RoboBoat 2026: Technical Design Report

Pittsburgh Electric Propulsion

Ben Swanson, Rapheal Amirkhanyan, Chris Yaeger,

Aidan Lefebvre, Soumik Chakraborty, Varun Patel

***Abstract*—This report presents the design and development strategy of Pittsburgh Electric Propulsion's autonomous surface vehicle (ASV) for RoboBoat 2026. As a first-year team, we prioritized developing a reliable and robust ASV that will serve as a foundation for future task-specific optimizations. In particular, our design strategy focuses on creating a mechanically and electrically sound platform capable of achieving the maneuverability, control, and speed necessary for competition. Likewise, the software architecture, built on ROS2 Humble and running on an onboard Jetson Orin, prioritizes core autonomy functionality, forming a foundation for rapid task-specific implementations. Throughout the design and manufacturing timeline, the ASV has been tested, both physically in our university's pool and through simulated test environments.**

I. Design Strategy

A. Hull

For the hull, we decided to go with a displacement catamaran hull to ensure stability and maneuverability. We designed the hull in SolidWorks using surface lofts and validated it using Orca3D, a specialized marine design software.

For the hydrostatics validation, we estimated our vehicle's weight as 80 lbs (36 kg), which was a rough weight estimation that was calculated using the weights of our known components and weight estimation for the hull. Using Orca3Ds hydrostatics feature, we were able to determine the stability of the craft in its upright conditions.

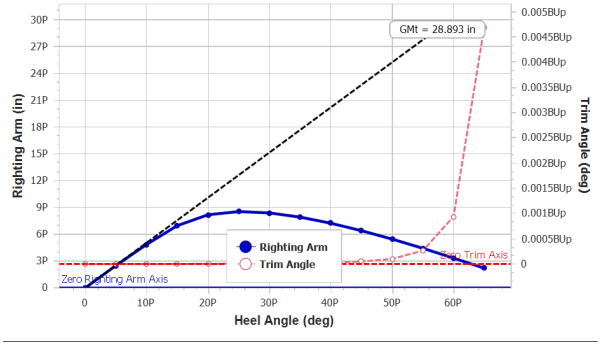

Fig. 1: Graph showing Righting Arm and Trim Angle with respect to Heel Angle.

During upright conditions, the vehicle is stable with a metacentric height of 24.6" and a maximum Righting Arm of 8.532" at a heel angle of 25°. The vehicle starts to slowly lose stability after that, but it becomes heavily unstable after 60°.

For manufacturing, we manufactured the hull using Carbon Fiber by making a 3D printed mold and laying it up with Carbon Fiber. Within the hull, we are adding 3D printed ribs to ensure structural stability from any side collisions. Additionally, we are using Carbon Fiber rods to join the distinct hulls together along with aluminum plates on the top to mount the necessary components for the boat. The hulls will also have lids to access the internal ribs and other components that will be inside the hulls themselves, such as the Speed Controllers.

B. Mast

For the mast, we decided to go for a design that allowed the LiDAR to be mounted well above the surface of the boat to allow maximum view. Additionally, the camera is mounted slightly below the LiDAR using an additional mount attached to the primary mount. We were planning on manufacturing the mount using 3D printed PETG as it allows us to quickly manufacture the mount and make any potential changes that we need quickly. Using ANSYS, we simulated both the mounts using a Factor of Safety of 3, and they both showed to have a minimum Factor of Safety of 15.

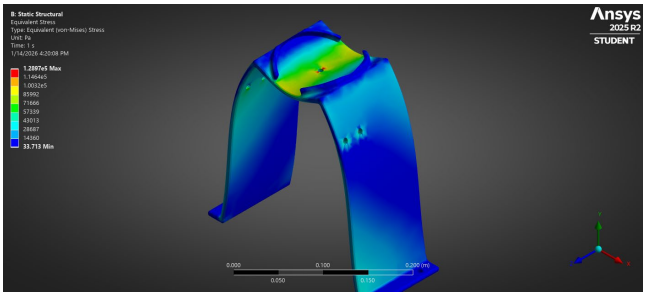

Fig. 2: Simulation of the mast showing the stresses at maximum conditions.

C. Launcher

To complete Task 4 (Supply Drop), we designed a custom launcher to launch both the balls and water. For the launcher mechanism, we

decided to utilize a vertical drop to hold 3 racquetballs at a time, with the ability to launch a single racquetball by triggering a servo motor which lifts momentarily to allow a single ball to pass. To launch the racquetballs, we utilized two motors connected to two flywheels which help launch the balls approximately 2 meters from the boat.

For the water launch, we decided to use a self-priming pump connected to a narrow-angled pipe which launches the water at an angle to help it travel further.

The entire launcher assembly is rested on a turret assembly which helps allows us to control the angle of the launcher with high precision.

