URI Hydrobotics: RoboSub 2024 Technical Report

Benjamin Annicelli, Cameron Tum, Naomi Gitelman, Devin Hunsberger, Elias Newall-Vuillemot, Gabriel Arabik, Willson El Hage, Julian Tamayo, Sam Rebuck, Aidan Donnellan, Sean Cooper, Simcha Schnee, Jose Mercado, Evan Asci, Nathan Agir

I. Abstract

The primary objective of our team this year is to get a robot to compete that is able to achieve all vision based tasks. We achieved this by using very simple steps for our competition strategy complemented to our design of the submersible itself. We designed our submersible with the mindset of a simple yet effective design with a focus on ease of maintenance for any pitfalls we might come across. We also conducted several individual tests in order to ensure an effective design for each component of the robot allowing us to detect any deficiencies in our components. This allowed for an easy final assembly and confidence in our final design.

II. Technical Content

A. Competition Strategy

Our primary strategy for the competition this year is focused on movement based tasks, as this is the first year we've been to RoboSub since 2007. The tasks we plan to do include; Rough Seas, Enter the Pacific, Hydrothermal Vent, and surfacing within the octagon. For these tasks to be done successfully we will be using our arrangement of sensors such as a 9 DOF IMU, and our 2 onboard cameras. (We are planning to add a front-facing Ping 1 Sonar from BlueRobotics but this is still WIP). We will utilize odometry with our IMU for tracking our movement to each task as well as circumnavigating the buoy and passing through the gate. As for locating tasks, we will make use of our front mounted camera for locating the gate, buoy, and sample table below the octagon. Our top camera will be used in looking for the octagon and making sure we surface within its borders. And for watching the side of the gate we pass through for determining our circumnavigation of the buoy.

Each task, and navigation is defined within a state for our state machine. Task states can be labeled as complete, locking them from being re-accessed in the case of being located by our cameras. The different states include; Start: the Tardigrade dives and locates the gate, RoSearch: does a general search of all currently viable tasks or directional tools using our front camera by rotating, ScanSearch: does a scanning movement across the pool floor looking for tasks and/or directional tools, Gate: Tardigrade passes through the gate utilizing our front camera for object detection, and our top camera for determining the side in which is passed through, Buoy: based off the side of gate, rotate the corresponding direction of the buoy, utilizing our front camera for navigation and our IMU/top camera for circumnavigation of the buoy, Surface: Tardigrade utilizes any directional tools and the front camera for navigating to

the Samples table, then using the top camera, surface within the octagon.

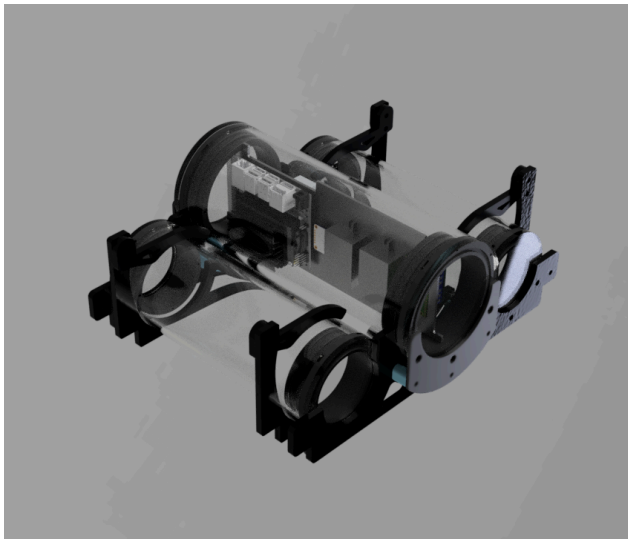
At this time the Tardigrade is not equipped with any grabbers, echo sounders, or torpedo launchers to complete the other tasks.

