demand that the autonomy stack requires. Finally, given that moving the boat in and out of the water via crane would be much less convenient than what we are normally used to during testing, we decided to increase our time on the water at the cost of battery size. Our 22.2V 22Ah 6-cell LiPo batteries provide both the capacity and approximate voltage required by both of our circuits.

2.4 Sensors

The Velodyne HDL-32E is a state-of-the-art LiDAR sensor for capturing 3D data in high resolution. With its 360-degree horizontal field of view, we are able to create detailed maps of the surrounding environment and navigate accordingly. We used this specific LiDAR simply because it was available to us on loan and we could not afford to purchase another LiDAR tower.

The ZED 2i RGB depth camera provides us with high-definition RGB color imagery, which is a necessary input to our vision stack for making navigation decisions. The ZED 2i sensor can capture stereoscopic video allowing us to reliably collect depth data as well. Moreover, the ZED 2i has a large user community, embedded virtual inertial odometry, RTMAP capability, and an IMU. It is also easy to connect to the computer as it maintains a power and communication connection over USB.

We utilize the Pixhawk GPS module to gather accurate global positioning data. This sensor not only aids in navigation since we can follow predefined waypoints, but it also logs GPS data which we can be used later to review the ASV's path and travel data. Furthermore, the Pixhawk has its own IMU which we can fuse with the ZED odometry to obtain accurate localization data. (More information on this is in the Software section).

2.5 Hardware implementation

We have two main power systems on board, the thruster system and the control system, each with its own isolated supply (Figure 8). While the thruster circuit is pretty simple, in the following sections we explain design choices for the power supplies and the control circuit.

2.5.1 Isolated Power Supply

A basis of our electrical design is the isolated power supplies for the computer and the thrusters. When one 22.2V LiPo battery is powering both the computer and thruster circuits, thrusters spinning near maximum speed draw too high of a current and shut off the computer mid-run. Isolating the powers for the thrusters from that for the computer avoids this issue. Each power circuit also has its own manual onboard switch, rated for 300A continuous and max DC 48V. The result of this is more specific checkpoints before turning on the boat as both need to be on for the boat to run, allowing for more thorough testing and debugging of each circuit. Moreover, separating power for both subsystems extends testing windows by reducing the need to switch out or recharge the battery.

2.5.2 Control Circuit

Our control circuit runs everything on the ASV except the thrusters. This system has three main hubs: our navigational control unit, 12V power supply, and 19V power supply.

The navigation control unit for *Ship Happens* is a Pixhawk 2.4.8 flight control unit. This unit acts as the intermediary between an Nvidia Jetson AGX Xavier, our onboard computer, and two Blue Robotics ESC200s that control our thrusters. The Pixhawk itself is powered through a power sense module that also

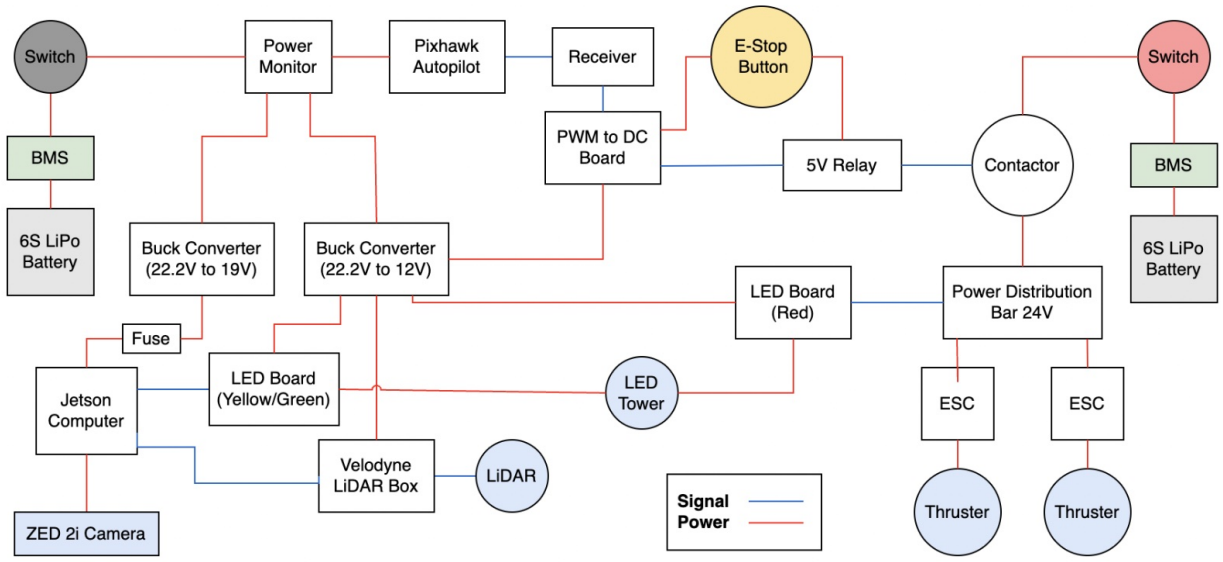


Figure 8: Block Diagram of the electronics diagram.

provides additional remote power monitoring capabilities. There is a 12V power bus and a 19V DC barrel jack for our Nvidia Jetson. These voltages are produced by two buck converters that step down the voltage from the battery. The 12V bus powers our safety circuit, e-stop module, Velodyne HDL-32e LiDAR sensor, and two LED signal boards. Our Nvidia Jetson is protected by a 5A fuse connected to the 19V power supply and powers the ZED 2I stereo camera and Wi-Fi antenna.