D. *Enclosure*

We manufactured two distinct enclosures for software and electrical related equipment.

One of the main concerns with the design of the software enclosure was to ensure airflow such that the Jetson would not overheat during the competition. To tackle this issue, we added waterproof vents to ensure natural airflow through the enclosure. Using ANSYS Fluent, we were able to determine the airflow from the vents and the cooling to the Jetson.

Within the Electrical Enclosure, we are mounting the battery along with various PCBs using a removable shelf which we can easily access. Both of the enclosures will rest on aluminum sheets which will be bonded directly to the hull using epoxy.

E. *Thruster Mounting*

For propulsion, we utilize the T200 thrusters due to their power, efficiency, and ease of control. We are mounting the thrusters in a differential drive configuration with respect to the boat, and the thrusters themselves will be mounted using the help of the inside ribs. The hole for the thrusters will be lined with gaskets to ensure water does not seep into the boat during operation.

F. *Safety Circuit*

This being our first year of competition, the most significant focus of the electrical team was ensuring the safe, reliable operation of the vehicle's system. The design includes multiple circuits to monitor system parameters, independent power management, and an auxiliary battery.

The system's main battery uses a custom, hardware-based battery management system to monitor cell voltage, pack temperature, and discharge current. Limits for cell undervoltage, overtemperature, and overcurrent are set by the configuration of the hardware, and the system will latch in a fault condition if any of these limits are exceeded.

In addition to automatic shutdown from the battery management system, a shutdown can also be triggered via the on-board emergency stop button or the wireless emergency stop button. The wireless emergency stop button transmits a constant heartbeat to the receiver on the vehicle, and if this heartbeat is interrupted for a period of more than 500ms, an emergency stop is triggered.

The safety circuit also incorporates a pre-charge system to limit the rush of startup current that would otherwise occur due to the speed controllers' DC bus capacitors. This prevents potential issues with blowing fuses or triggering the overcurrent fault on startup.

In the event of any fault or emergency stop signal, the safety circuit will latch in the fault state and open the main battery contactor. When this occurs, power is cut to all parts of the boat except for the safety circuit, which will automatically switch to its auxiliary battery. The only ways to release the safety circuit from the fault state are to manually clear the fault using the button on the safety circuit or to power cycle the safety circuit.

The vehicle cannot be powered on unless the safety circuit is receiving power from its auxiliary battery, and even if power from the auxiliary battery is lost, the safety circuit will continue functioning and will still shut down in the event of a fault. The purpose of the auxiliary battery is to power the safety circuit during startup and to allow the reason for a fault to still be saved and displayed after the main battery is disconnected.

G. *Speed Controllers*

The thrusters are independently controlled by four speed controllers interfaced with the Jetson over the vehicle's CAN bus. Each

controller is able to detect and report faults, conduct a self-check on startup, and perform all other tasks related to the motor it is responsible for. The only inputs required from the Jetson are speed commands and a periodic heartbeat signal, but it can also report faults and send statuses if requested by the Jetson.

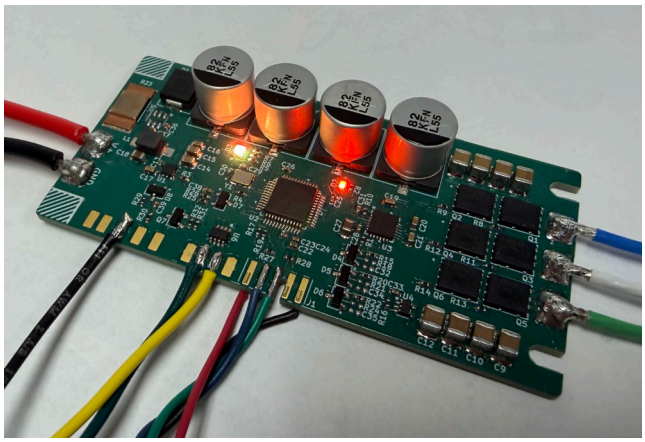

Fig 4: Custom ESC for T200s

H. *Power Management*

The main battery voltage is fed into a power management board for distribution to the rest of the vehicle's systems. This board includes three buck converters: a 12V, 10A converter, a 5V, 2A converter, and a 3.3V, 2A converter. Each of these converters are built around TI converter chips with built in undervoltage lockout and protection for overtemperature, overvoltage, and overcurrent.

