RoboSub 2026 Technical Design Report

The University of Southern Mississippi Robotics Club – Autonomous Underwater Vehicle

Cody Atkinson, Drishya Bashyal, Zachary Bennet, Adesh Singh Bharti, Mason Bryan, Timothy Evans, Jonathan Huerta, Muhammad Hussnain, Joseph King, Gabriel Law, Jose Martinez, Andrew Montgomery, Gero Nootz, Prabesh Pathak, Justin Thornhill, Parker Tuminello, William Wroten

Abstract – This technical report presents the design, development, and testing of Marv, an autonomous underwater vehicle designed and built by The University of Southern Mississippi Robotics Club for the RoboSub 2026 competition. The AUV was developed through multiple design phases, beginning with a modified SeaPerch remotely operated vehicle and progressing into a custom-built autonomous platform. The final system uses a six-thruster configuration for maneuverability and depth control, a modular aluminum frame for stability and expandability, and a sealed electronics hull to protect onboard computing and control hardware. An NVIDIA Jetson Orin NX supports vision processing, while ROS2 Humble coordinates communication among stereo cameras, perception nodes, navigation logic, and an ARK flight controller. Custom YOLO-based vision models are used for underwater object detection, obstacle avoidance, and task-specific navigation. Testing focused on waterproofing, buoyancy, balance, vision performance under varied lighting conditions, and competition task simulation in both ROS2 Gazebo and pool environments. The resulting platform demonstrates a practical and adaptable approach to autonomous underwater robotics, while preparing students to address real-world challenges in maritime technology, search and rescue, and autonomous systems.

Keywords – autonomous underwater vehicle, ROS2, underwater robotics, stereo vision, object detection.

I. ACKNOWLEDGMENTS

The development of our autonomous underwater vehicle would not have been possible without the support of multiple parties involved. We are very grateful to Dr. Henry Jones and Linda Bond for financial and technical assistance. Dr. Chad Carpenter and Ishan Agrawal from Integer Technologies for their technical support, Kyle Abrahm from Ocean Aero for his mechanical engineering input towards the vehicle assembly, Zachary Bennett, now systems engineer at Systems Integrated, for his leadership and groundwork for what the team has become, Mr. Bill Powe and his team at the USM Natatorium for allowing us to use the pool to test our vehicle countless times. Without their support, the development of this project would have faced unbearable challenges. We also want to acknowledge all family and friends who have supported all team members towards the extra efforts needed for long hours of work during the building process of our concept and competition vehicles.

A. Where We Started

As the first Mississippi team to enter RoboSub, The University of Southern Mississippi aims to develop talent for the state's budding Blue Industry centered around leveraging technology in maritime roles for civilian and defense industries. Mississippi's position on the Gulf of Mexico provides a natural advantage for the maritime industry, which has spawned many companies with facilities near the state's ports. The challenge for these companies is procuring engineering talent local to the area. The University of Southern Mississippi aims to help fill that need with motivated, multi-disciplinary students, with the experience and education necessary to meet these challenges. RoboSub serves as a way to build that experience. With the help from our supporters and sponsors, students from the School of Computer Science and Computer Engineering as well as the School of Ocean Engineering spent the past two years developing an autonomous underwater vehicle capable of taking on the challenges presented by the RoboSub challenge. With undergraduate and graduate students specializing in control systems hardware, machine vision, robotics, CAD modeling and rapid prototyping, and maritime engineering, we are confident our team can rise to the challenge and deliver a winning entry, while gaining experience and learning valuable lessons along the way.

AUV (autonomous underwater vehicle) development at USM has progressed through four clearly defined stages over the previous two years. The first stage, referred to

internally as Phase 0, consisted of working with a commercial SeaPerch ROV kit modified, as seen in "Fig. 1", gradually to improve performance while operating remotely, though not autonomously. From there, we took the same frame and attempted to make it operate autonomously in what we referred to as Phase 1. This was accomplished with a Raspberry Pi 5 (16 GB), the same DC motors in a three-motor configuration as the initial kit, and driven by L298N motor drivers. It was at this stage that we began working with Ubuntu and ROS2 as the underpinnings of the system we still use to this day. Ultralytics YOLOv5n was chosen as our vision model as it was light enough to run on the Pi5's limited resources and implemented in more advanced research AUV fields [1] [2]. Our nodes were written in Python with simple logic, the object was hard-coded into the node, the instructions were simple, if the object was the one desired, drive motors in such a manner that would cause it to fill a certain percentage of the image frame (as an analog for distance from camera to target) and position it in the center of the frame by denoting pixel values upon which the centroid of the bounding boxes should be attempted to be located within. Unfortunately, with all of the additional equipment added to the lightweight SeaPerch kit, the frame gave up the ghost, so to speak. This necessitated a rapid transition to the third phase, a sprint production completed over the course of a weekend in preparation for a public demonstration. This final phase before the current hull is known internally as Phase 1.5 or "Killdozer" due to all of the aluminum plate that was grafted to the PVC box frame.