Controlling the thruster power is a Kilovac EV200AAANA contactor, a high-power switch with a built-in economizer. When powered on, the contactor's coils close, and the battery connects to the ESCs and powers the thrusters. Normally, the contactor stays open, and the thrusters are not powered. The contactor signal is controlled by a manual kill switch (Baco Emergency Stop button), a power switch, and a remote e-stop module. If either the remote e-stop or the kill switch is engaged, the power to the contactor and therefore the thrusters is cut. The remote e-stop module ensures that signal to the high-current contactors that control power to our thrusters is cut when e-stopped remotely or when communication with the emergency transmitter is lost. The module interprets a PWM signal from our RC receiver and translates it into the DC input for a 5V relay that controls the contactor. This circuit is based on a simple RC low pass filter circuit which uses comparators and op-amps to output a 5V high signal when the PWM is high and 0V low signal when the PWM is low (Figure 9). This conversion was necessary as the RC receiver only outputs PWM. The low PWM output corresponds to switch A on the RC being in state 0 or the RC not communicating with the receiver, and the high PWM corresponds to switch B on the RC being in state 1.

We also implemented an LED tower that provides visual cues for our team to quickly and clearly understand the status of the ASV during testing aside from our remote monitoring. The LED tower features three lights: the green light indicates that the boat is in guided (autonomous) mode, the yellow light indicates that the boat is in manual (remote control) mode, and the red light indicates that the thrusters have no power. To send voltage to the LED tower in all three cases, two circuit boards were designed: one for the yellow and green lights, and one for the red light (Figure 10). The green and yellow LEDs are simple to implement. They each receive a 5V signal from the computer which indicates which mode the boat is in, and the MOSFET acts as a switch turning the lights on and off according to the signal.

Our approach to switching the Red LED was to use N Channel enhancement MOSFETs to gate the signals and power to basically create a custom NOT gate (Figure 11). For the red light, we hook directly into the 24V hub powering the thrusters and use a linear regulator to bring the voltage down to the acceptable 5V controlling the MOSFET gate. This first N Channel MOSFET then further controls the gate to the second MOSFET that actually controls the red light. This is because the N Channel enhancement MOSFET connects the source and drain if the gate is high, but we need the light to turn on when there is no power to

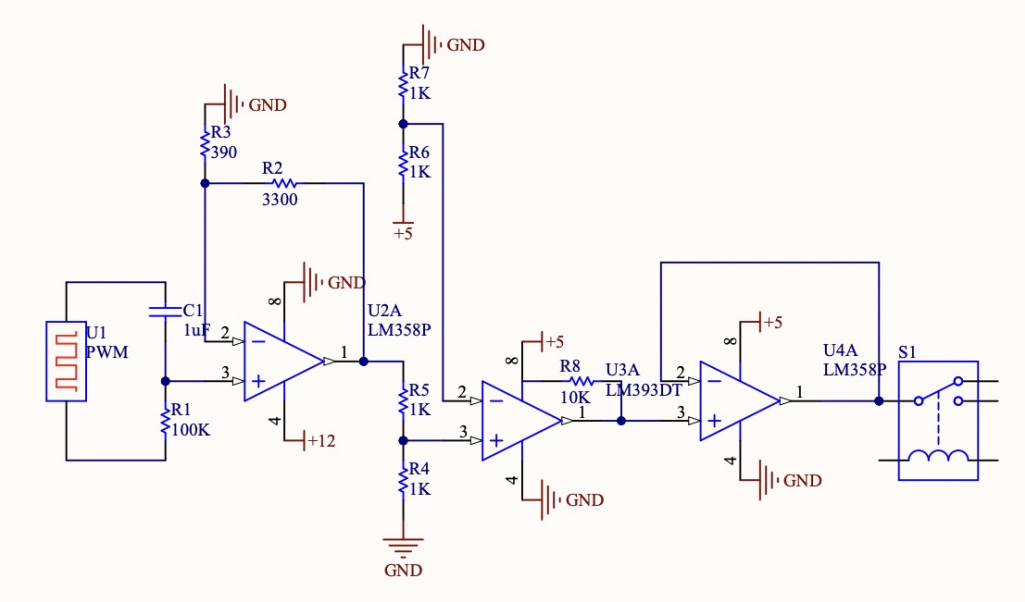


Figure 9: Schematic of our circuit that converts PWM to DC.

the thrusters i.e. when the gate voltage is low. So, instead, we added the second gate such that it polls the circuit right between the 12V source (powered from the 12V power bus) and the drain on the first MOSFET. If the first MOSFET is open (0V from the thruster circuit), we get a 12V signal to the second MOSFET's gate which closes the MOSFET and turns on the Red LED. If the first MOSFET switch is closed (24V to thrusters), the poll to the second MOSFETs gate happens after the current goes through the resistor to the ground and is therefore at 0V, so the light is off.