I. *Additional Circuitry*

In addition to the previously mentioned circuitry, there are several peripheral boards around the vehicle. The Jetson has a peripheral board for interfacing with the IMU, magnetometer, and CAN bus. There is also a board

II. Software/Autonomy System Design

As a first-year team, we chose to prioritize the development of a modular, general-purpose ASV software architecture over task-specific features, with the intention of meeting those specifications in the upcoming years. By focusing on developing robust core features necessary for autonomy, we hope to lay out the foundation for rapid implementation of task-specific challenges in the future. Our efforts focused on five fundamental areas: localization, perception and mapping, navigation, telemetry, and low-level control systems. While this philosophy meant accepting some trade-off in immediate task-specific performance, we believe that our efforts laid out a necessary foundation for future competition tasks, testing, and growth.

A. ***Localization***

Originally, an extended Kalman Filter [1] was used to fuse GPS RTK and IMU data streams to estimate robot pose. Physical testing revealed that this method of localization was prone to IMU drift over time, leading to the decision to implement the use of LiDAR odometry to generate smooth, low-drift local pose estimates that would prove more accurate. Using DLIO (Direct LiDAR-Inertial Odometry) [2], the localization stack fuses point cloud information with IMU measurements, with an extended Kalman Filter then fusing the DLIO output odometry with GPS data to provide more accurate pose estimates as opposed to the GPS-IMU fusion alone.

B. ***Perception & Mapping***

The mapping system is designed around the creation of two maps, the first being a cost map fed into our motion planning and obstacle avoidance algorithm, and the second being a semantically labeled map utilized in the higher-level task controllers.

Both maps begin with LiDAR point clouds that are initially filtered to remove the water surface, the boat itself, and any outliers, with the remaining points then fed into a Euclidean clustering algorithm to identify the location of any potential obstacles.

Simultaneously, a YOLO-based neural network [3] processes camera imagery to detect and classify objects of interest, such as buoys, markers, and shapes. We then associate these two data streams by synchronizing the cluster and bounding box timestamps before projecting the LiDAR clusters into the camera frame, matching them with detected bounding boxes, taking advantage of the LiDAR's ability to locate objects

with the camera's ability for semantic classification.

Fig. 4: LiDAR clusters projected onto YOLO bounding boxes

These detections are then aggregated into a persistent semantic map that tracks obstacles over time, which is then projected onto a 2D occupancy grid with an inflation layer applied to ensure safe navigation margins during path planning. We hope that, as we compete in future competitions, this system will continue to be serviceable, barring training new YOLO models.

C. *Navigation*

The navigation system is divided into path planning and motion control components, with a multiplexer used to switch between various path planning operational modes, such as waypoint navigation, station keeping, and return to home.

The path planner currently utilizes an A* algorithm to calculate a collision-free trajectory based on the current occupancy map and the desired target pose, with this trajectory frequently tested against future occupancy grids to ensure a new obstacle has not been detected that would cause a collision. If so, the current path will be interrupted, and a new one will be calculated. The motion planner utilizes PD control to determine the boat's desired translational and angular velocity, which are then translated into lower-level controls to communicate with the thrusters.

D. *Low-Level Control*

We utilized the CAN protocol for its reliability and lightweight physical layer, sending individual thruster commands on one CAN bus to each speed controller. In response, each speed controller communicates its current state, such as current velocity, temperature, and fault status, back to the Jetson Orin.

E. *Telemetry*

On shore, our base station consists of a laptop running our custom control GUI, a GNSS module providing RTK corrections, and an emergency stop controller. Via a Wi-Fi link, we can send manual overrides and test waypoints to the ASV, receiving odometry information, battery levels, and a live camera feed in return. A heartbeat signal is also exchanged in both directions to monitor connection health and trigger failsafe behaviors if communication is lost.

III. Competition Strategy

Given our focus on building robust core systems in this first year, task-specific course strategies received less attention. We viewed this as an unfortunate but necessary trade-off that positions the team for stronger event-specific performance in future competitions.

Task 1 - Evacuation Route & Return

Our approach leverages our semantic map, which utilizes LiDAR clustering and YOLO classification to locate, identify, and place key obstacles on a map. In Task 1, we follow the following routine:

- From the semantic map, locate the nearest red and green buoy in the direction of the ASV's heading.
- Designate a waypoint between the two markers
- Move to the designated waypoint
- Repeat for the second set of buoys.

Task 2 - Debris Clearance

Our approach for Task 2 builds off our implementation of Task 1, with the key difference in keeping a running tally of hazards located in the debris field.

The system will follow the following routine: Identify the nearest pair of buoys in the direction of the ASV heading Designate a waypoint between the two buoys Move to the designated waypoint while avoiding debris

Repeat until the next available marker is the red or green debris. Once the task is determined to be completed (by the completion of a circle around the survivor), the system will relay the locations of all debris back to the onshore base station.