In this iteration, the lower hoop of the frame was filled with steel weighted shot until it was neutrally buoyant and perfectly balanced, an issue that previously plagued the SeaPerch-based frame. The aluminum plate provided a solid base for thruster mounts. It was also in this phase that we began implementing PID, BLDC thrusters, and different vision models, as seen in "Fig. 2". Here we learned valuable lessons about the necessity of having a custom-trained model as the lighting conditions in our various test environments provided too much light which washed out the edges of the objects we were trying to detect. While the model performed amazingly well out of the water in the lab, it suffered detection issues as soon as it was submerged.

B. General Overview of USM's AUV

Using a common six-thruster layout, with two providing active depth control and four, positioned at each corner, providing lateral thrust, our AUV "Marv", as seen in "Fig 3", has proven to be extremely well balanced and maneuverable. Compute for Marv's vision models is provided by an Nvidia Jetson Orin NX. Marv detects the environment via two stereo cameras and a depth sensor. The software framework managing the communication between the inputs and outputs is accomplished via a ROS2 "Humble" layout of publisher and subscriber nodes. Coordination of Marv's thruster values is controlled by an ARK flight controller.

II. DESIGN STRATEGY

A. Hardware

1) Frame: Marv's Backbone

The main structure of the AUV is constructed from quarter inch 6061 aluminum plate cut on a laser cutter to a symmetrical trapezoidal pattern with radiused corners. The perpendicular structures are held in place by a series of 3D printed inner and outer corners that clamshell around the joints and are secured in place with M5 bolts of varying appropriate lengths, as seen in "Fig. 4". Within these same corners, a rail system was created to allow for easier exchange of the commercial thrusters we chose for the project. Given their value-based selection, it seemed prudent to facilitate rapid exchange in the event of a thruster failure. These corners themselves were printed on our Bamboo P1S in PET-G at 100% infill to create a consistent level of buoyancy throughout all testing. Previously, we experimented with varying levels of infill and found that while we could save some weight with the lower percentages, they would fill with water and create imbalance and lack of buoyancy as testing went on.

The thruster rail configuration was based primarily off of the US military 1913-Picatinny accessory rail system, as seen in "Fig. 5". A referencing bolt passes through the mounting and rail interface to ensure consistent fore/aft positioning. The rail system itself is held in place laterally by a symmetric bevel. Into the rail an inlet pocket is cut to which a two-part epoxy is applied and the thruster body is affixed. For redundant retention, a cotter-pin is inserted into a cross-drilled hole through both the rail and thruster mount.

To fulfill the requirement for diver-accessible handles, the frame has two mid-ship panels which contain the twin vertical thrusters for active depth control on the inside of the frame rails, as seen in "Fig. 6". On the outer side of that same flat panel, handles are printed in PET-G in such a manner as to both retain the thruster rail against the inside of the thruster mount and provide a solid interface for divers to

move the vehicle while in the water and team members can manipulate the AUV above water.

In an effort to maintain a neutral balance in the water, the lower portion of the frame is supported with tubes instead of aluminum plate. This has dual benefits of both providing a more stable lateral structure for the normally unoccupied, lower portion of the frame, and being a convenient place to fill with weight at the lowest-most part of the frame. This leads to a naturally very stable hull in the water which requires minimum input from the vertical thrusters in order to maintain stability.

Finally, an advantage of the aluminum frame is that it provides a good deal of modularity for additional sensors and manipulators, and as we are in talks with several law enforcement and search-and-rescue agencies to begin research on a specialized AUV to serve their mission, the frame design will likely be a centerpiece of that design going forward.

2) E-Hull: Containing Sensitive Hardware

Initially, we attempted to create our own e-hull (electronics hull) using a heavier schedule-80 PVC tube. Bearing surfaces were printed in 100% infill PET-G. These bearing surfaces primarily consisted of a large insert that was epoxied into the tube to create a single hull, on the protruding portions of the bearing surfaces, we printed a large, coarse thread on which a cap could be threaded. That cap was modeled after a Parker-Hannifin "John Deere" O-ring Face Seal hydraulic fitting, as seen in "Fig. 7". The cap had an open area in its center in which a round aluminum plate, which served as a bulkhead, could be held stationary against an o-ring on the bearing surface. A concentric ring maintained constant centering for the bulkhead. This isolation was necessary to pass both three-phase wires for the BLDC thrusters as well as data input/output cables for cameras, LAN, and servo motors, while not twisting the cables inside the hull, as seen in "Fig. 8". Given that all of this was 3D printed, a threaded configuration seemed a more bulletproof implementation than a flange containing multiple bolts. What we didn't anticipate in this initial configuration was the constant assembly and disassembly would introduce the change for the face o-ring to move out of its optimum position. This became apparent during our testing protocol, thankfully before catastrophe struck.