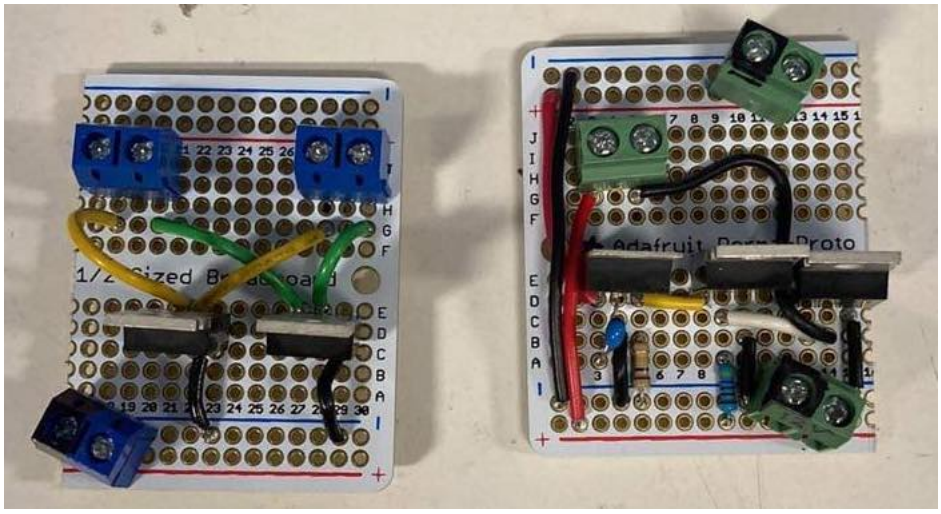


Figure 10: Final protoboards that control when the colors on the LED Tower turn on and off.

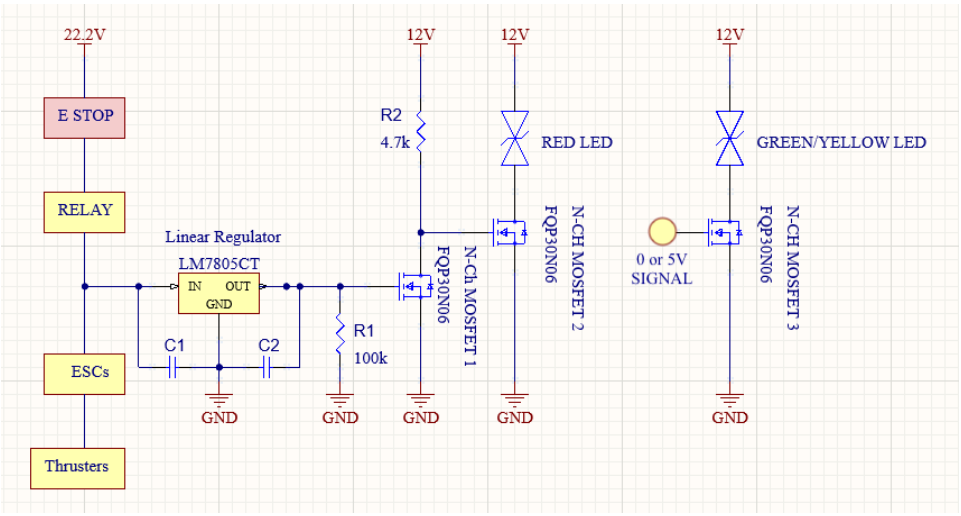


Figure 11: Schematic of our LED tower circuit.

3 Software

3.1 Software Design

The competition course consists of red and green colored buoys and east and west cardinal marks. In order to navigate successfully around these four sea markers, we integrated the YOLOv5 architecture into our vision stack to detect bounding boxes with confidence probabilities. We trained the YOLOv5 model on a few hundred images of colored buoys and cardinal marks that we manually annotated, and then we utilized Roboflow to automate the creation of training, testing, and validation datasets.

We initially treated colored buoy detection and cardinal mark detection as separate tasks, and thus we trained two separate binary classifiers. However, we discovered that overlapping bounding boxes of different classes would sometimes be detected in test images. For example, an image of a true west cardinal marker may show bounding boxes for a red buoy and a west cardinal marker, as the cardinal buoy classifier was not extensively trained on OOD buoys in the absence of relevant robust data and thus would likely associate colored buoy data as a certain cardinal mark (albeit at lower confidence probability – this misclassification would occur vice versa as well). As such, we transitioned to training a single multiclass classifier for all buoy types, which solved the issue of overlapping bounding boxes. We also utilized Intersection over Union (IoU) thresholds with prior bounding boxes to ensure the stability of the bounding box predictions. Figure 12 depicts the workflow of our perception suite.

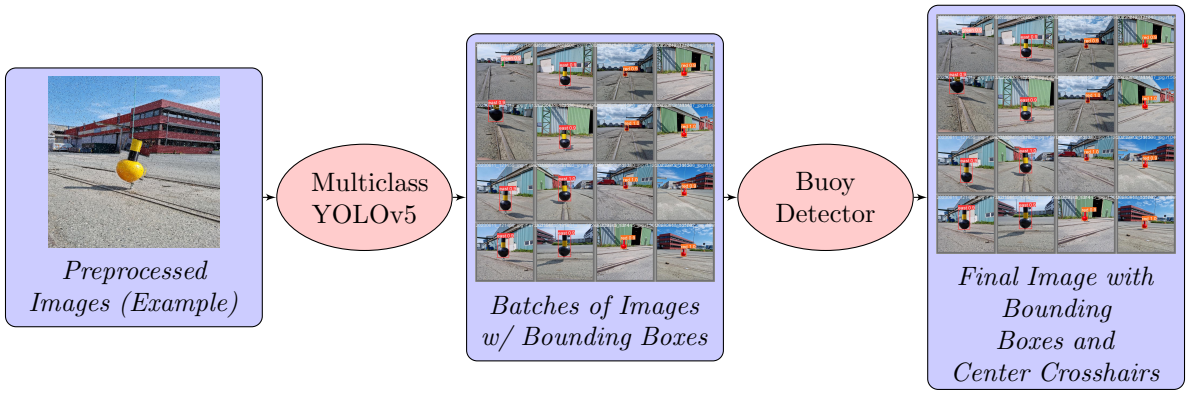


Figure 12: Perception suite workflow.