B. Technical Design

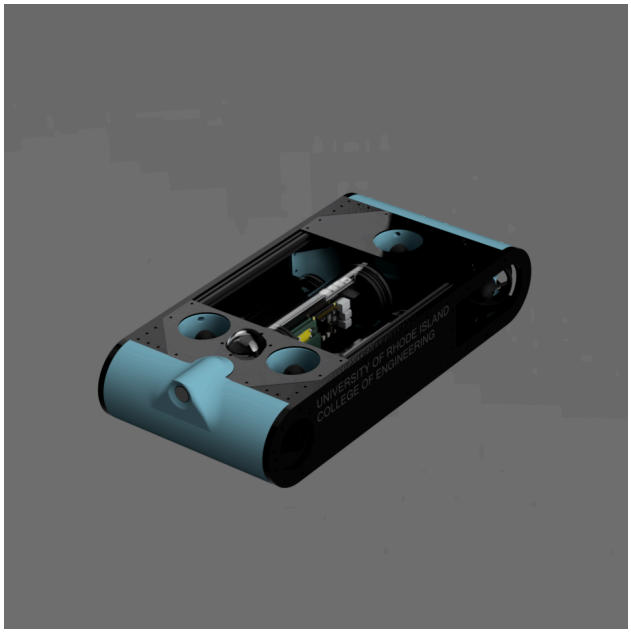
1. Mechanical

Our approach to the robot leverages standardization, modularity, and smart use of stock components towards making a simple and effective design. The backbone of Tardigrade is a T-slot aluminum frame and .25” HDPE side panels. It is to these parts that all the other structural elements are attached by either dovetail nuts to the aluminum or bolted to the HDPE. Parts that are flat such as the top and bottom panels, the bottle brackets, mounting plates for the thrusters, etc. are laser cut from acrylic or delrin. Those that can’t be are resin printed such as the exterior fairings. As to propulsion, Tardigade sports 6 Blue Robotics T200 thrusters: three vertical heave thrusters, two rear surge thrusters, and one forward yaw thruster. For vision, there are two cameras: a 180-degree FOV upward facing camera and a 150 degree FOV main forward camera. Care has been taken to make the exterior of Tardigrade aesthetically pleasing and easy to model, every exterior screw is countersunk and the forward profile is a semicircle. Most of the interior of the robot is occupied by three bottles that house the batteries and computer components. The bottles, flanges, and cable penetrators are all stock from blue robotics but laser cut acrylic caps with different penetrator layouts have been made to suit our needs. Last but perhaps most vital is cut to form R-3312 buoyancy foam giving our robot flotation.

(Bottle mounting system with computer bottle in center and battery bottles to the sides rendering)



(Full AUV assembly)



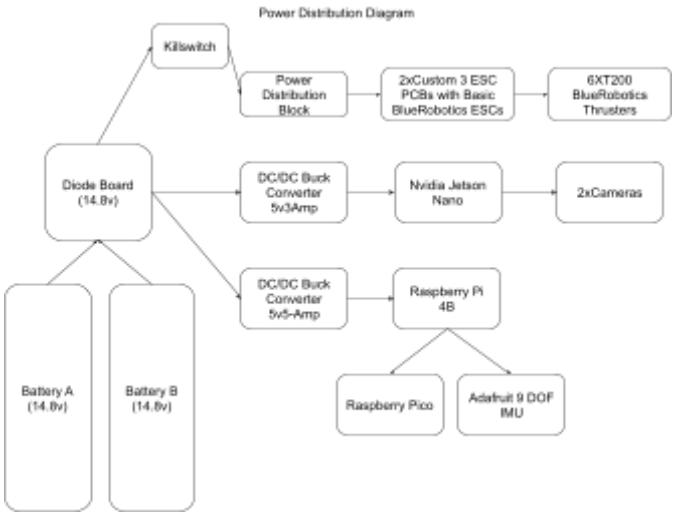
2. Electrical

We utilize two Blue Robotics batteries, each providing 14.8 volts with a current draw of 60 amps and a maximum burst current draw of 132 amps. These batteries are connected

to a main power wire, which then splits into two separate wires, each linked to individual diodes on a custom-printed PCB board. From the PCB board, our kill switch is connected to a connected power distribution board that distributes power to the thrusters. This setup ensures that pulling the kill switch removes power from the thrusters while leaving the rest of the system operational.