With lessons learned and time moving quickly towards deadlines, we opted to use a commercially available Blue Robotics unit this season, roughly the same size as the tube we had originally specified. Upon its arrival, we all felt somewhat satisfied to know that the professionally designed hull also implemented a face o-ring of nearly the exact same specifications as the one we chose for our initial design, as seen in "Fig. 9".

In both cases, two-piece waterproof bulkhead connectors are used to isolate the e-hull's interior from the surrounding water. Their quick connect/disconnect nature speeds assembly and disassembly for rapid deployment.

3) Computing/Electronics Components

The single-board computer responsible for our vision model and navigation logic is an Nvidia Jetson Orin NX 16 GB. It is isolated on its own battery, apart from the Blue Robotics ESCs and other peripherals. This is to maintain a constant voltage to both systems. The Jetson communicates via USB with the ARK Flight Controller, which is compatible with ArduSub. The ESCs receive their PWM signal from the flight controller. "Fig. 12" gives an illustration of the control architecture for the computing processes.

Sensors present in the final design are two commercially available stereo-cameras which provide enough environmental information such that we can utilize ROS2's SLAM/VSLAM toolboxes for mapping and positional awareness [3] [4] . One of these cameras faces forward for target detection as well as obstacle avoidance. The other faces downward for object retrieval and object release duties. Each one works on its own vision model with different data sets given the different target requirements and light effects at the different positions in the water. The flight controller has its own pressure sensor which it uses to set depth as a constant operation.

The two manipulators external to the core hardware are the grabber/arm, a basic linkage driven by a waterproof servo motor intended for grabbing and releasing objects. And a torpedo launcher as seen in "Fig. 10".

B. Software

1) ROS2 overview

Underpinning our entire software stack is the Jetson-specific Ubuntu 22.04 Linux distribution, which necessitates the use of ROS2 "Humble" to arrange communication between the different publisher and subscriber nodes. While previous iterations based on the Raspberry Pi 5 could use later Ubuntu releases and thus more recent ROS2 releases, the Jetson is, at the time of this writing, limited to Ubuntu 22.04.

2) YOLOv8n Vision Model Trained on custom data set

In an effort to address the "wash-out" we experienced in earlier iterations of our vision model implementation, the decision was made early in Phase 2 to train our competition AUV on a custom dataset that accurately reflects underwater lighting and refraction effects. Given we have two different stereo camera setups, one facing forward and one facing downward, each will have its own vision model with different target/object detection requirements. The forward camera set will primarily be responsible for the navigation and obstacle avoidance, while the downward facing camera set will be looking for target crates, target objects, and navigation-aiding pathways where present.

3) Publishing Nodes

Naturally, sensor inputs each have their own publishing node. As discussed previously, forward and downward stereo cameras will each have a publishing node. This gives some modularity in which to subscribe to, given a particular task; for instance, slalom or gate traversal tasks might have little need to receive information from the downward-facing camera, and vice versa for object retrieval or dropping, which would have a more weighted focus on the downward-facing camera instead. These two nodes are denoted as f_stereo and d_stereo respectively.

4) Depth and pressure sensors

Initially, we set out to build our own pressure sensor and had several team members make great strides in doing so across our Ocean Engineering, Computer Engineering, and Computer Science programs, each delivering a very accurate sensor-microcontroller pairing. However, when the decision was made to include an ArduSub-compatible flight controller, that particular configuration had a more seamless integration of a pressure sensor that didn't need to be handled as an input in our ROS2 nodes. Each of the two stereo cam nodes publishes data as a string to be analyzed by the functions in the master control node.

5) Publishing and Subscribing Nodes

With the sensors publishing data, we created a master node that is responsible for calling varying library functions that correspond with each individual task. This lets us create tasks with modular functions that can all be called from the central control node. For instance, every task should have an obstacle-avoidance function running regardless if it is gate traversal or object retrieval, and with a library of these functions, calling the needed functions for a given task is simple. This master/central node subscribes to the sensor nodes and publishes navigation data for the ArduSub node which directs the flight controller via serial port.

6) Subscribing Nodes

Finally, the remaining subscriber-only node is the one running ArduSub. It is subscribed to the master/central control node and receives navigation direction from the running task. It directs the flight controller with directional and desired depth inputs, and the positional sensors (pressure/depth, gyro, etc.) handle the minutiae of individual ESC PWM input signals, determining how much, which motor, and for how long.