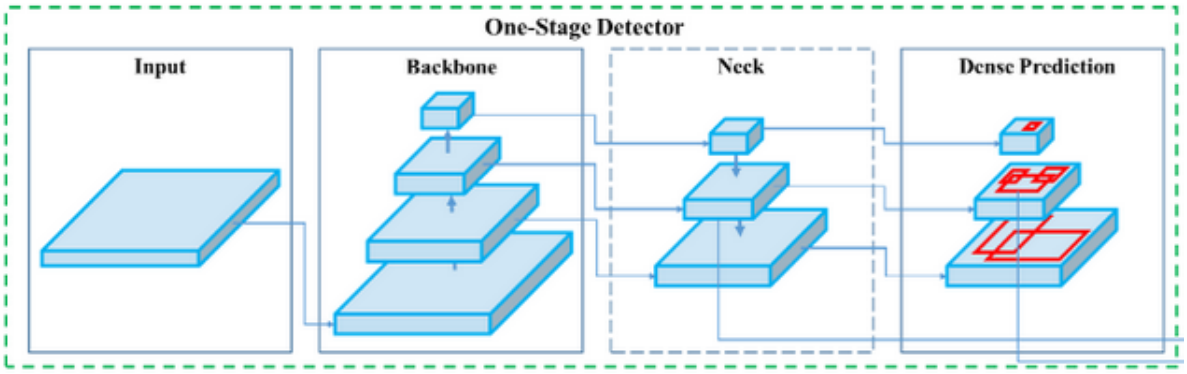


Figure 13: Architecture of YOLOv5, a one-stage detector.

3.2 Architecture

Building upon the insights gained from the previous year's experience with an ArduPilot-centric architecture, our team engineered a new software stack that represents a significant leap forward in terms of customization and control. Unlike prior iterations, this system has been designed to be entirely modular and editable, eschewing reliance on black box algorithms that were previously sourced from external third-party packages. This design philosophy provides us with unparalleled transparency and authority over each aspect of the system's functionality.

Recognizing the critical limitations in control authority that arose from the usage of mavros-based GPS control in the previous system, the new architecture pivots towards a more robust and dynamic configuration. Central to this approach is the integration of a bespoke controller that is tailored to the specific needs of the application, complemented by a state-of-the-art robot_localization implementation and a meticulously designed hector map for the creation of occupancy maps.

One of the most innovative aspects of this new architecture is the integration of ekf-localization for data fusion, allowing us to amalgamate information from multiple data sources seamlessly. Extended Kalman Filter (EKF) localization brings substantial improvements in estimation accuracy, robustness, and adaptability, allowing the system to interpret and reconcile disparate data sources in real time. By leveraging ekf-localization, we have been able to optimize the information flow within the system, enhancing both its reliability and performance.

Furthermore, the entire stack has been designed to operate seamlessly within a ROS1 architecture, embracing its capabilities and standards. This cohesive integration ensures that only published topics are utilized from the MAVlink/ArduPilot system, reinforcing the independence and integrity of our custom-built components. It aligns with our commitment to creating a fully transparent and adaptable framework, one that is poised to drive future advancements in the field.

In summary, this new architecture represents a concerted effort to elevate the system's design to a new level of technical excellence. Through a careful amalgamation of custom design, innovative algorithms, ekf_localization, and a strict adherence to ROS1 standards, we've crafted a solution that is not only powerful and reliable but also stands as a beacon of what can be achieved with focused engineering and a deep understanding of the underlying technologies.

3.3 Implementation

3.3.1 Robot Visualization/Description

The creation of a custom modular .xacro file represented a sophisticated approach to ensuring precise and coherent integration within our robot-localization package. This process began with the careful definition and orientation of the "base-link," serving as the foundational reference frame for the entire robot structure.