Task 3 - Emergency Response Sprint

The ASV first locates the entry gate and navigates through it, saving its location with the mapping stack. Note that, as our occupancy grip follows a sliding window approach, to ensure we do not lose track of this saved location, we treat it as we would a labeled buoy on a world-frame semantic map. We then identify the yellow target buoy and the color indicator buoy, with the color then determining if waypoints are generated for either a clockwise (green) or counterclockwise (red) circumnavigation of the yellow buoy.

Task 4 - Supply Drop

After locating yellow and black vessels, the ASV is positioned to station-keep at a fixed distance and heading from each vessel, with the onboard Jetson Orin communicating serially to an Arduino to either fire the water pump or the racquetball launcher. We utilize solenoid actuators to reset the racquetball launcher, allowing for three shots before reloading is necessary.

Task 5 - Navigate the Marine

Unlike buoys and smaller markers with regular shapes, the dock cannot be reliably detected by our current semantic mapping system due to its irregular geometry and large size being difficult to summarize into a point cloud cluster. Instead, our approach relies heavily on our computer vision pipeline and our occupancy grid generator. The ASV will navigate toward the general area of the marina using our traditional mapping stack, saving a waypoint near the entrance. The ASV will then slowly enter the marina, taking advantage of the wide FOV of our camera to identify green indicators signaling available bays before slowing docking itself. The ASV will station-keep for a fixed time before returning to the previously saved position.

Task 6 - Harbor Alert

To detect audio cues, we implemented a microphone that feeds data into a node that is constantly computing a Fast-Fourier Transform (FFT) to detect blasts at certain frequencies. Upon detection of any number of blasts within a time threshold, the event coordinator would interrupt the current task, and the navigation system would generate a path to either the emergency response zone (one blast) or the marina (two blasts).

IV. Timeline and Testing

A. Timeline

We started the initial design for the boat early in the summer to ensure that we would be able to get everything ready in time. After the school year started, we distributed work to all the members to design different subassemblies for the boat. We planned to finish most of the design in November to give us enough time to manufacture in December and January.

During manufacturing, we are testing the different subassemblies individually before performing a full boat test in late January and early February to help debug any potential issues the boat may have.

B. Manufacturing

For the hull manufacturing, we are 3D printing a mold for half of one hull, which we will then sand and layup with carbon fiber. Since the hull is symmetrical about the x and y axes, using one mold we are able to do four layups which combined gives us our finalized catamaran hull. We are buying premade composite rods to join the hulls together, along with using aluminum sheets on top of the hulls to mount all the enclosures and provide an additional layer of adhesion. We are 3D printing most of the enclosures, masts and thrusters using PLA and PETG as it allows us to quickly customize and manufacture the needed components, while also ensuring they are as light as possible. Our club currently has a Bambu P1S, Prusa XL Dual Head and an Ender 3 V2 Pro which allows us to quickly print and test our designs.

C. Testing

Being PEP's initial year of competing in RoboBoat with no prior ASV to test with, our testing and manufacturing timelines were linked.

Before a physical system was ready and deemed safe, all autonomy software was developed and testing in Gazebo using the Virtual RobotX (VRX) environment [4], providing realistic water dynamics and sensor models alongside the ability to easily add obstacles. We extensively utilized ROS2's rosbags feature throughout software testing, recording sensor data and system outputs to establish benchmark unit tests. Once algorithms performed reliably in simulations and a physical system was created, we transitioned to real-world validation in our local pool. Pool testing focused on resolved issues that are not visible in simulations, such as glare, waves, and drift, in addition to tuning control parameters for the actual boat dynamics.

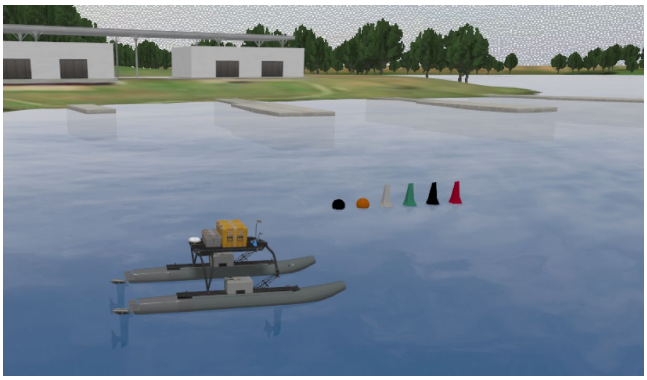

Fig. 5: VRX simulator testing environment

V. Acknowledgments

Pittsburgh Electric Propulsion would like to express our gratitude to those who made this project possible. We thank the University of Pittsburgh for their continued support of student engineering initiatives. We are grateful to the Student Organization Resource Center (SORC) through Pitt Student Affairs, the Green Fund, and the Mechanical Engineering and Materials Science (MEMS) Department for their financial support of our projects. We would also like to thank SolidWorks, ORCA, and ANSYS for providing software licenses that were essential to our mechanical design, analysis, and testing process. We would like to thank Dr. Tony Kerzmann for his mentorship and support throughout this project. Finally, we would like to thank the Student Electronic Resource Center for their support and manufacturing assistance.