7) ROS2 Toolboxes

Given that we are running ROS2, it's only natural that we attempt to leverage some of the built-in features for mapping and system analytics. Primarily, the focus here is on SLAM/VSLAM [5] using the stereo cameras' depth-detection capabilities, as depicted in "Fig. 13". This should allow us to set a desired pose when calling the return_home function at the end of each run, to run a best/shortest-path algorithm to place the AUV at its origin for the run.

III. TESTING STRATEGY

A. Waterproofing

This was the central issue in the early phases of AUV development. While we initially made the mistake of neglecting water testing the hull regularly. After a few leaks on Phase 1 and Phase 1.5 hulls, we opted to begin the procedure of testing waterproofness for every single change we made to the electronics hull. This was accomplished both in a tank in the lab as well as in several pools, both at the dispense of different teammates and at the University's olympic pool in the fitness center.

B. Balance, Buoyancy, and Thrust

Secondly, in addition to leak testing, maintaining a balanced hull was the most valuable testing parameter. A balanced hull requires less input from the active depth and attitude control system, thus conserving battery and thruster service life. As with leak testing, with any change to the hull, balance and buoyancy were carefully tested and set to as neutral as possible given the built-in ballast tubes at the bottom of the hull.

C. Vision Model Operation

As we found out at the end of Phase 1.5, maintaining the function of object detection algorithms was a crucial component to the function of our AUV as a whole. Without

some sort of sonar/lidar to provide environmental awareness, Marv's stereo cameras are the only way it can sense the environment surrounding it. As such, testing in varying depths and lighting conditions, from bright mid-day sunlight to the bottom of a four meter deep pool in the evening, obstacles and target detection needed to be tested for function in multiple settings to ensure functionality throughout all foreseeable conditions to avoid problems under uncertain light conditions [6].

D. *Special Task Testing*

Finally, with the knowledge of individual tasks from the competition, we were able to set up an analog in many of our pool sessions. Initially, function was tested in Gazebo with the Marv's physical dimensions and mass added to our virtual model [7], then the same conditions were replicated in the testing sessions, as seen in "Fig. 11".

IV. COMPETITION STRATEGY

A. *Heading Out: Gate Choice and Traversal*

Rather than electing for the coin toss, we'll be selecting our gate to reduce the complexity. Gate traversal has been our most-tested task.

B. *Avoid Debris: Slalom Course*

Running an obstacle avoidance function throughout all tasks as well as a path-detection function, we'll shoot for a no-contact traversal of the slalom course. Opting to attempt to pick up bonus points for the correct path in this task.

C. *Recon: Bin Drop*

Our downward-facing stereo camera gives us the advantage of more precise bin-selection as well as more accurate targeting, as we can get within a prescribed depth, determined thanks to the utilization of stereo vision. We'll attempt to pick up bonus points here.

D. *Deploy: Torpedoes*

Here we'll leverage our Ocean Engineering department's expertise. Our torpedoes will have active stability control which will give us a shot at the bonus points for distance to target.

E. *Resupply: Octagon*

Given, again, our downward facing stereo camera, the action of retrieving the obstacles seems easily within reach here. Surfacing inside the octagon seems manageable as setting distance from an edge to let that be a guide for the center of the octagon itself.

F. *Return Home*

Here, we'll attempt to leverage the VSLAM toolbox within ROS2, mapping the environment through the run, then calling a go_home function at the task termination, which would include a best-paths algorithm to move back to the starting location.

V. REFERENCES

[1] D. Ma, Y. Li, T. Ma and A. M. Pascoal, "The state of the art in key technologies for autonomous underwater vehicles: a review," *Engineering,* 7 August 2025.

[2] Y. Zhang, T. Qin and Y. Zhou, "A YOLO-OC real-time small object detection in ocean scenes," *Computer Vision and Image Understanding,* vol. 264, no. 104625, February 2026.

[3] X. Li, T. Li, Y. Zhang, Z. Li, L. Ban and Y. Ning, "GL-VSLAM: A General Lightweight Visual SLAM Approach for RGB-D and Stereo Cameras," *Sensors,* vol. 25, no. 24, p. 7467, 2025.

[4] S. Zhang, S. Zhao, D. An, J. Liu, H. Wang, Y. Feng, D. Li and R. Zhao, "Visual SLAM for underwater vehicles: A survey," *Computer Science Review,* vol. 46, no. 100510, 2022.

[5] S. Sinanan, A. Naser, M. Delatorre, G. P. Glaspell and A. Soylemezoglu, "Autonomous robotics development in Robot Operating System (ROS) 2 Humble," Engineer Research and Development Center, 2025.

[6] M. Grimaldi, D. Nakath, M. She and K. Koser, "Investigation of the Challenges of Underwater-Visual-Monocular-SLAM," arXiv preprint, 2023.