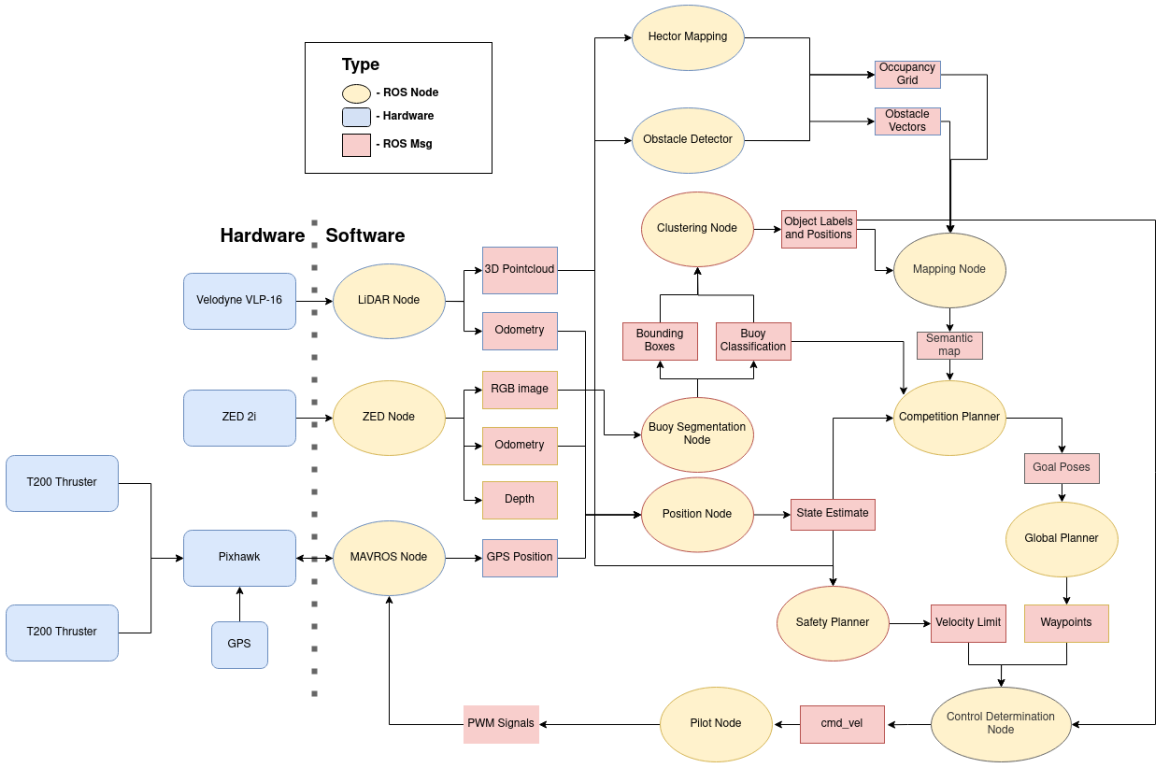


Figure 14: Software architecture diagram

Positioned relative to the robot's physical center, the base-link was meticulously aligned to encapsulate the robot's core structure and geometry.

Subsequent to this foundation, the specialized links were methodically positioned relative to the base-link. The "lidar-link" was defined to capture the mounting position and orientation of the LiDAR sensor, allowing for the seamless transformation of LiDAR data into the robot's frame of reference. The "camera-link" underwent a similar procedure, with its transform being tailored to match the specific mounting geometry of the camera system. By aligning these links, we ensured that visual and depth data were precisely oriented within the robot's coordinate system.

The "imu-link" and "gps-link" were crafted with a specific focus on the unique characteristics of inertial and geospatial measurements. The imu-link was positioned to correspond to the Inertial Measurement Unit's physical placement, enabling accurate transformation of acceleration and rotational data into the robot's frame. The gps-link required additional considerations to align with the Global Positioning System, translating longitude and latitude into the robot's local coordinate system.

Finally, the "propeller-links" were configured to represent the physical placement and orientation of the robot's propellers. These links were vital in modeling the dynamics of the propulsion system, reflecting not only the physical arrangement but also the kinematic relationships governing the propellers' operation.

To visualize this intricate interconnection of links, we employed the rviz-satellite package. This tool allowed us to overlay satellite tiles sourced from tom.tom onto our map frame, converting the longitude and latitude into Universal Transverse Mercator (UTM) coordinates. The transformation facilitated a geographically accurate representation, highlighting the precise alignment of our custom links with real-world spatial data.

Together, the design and execution of these specific links within the custom modular .xacro file stand as a testament to our team's dedication to precision and technical excellence. The harmonious integration of base-link, lidar-link, camera-link, imu-link, gps-link, and propeller-links has resulted in a robust and adaptable system that accurately mirrors the physical robot, providing a platform for advanced navigation, control, and spatial understanding.