VI. References

[1] *T. Moore and D. Stouch, "A generalized extended Kalman filter implementation for the Robot Operating System," in Proc. 13th Int. Conf. Intell. Auton. Syst. (IAS-13), Jul. 2014.*

[2] *K. Chen, R. Nemiroff, and B. T. Lopez, "Direct LiDAR-inertial odometry: Lightweight LIO with continuous-time motion correction," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2023, pp. 3983–3989, doi: 10.1109/ICRA48891.2023.10160508.*

[3] *G. Jocher, A. Chaurasia, J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics*

[4] *B. Bingham, C. Aguero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar, "Toward maritime robotic simulation in Gazebo," in Proc. MTS/IEEE OCEANS Conf., Seattle, WA, USA, Oct. 2019.*

[5]

APPENDIX A:
OVERALL SOFTWARE SYSTEM ARCHITECTURE

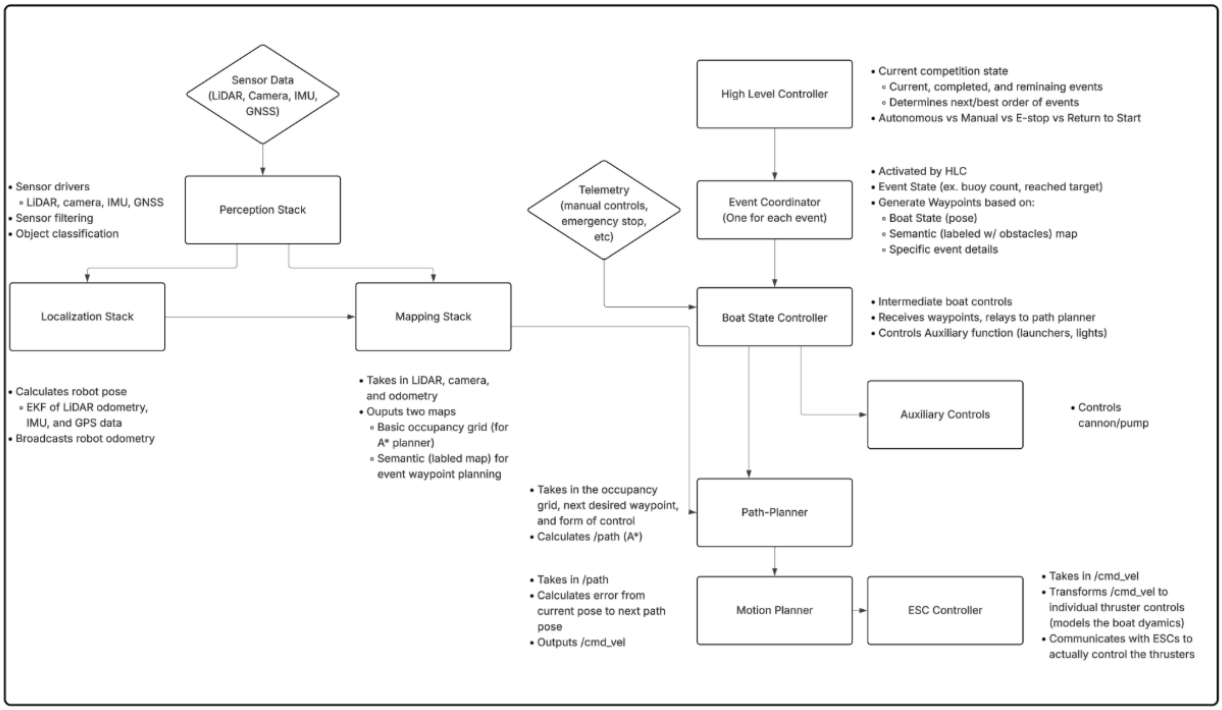

Fig 6: Diagram of overall autonomy stack structure

APPENDIX B:
LOCALIZATION SUBSYSTEM

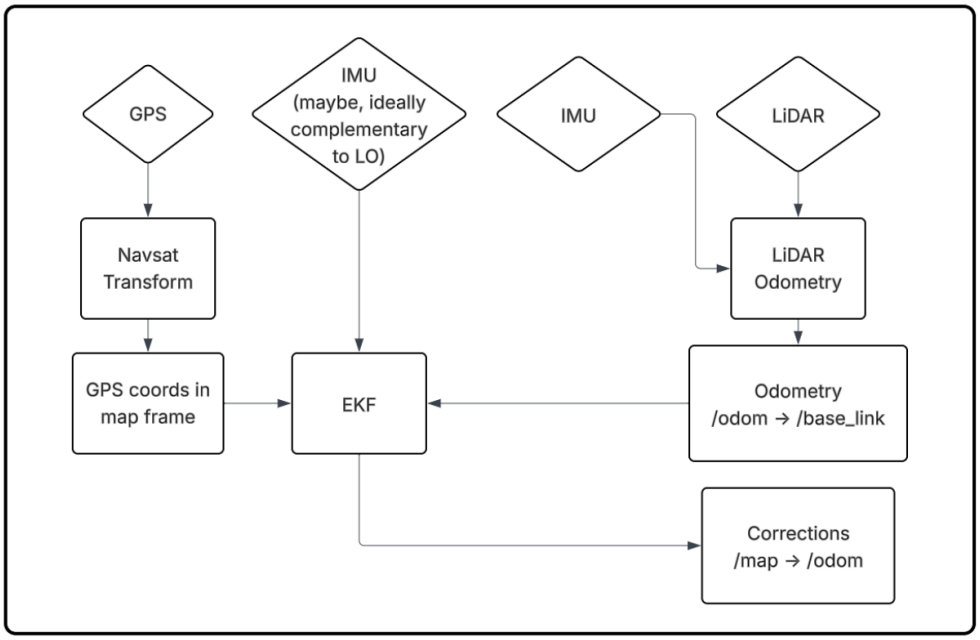

Fig 7: Diagram of localization subsystem

APPENDIX C:
MAPPING SUBSYSTEM

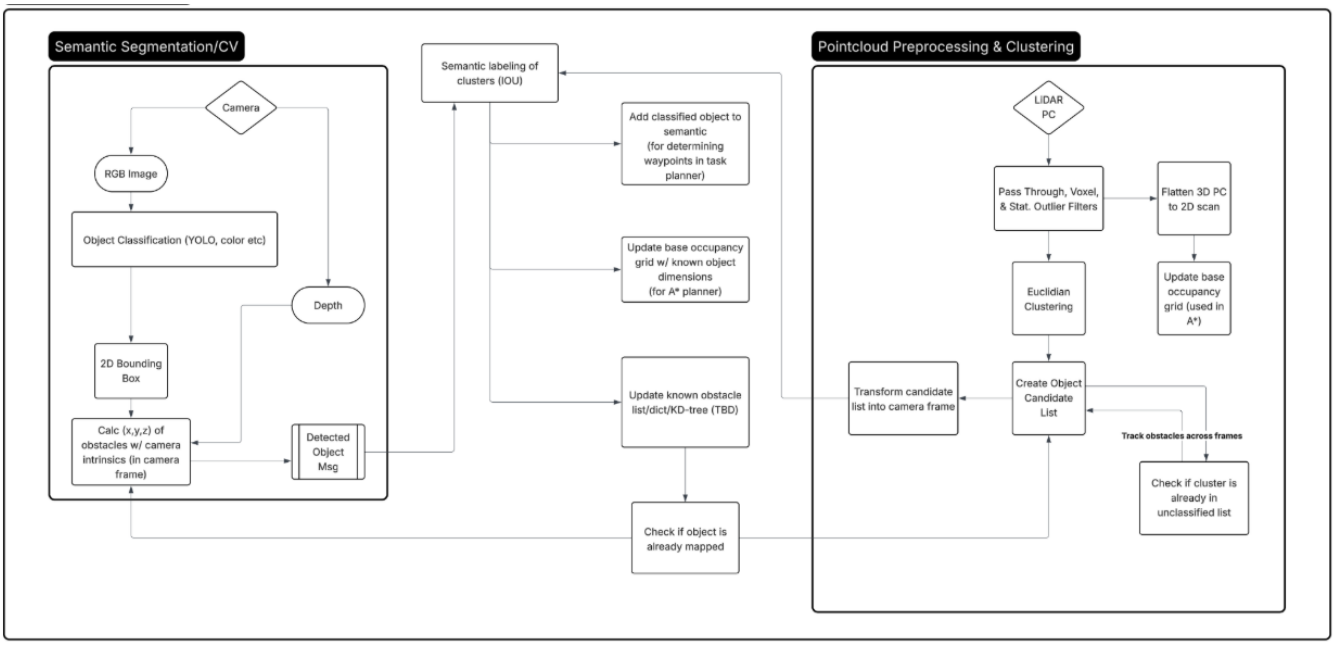

Fig 8: Diagram of mapping subsystem