[7] I. Davila, A. Velenzuela, R. Ortiz, J. Choi and F. Yumbla, "Software Architecture and Simulation Interface for Autonomous Underwater Vehicles," in *IEEE Eighth Ecuador Technical Chapters Meeting (ETCM)*, 2024.

VI. APPENDIX A

A. Figures and illustrations

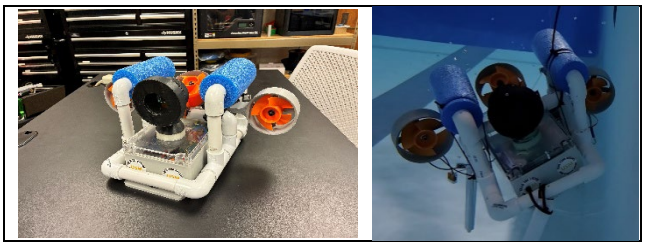

Figure 1.Modified SeaPerch ROV kit.

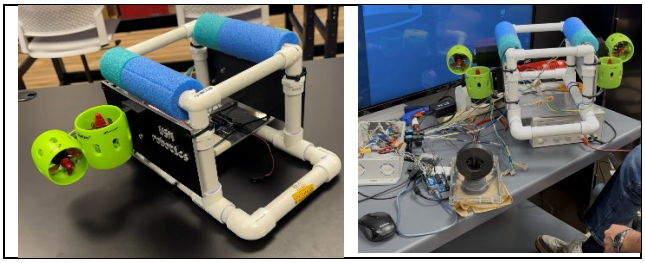

Figure 2. "Killdozer" test vehicle for PID, BLDC, and vision models.

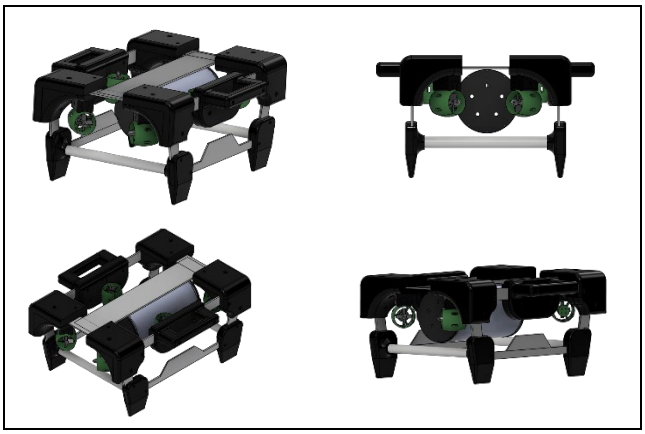

Figure 3. AUV "Marv".

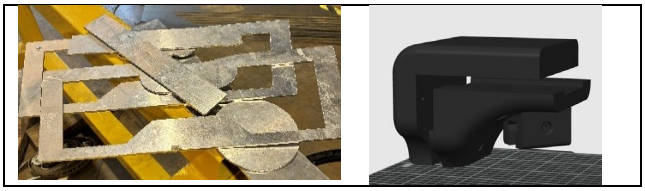

Figure 4. Frame and corners holding the main structure of Marv.

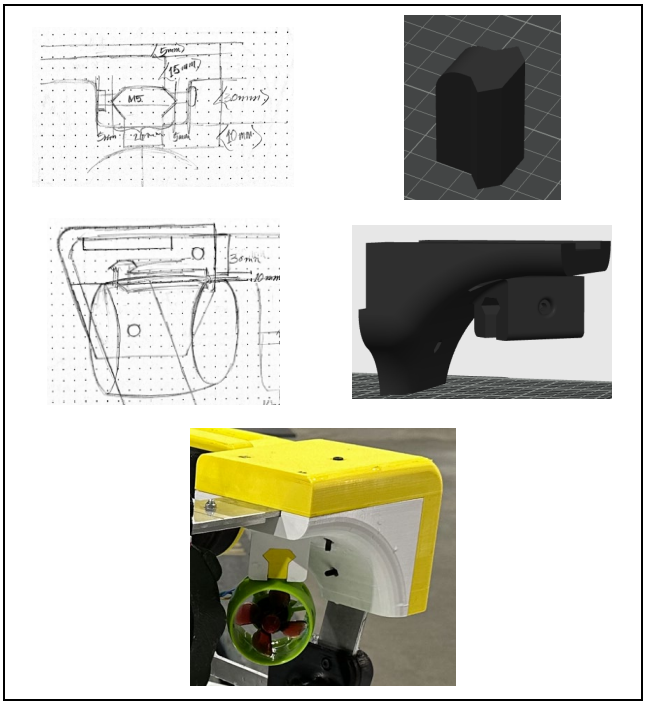

Figure 5. Thruster mount.