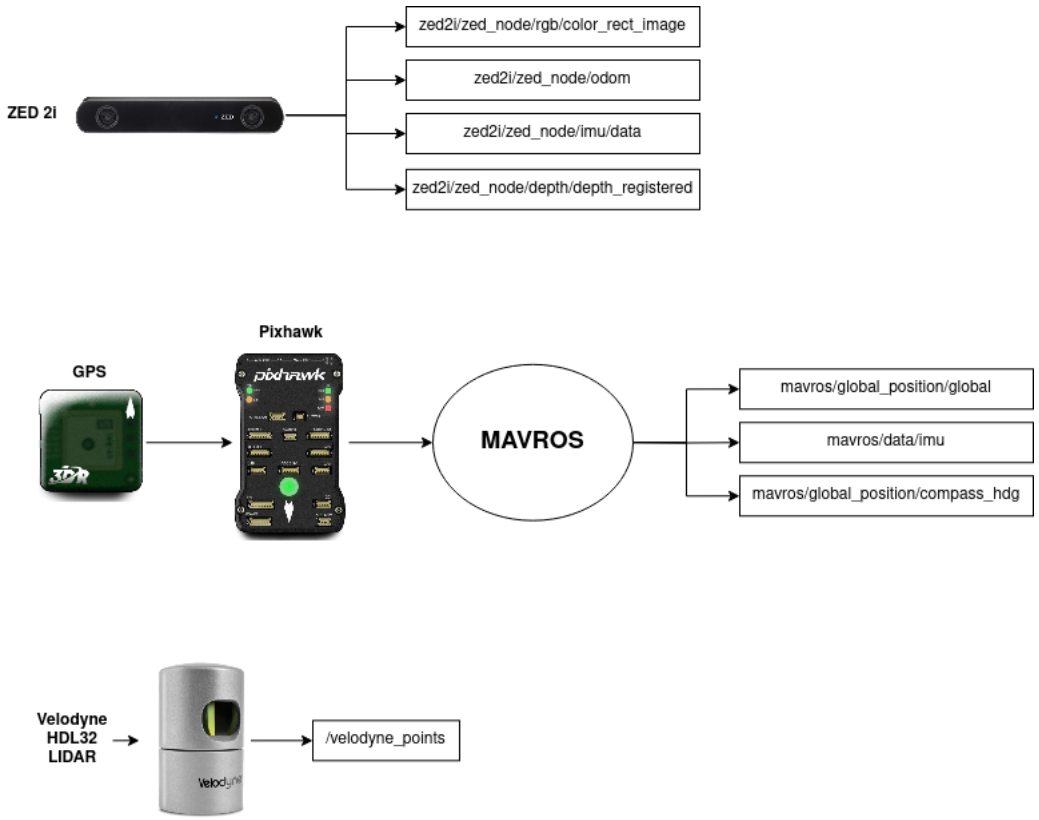


Figure 15: Overview of our perception suite

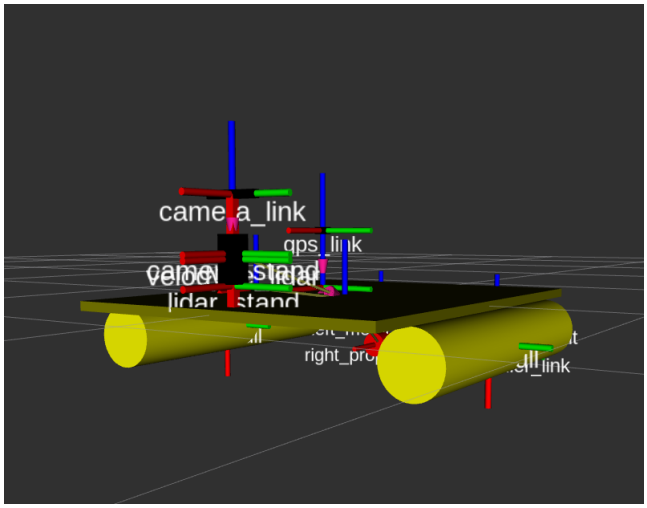


Figure 16: Diagram of our basic URDF model with defined transform frames

3.3.2 Obstacle Detection and Representation with RANSAC and LiDAR

The task of obstacle detection and representation onboard our boat involved leveraging the rich data provided by the Velodyne HDL-32 LiDAR, known for its high-density, 360-degree environmental sensing capability. We implemented an innovative combination of RANSAC and LiDAR point clustering algorithms to detect obstacles and create 3D bounding boxes that encapsulate them. Here's an in-depth look at how these processes worked together:

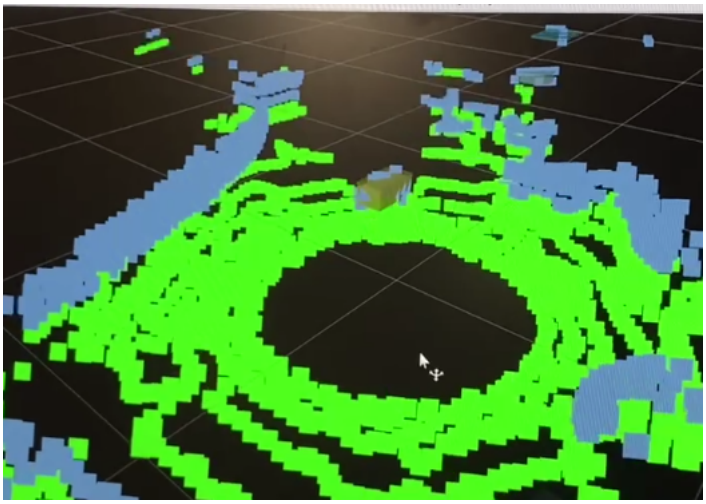


Figure 17: Visualization of our obstacle detection algorithm

3.3.3 RANSAC for Simulating the Floor

The first step was to simulate the floor, or water surface in our case, which acts as a reference plane for distinguishing obstacles. This was achieved through the RANSAC algorithm, a robust model-fitting method.

1. **Data Sampling:** From the LiDAR's point cloud, we randomly selected a subset of points.
2. **Model Hypothesis:** Using the sampled points, we hypothesized a plane model that could represent the water surface.
3. **Consensus Building:** We then iteratively compared all other points in the dataset to this model, determining which points fit within a specified tolerance. Those within this tolerance were considered inliers, contributing to a consensus on what constitutes the floor.
4. **Refinement:** The process was repeated with different subsets of points to find the model that had the most inliers, thereby representing the best estimate of the water surface.
5. **Final Model Determination:** The inliers for the best model were used to compute the final plane equation, representing the simulated floor.

By modeling the water surface, we could then differentiate between points that are part of the floor and those constituting potential obstacles.

3.3.4 LiDAR Point Clustering for Obstacle Detection

With the water surface modeled, the next step was to analyze the points above this plane to detect obstacles.

1. **Point Segmentation:** Points were segmented from the main cloud based on their distance from the RANSAC-generated plane. Those significantly above the plane were considered for obstacle detection.
2. **Clustering Algorithm:** We employed clustering algorithms (e.g., k-means or DBSCAN) to group these segmented points into clusters, each representing a potential obstacle.
3. **Bounding Box Creation:** For each cluster, a 3D bounding box was constructed. This entailed determining the minimum and maximum extents in each dimension and fitting an oriented bounding box that encapsulates the clustered points.
4. **Obstacle Representation:** These 3D bounding boxes provided a geometric representation of the detected obstacles, allowing for collision avoidance and path planning.

This synergistic combination of RANSAC for simulating the floor and LiDAR point clustering for obstacle detection was key to our boat's navigation capability. Utilizing the Velodyne HDL-32's rich dataset, we could effectively represent the marine environment, distinguish obstacles with precision, and navigate with confidence and efficiency. The process was not only technically rigorous but also adaptable, providing a robust solution for real-time marine navigation.