APPENDIX D:
NAVIGATION SUBSYSTEM

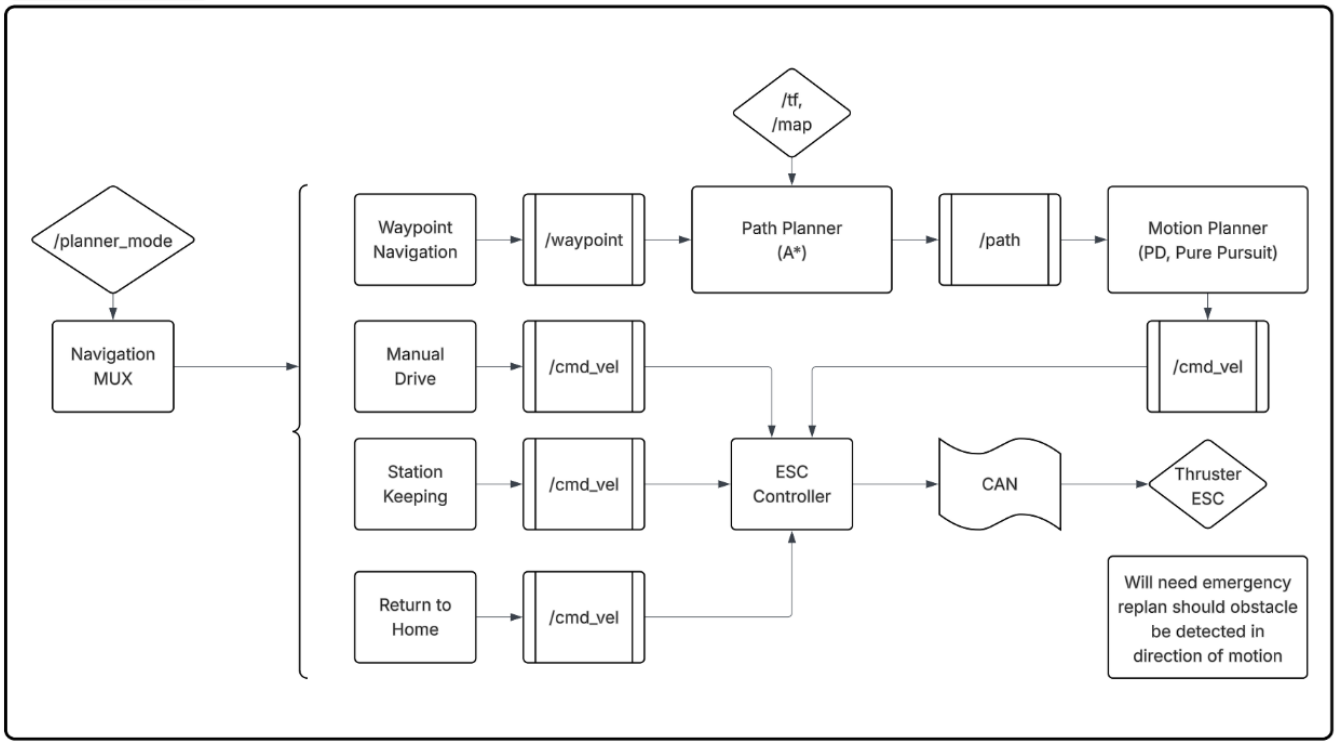

Fig 9: Diagram of navigation subsystem

APPENDIX E:
HARDWARE CONTROL SUBSYSTEM

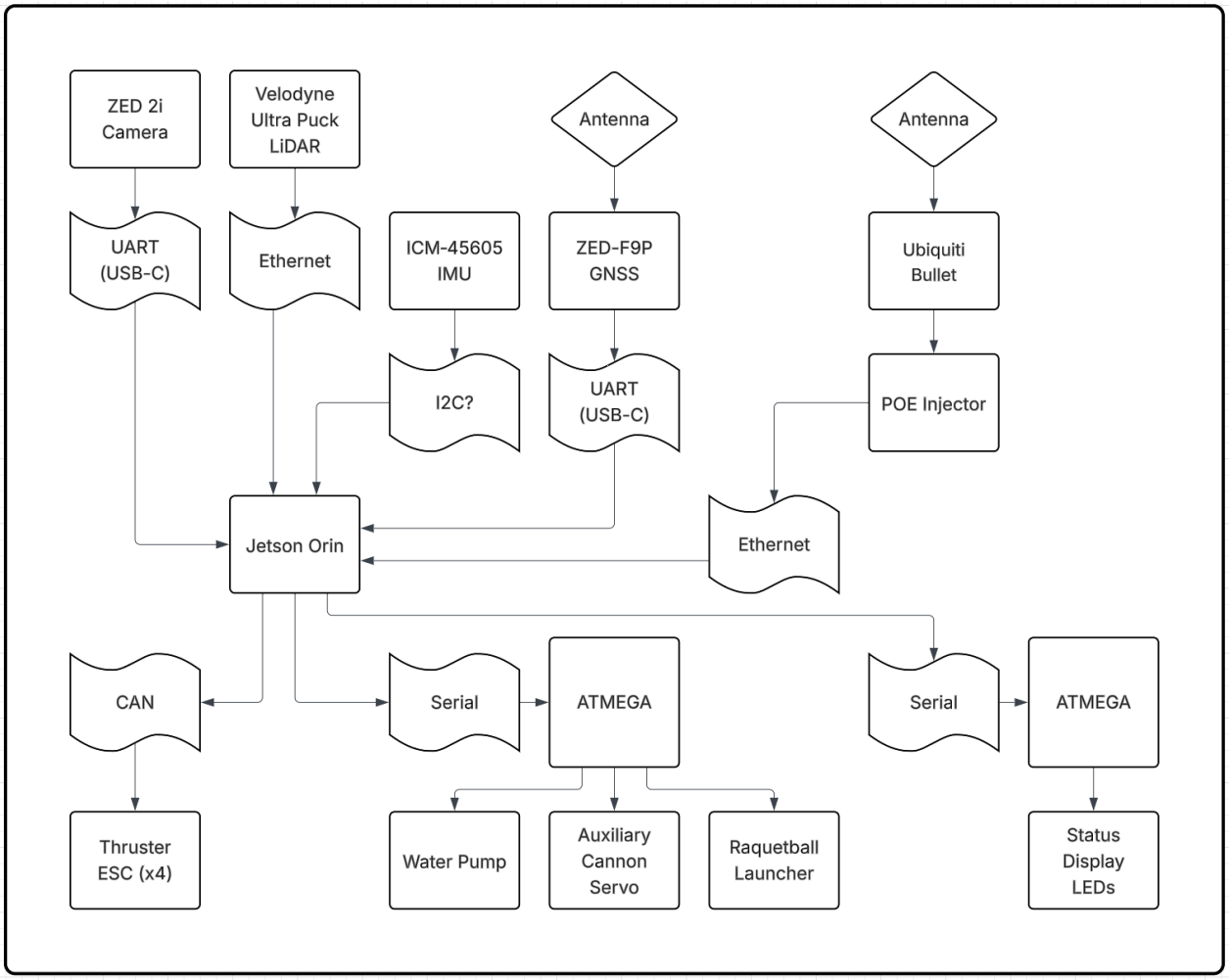

Fig 10: Diagram of hardware control subsystem

APPENDIX F:
ONSHORE BASE STATION

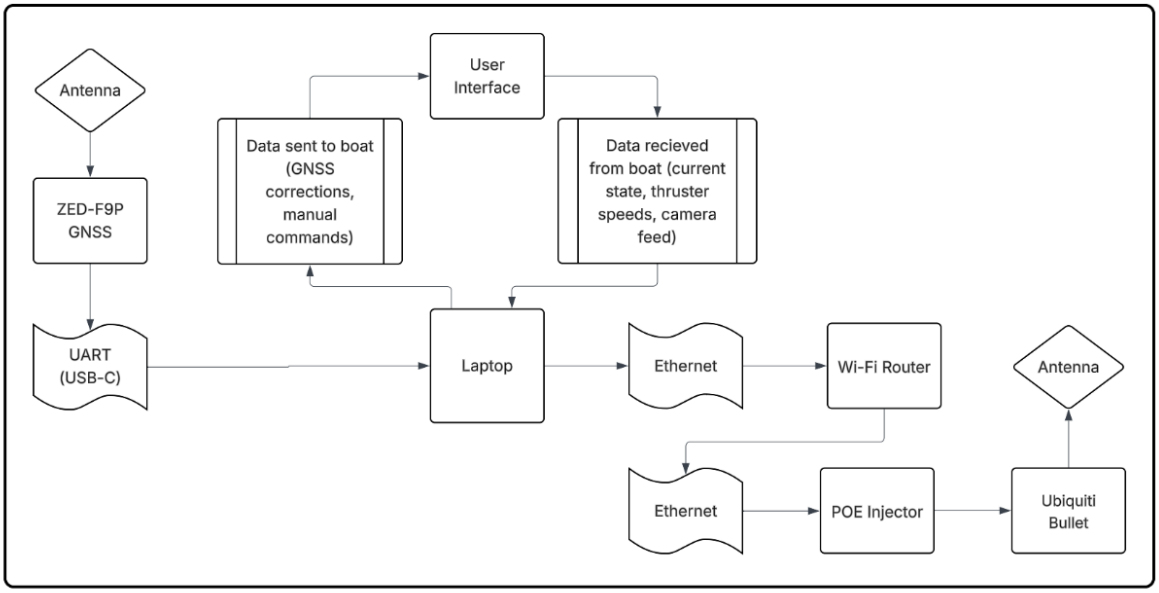

Fig 11: Diagram of onshore base station functionality