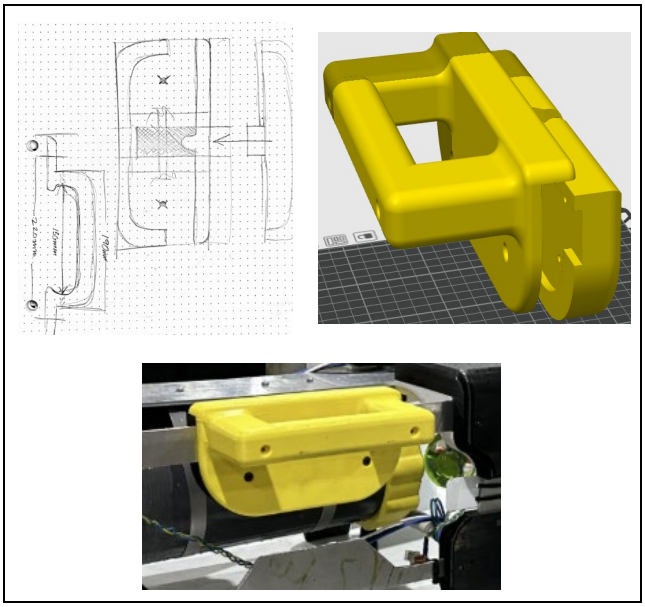

Figure 6. Handles and vertical rails.

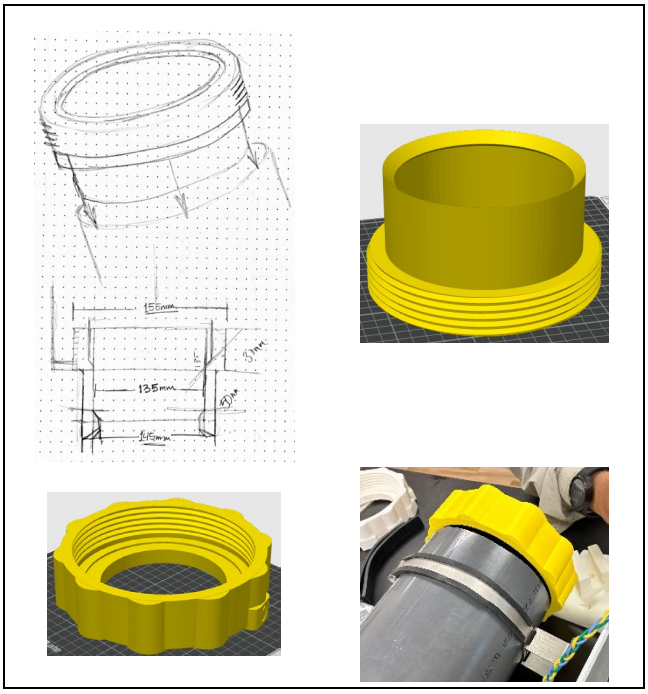

Figure 7. End cap and hull.

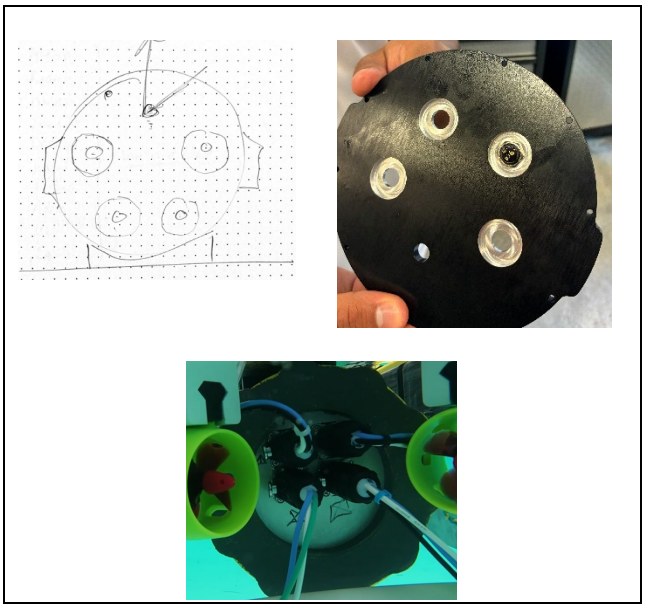

Figure 8. Bulkhead waterproof connectors.