3.4 Implementation of RTAB-Map with Visual-Inertial Odometry on Water

RTAB-Map (Real-Time Appearance-Based Mapping) is a sophisticated RGB-D, Stereo, and Lidar Graph-Based SLAM approach used in our marine navigation system. Based on an incremental appearance-based loop closure detector, RTAB-Map uses a bag-of-words approach to determine whether a new image comes from a previously visited location or a new location. This ability to detect loop closures enables the creation of a consistent and accurate map, an essential feature for autonomous navigation.

In our specific implementation, we utilized visual-inertial odometry data from the ZED camera, along with depth data and RGB image data. Visual-inertial odometry fuses visual data from cameras and inertial data from Inertial Measurement Units to estimate the pose of the system. The RGB images captured by the ZED camera provided detailed visual features, while the inertial readings captured acceleration and angular velocity. These were fused in real-time to provide robust and accurate pose estimation.

The depth data from the ZED's stereo vision further enriched the mapping process. It allowed for 3D reconstruction, creating a detailed 3D model of the environment, identifying potential obstacles in the path, and representing significant environmental features in three dimensions. This depth integration added a layer of complexity and precision to the mapping.

The RTAB-Map process was executed through continuous capture of RGB images, depth data, and inertial readings, with a focus on loop closure detection. Using the bag-of-words approach, the system could determine if a new image came from a previous or new location, constructing and updating the map in real time through a Graph-Based SLAM approach. This provided a consistent and robust mapping capability that adapted to the ever-changing marine environment.

We also leveraged the existing implementation within the 'zed_examples', a collection of samples provided by ZED for various applications. This existing workflow allowed us to quickly prototype and customize our solution, following proven methods and best practices within the ZED community. The 'zed_examples' were instrumental in our successful implementation, providing a solid foundation that we could adapt and optimize for performance on water.

In conclusion, the integration of visual-inertial odometry, depth data, and RGB image data, in accordance with the RTAB-Map approach, facilitated real-time appearance-based mapping on the water. This innovative and technically complex implementation, enriched by the insights and existing workflow from the 'zed_examples', has been a pivotal advancement in our navigation system's capabilities, ensuring both precision and efficiency in navigation on water.

3.4.1 Implementation of Navigation Controller with Multi-Level Priority Scheme

In the complex environment of marine navigation, identifying and following an optimal path is crucial. Within our controller suite, we implemented a multi-level priority scheme that integrates various heading control variables to navigate our robot through the environment.

The central concept revolves around two buoys, one green and one red, which are used to create a gate for navigation. The controller processes the bounding boxes around these buoys, specifically identifying the center point of the two bounding boxes, which represents the center of the gate. This perception stack output plays a vital role in the control strategy.

Angle Calculation and Heading Error: The detected center point of the gate is first converted into an angle. This conversion is based on the pixel distance of the calculated center from the midline of the image frame. Depending on this difference, an angle is derived, defining a heading error for traveling to the buoys. This error represents a significant component in determining the heading direction and is the highest priority variable in our control scheme.

Incorporating Additional Heading Variables: Apart from the buoy center, two more variables are considered in our navigation strategy:

- **Cardinal Marks:** The heading direction defined by cardinal marks serves as a supplementary guide, holding the second level of priority. It becomes higher priority by adding a direction component that corresponds to the direction indicated by the mark (east/west). This allows the robot to move either to the left or the right of a cardinal mark that is positioned at the center of a large area spanning the buoy gate.
- **GPS Waypoints:** Finally, the heading angle needed to reach the GPS waypoint is also considered, as the third priority level, ensuring alignment with planned routes.

Control Heading Composition: All three variables are composed into a single control heading within our 'controller_suite'. The process of composing these values follows a hierarchy where the buoy center's heading error is given the highest priority, followed by the cardinal marks, and then the GPS waypoint. By doing so, it ensures not only that the robot navigates towards the gate defined by the buoys but also aligns with other navigation markers and predefined paths.

Controller Suite Implementation: The implementation within our 'controller_suite' is characterized by real-time processing and adaptability:

- **Real-Time Processing:** Continuous adjustments are made in real time to the control heading as the environment changes.
- **Adaptability:** Flexibility to adapt to variations in buoy positioning, cardinal marks, and waypoints.
- **Multi-Level Priority:** Intelligent integration of different control variables based on a clear hierarchy of importance.

In conclusion, the controller's implementation represents a robust and intelligent solution for marine navigation, taking into consideration the dynamic nature of the environment and the complexity of combining various control factors. By prioritizing the center of the gate created by the two buoys and effectively integrating other navigation parameters, the 'controller_suite' ensures precise and efficient navigation on the water.

4 Testing

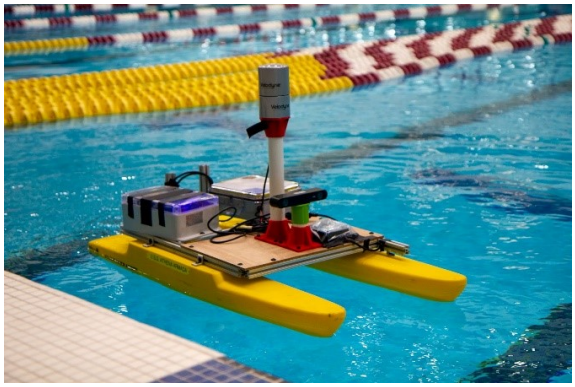
4.1 Sister vessel

Just a few months ago in March 2023, *Ship Happens* had the chance to compete in RoboBoat 2023 in Florida, USA where the boat underwent rapid extensive on-site testing in and out of the water as we competed, as seen in Figure 19a. From this experience, we learned that most of the mechanical construction of the hulls and the base platform was stable and strong, but the electrical systems needed to be more robust and required more failsafes. Therefore, the primary focus for hardware testing leading up to the Njord competition was the onboard electrical systems that we knew we needed to iterate on.