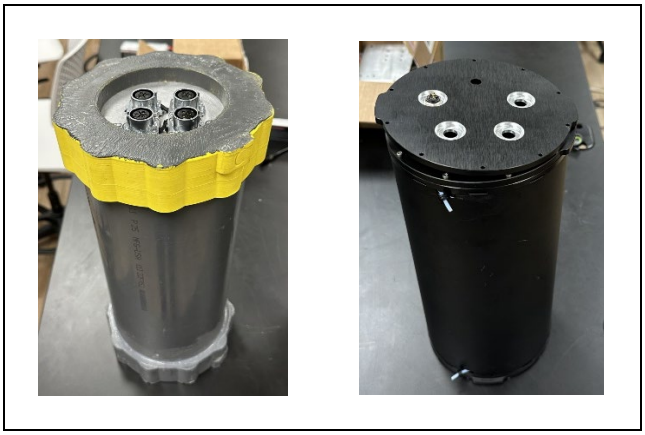

Figure 9. Customized hull designs.

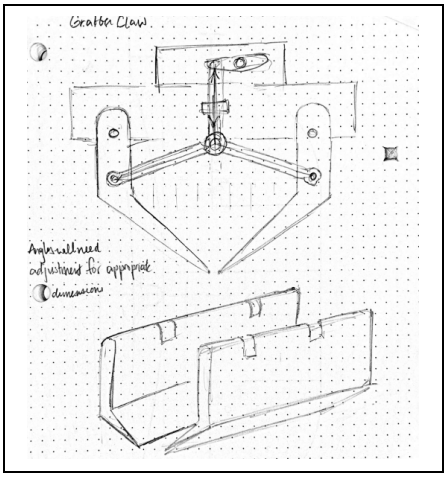

Figure 10. Grabber/arm prototype.

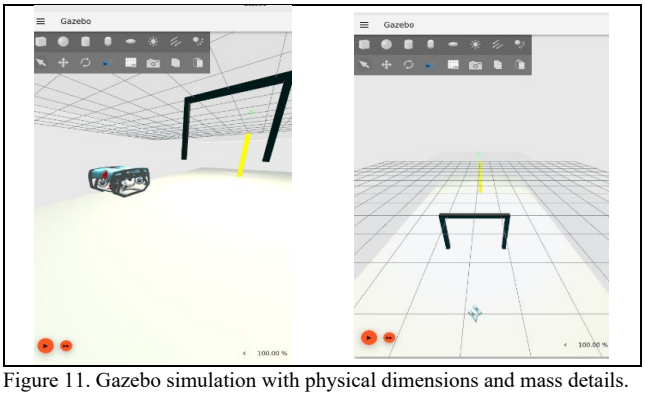

Figure 11. Gazebo simulation with physical dimensions and mass details.

VII. APPENDIX B

A. Control and Architecture Diagrams

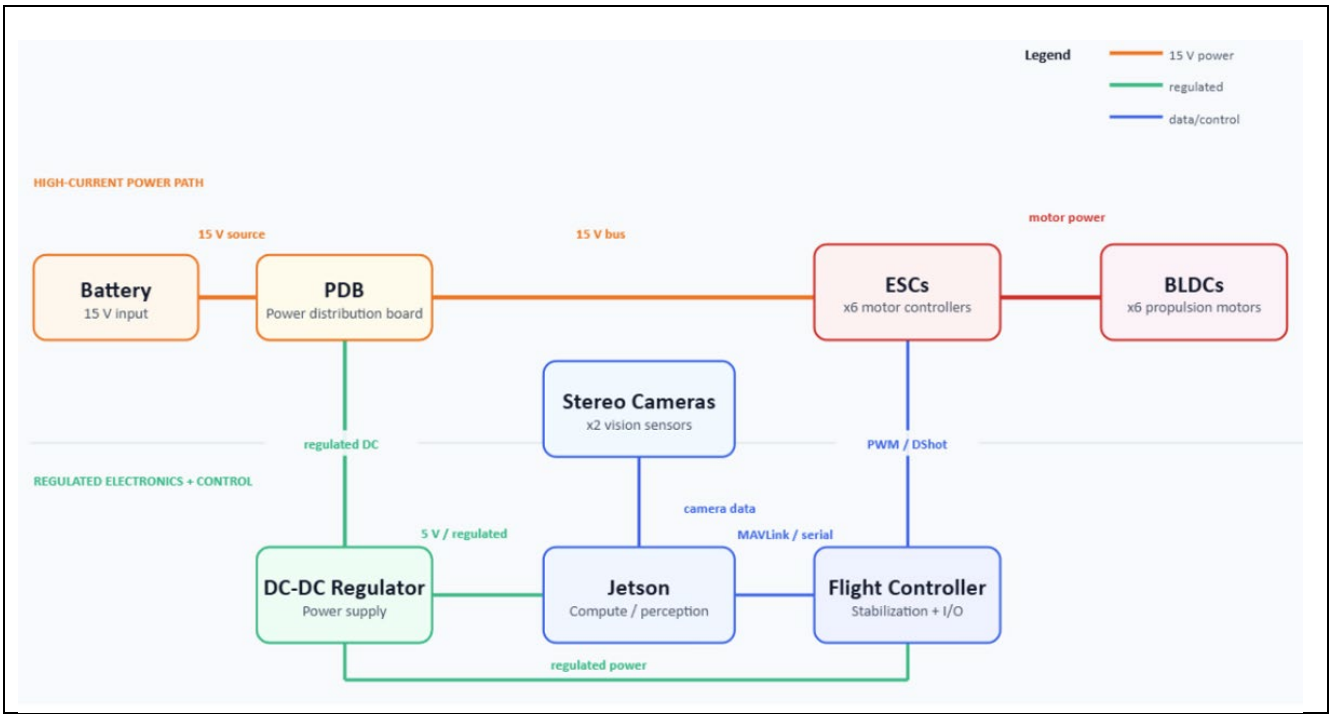

Figure 12. Control Architecture.

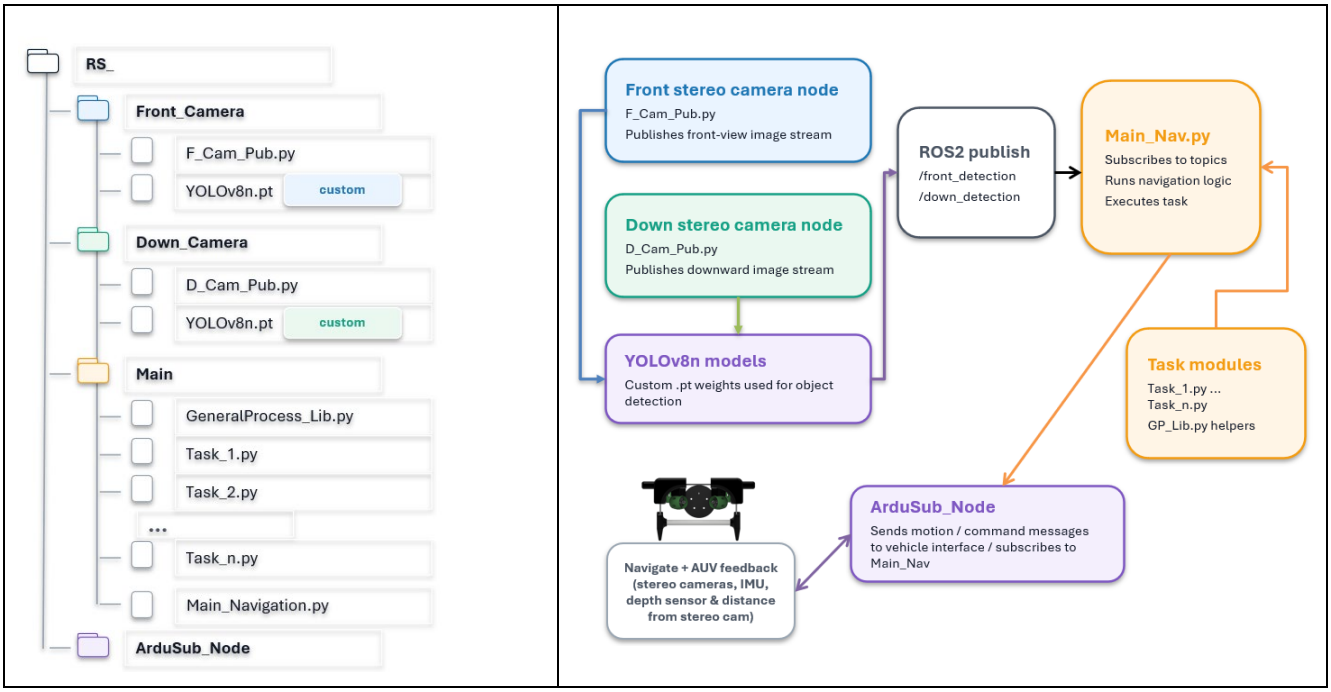

Figure 13. Runtime data flow.

TABLE 1. LIST OF COMPONENTS

Jetson Orin NX 16GB
Power Delivery Board
11.1 3S Battery
Orbbec Dabai Stereo Camera
Blue Robotics 8-bit ESC
ARK flight controller
Cloudpower BLDC
BlueRobotics Acrylic waterproof housing 3"
BlueRobotics Aluminum waterproof housing 6"
BlueRobotics Switch 5A
BlueRobotics Wetlink Penetrator 4.5 HC
BlueRobotics Bar02 & Bar30 Depth/Pressure Sensor
Underwater LED lights
USB C cables
EW-LP12 Waterproof connectors
Misc.: • PETG filament • 6061 aluminum plate • JBWeld • bolts/nuts M5 0.8 hardware