After the previous competition, we gutted the electrical box and in a couple of sprints, rebuilt the bare minimum requirements our navigations team needed. This setup included the thrusters, the Xavier and its associated sensors, and our physical switches mounted onto off-the-shelf hulls.

This sister vessel, which we named *Athena* (seen in Figure 19b), allowed us to continue to program our navigation stack while the hardware on *Ship Happens* was being modified. *Athena* also allowed us to more easily test and deploy our vehicle, as it was much smaller at only 1m and could fit within our test tank in our lab. *Athena* also enabled us to continue testing during the two months when *Ship Happens* was being shipped to the competition.

(a) *Ship Happens* in Sarasota, Florida, USA for RoboBoat 2023.



(b) *Athena*, on the water at the MIT Zesiger Athletic Center pool.

4.2 Electronics testing

Once we had a base electrical system down, we copied that over to *Ship Happens* but with isolated powers for the thruster and control circuits. Then we shifted focus to the other planned improvements: a remote emergency stop module, an LED tower, the BMS, and the contactor (described in section 2). Each of these was first bench-tested and then integration tested before getting permanently added to the onboard systems.

The first two also required circuit boards designed in-house: an LED controller and a PWM to DC converter. These two circuits underwent a great deal of testing themselves on breadboards before we protoboarded them. Due to the tight timeline, we chose to protoboard instead of making PCBs. Once protoboarded they were incorporated into the greater module which was tested and added to the boat. In addition to dry testing, we found that the boat fared well during deployments on the Charles River and that our electrical issues were mostly resolved. That said, testing the electronics will remain a continuous process, especially given the international shipping of the boat and the inherent disturbances from disassembly and reassembly.

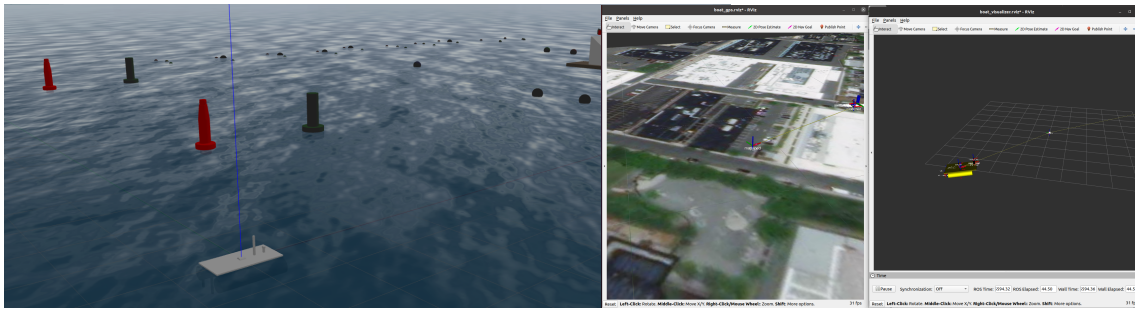
4.3 Testing and Simulation of the Autonomy Stack

The testing and validation of the autonomy stack were particularly challenging, given that the team members were scattered across the country. It necessitated the expedited development of tools and workflows that enabled remote development and testing. A two-fold approach was employed to ensure the robustness and correctness of the stack.

RVIZ Playback and Visualization: RVIZ was utilized extensively for playback and visualization using a variety of bag files. These files contained recorded data that represented different scenarios, and through RVIZ, team members could analyze the behavior and performance of the stack in diverse situations. This method was instrumental in the iterative improvement of the algorithms and tuning of parameters.

Gazebo Simulation and URDF Integration: A full Gazebo simulator played a vital role in testing dynamics, transforms, and control. Leveraging existing wamv plugins and an updated URDF file that integrates Gazebo plugins, the simulator provided a virtual environment that closely mimicked real-world conditions. This environment existing for another competition called RoboBoat. Additionally, an RVIZ display dashboard was used to ensure that transformations were all correct for robot localization, providing an additional layer of verification using bag data form a mavros gps and imu.

In essence, simulation was not just a convenience but a necessity in the development process. It allowed for continuous and consistent testing, regardless of the geographical dispersion of team members, and facilitated a collaborative development environment. The integration of tools like RVIZ and Gazebo, coupled with the appropriate plugins and files, resulted in a well-tested and verified autonomy stack, ready for deployment in real-world conditions.



(a) Gazebo simulation.

(b) Full simulation.

References

[1] M. Kornberg et al., "Autonomous vehicle for oyster aquaculture," OCEANS 2021: San Diego – Porto, San Diego, CA, USA, 2021, pp. 1-6, doi: 10.23919/OCEANS44145.2021.9705669.

Appendix

A BMS Specs

- Overdischarge voltage per cell: 2.8
- Discharge Continuous Amps: 100
- 10 second Discharge Burst: 200a
- Balancing Current: 60mA
- Low temp cutoff: 32° F (0° C)
- High temp cutoff: 167° F (75° C)