(Computer bottle rendering)

Additionally, two DC-DC converters are separately connected to the diode board, providing power to our Raspberry Pi 4B and NVIDIA Jetson Nano without risk of damage. The Raspberry Pi 4B acts as the system's central processor, sending commands to and interpreting data from all onboard devices. It powers and communicates with a Raspberry Pi Pico, enabling thruster control via Blue Robotics bidirectional ESCs (electronic speed controllers) for our Blue Robotics T200 Thrusters. The NVIDIA Jetson Nano, also connected to the Raspberry Pi 4B, transmits information from our front and top cameras. Lastly, an Adafruit 9-DOF Absolute Orientation IMU is attached to the Raspberry Pi 4B for localization and odometry purposes.



3. Software

Sure, I can refine and check the technical accuracy of your statements. Here's the revised version:

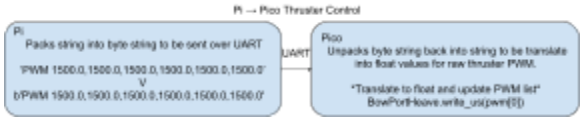
The core processing unit of our robot is a Raspberry Pi 4B, accessed remotely over a Secure Shell Protocol (SSH) connection via a predefined router. The Surface laptop connects to the router via Ethernet for stable communication. We are using Ubuntu 20.04 Server on the Pi for a headless, lightweight operating system, with ROS-1 Noetic Ninjemys as our primary Robot Operating System (ROS) library, due to ROS-MVP's ongoing ROS 2 development.

The Raspberry Pi Pico functions as a microcontroller for our thrusters, communicating through byte strings over a serial UART interface via USB to the Pi. These byte strings are unpacked, parsed, and used to generate PWM signals for each thruster via a continuously running thread loop.

For object detection, we employ two YOLOv8 models—one for the front camera and one for the top camera. Our datasets are annotated and labeled on RoboFlow to handle the large volume of data efficiently. Currently, we are training for five classes: Gate, Buoy, SamplesTable, Path, and Octagon. The models are trained locally on our systems. Our preprocessing techniques include random removal, random Gaussian blurring, and random rotations. The dataset is split into a 70-30 train-test ratio. The models are hosted on an NVIDIA Jetson Nano, which communicates with the Pi over Ethernet through SSH, providing class predictions, confidence scores, and (x, y) coordinates. This communication is facilitated by ROS' multi-machine package, with the Pi acting as the master and the Jetson Nano as the slave.

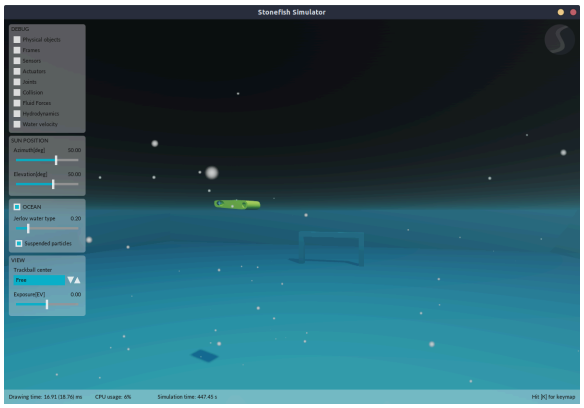
For maritime navigation and localization, we utilize a combination of an Extended Kalman Filter (EKF) and the Madgwick Filter. The EKF smooths observed and estimated positions to provide a more accurate dead reckoning trajectory. The Madgwick Filter assists in determining orientation using data from a BNO085 IMU and Magnetometer, offering close-to-accurate measurements of position and pose in the absence of Sonar, DVL, or depth sensors.

(Pi to Pico Diagram)



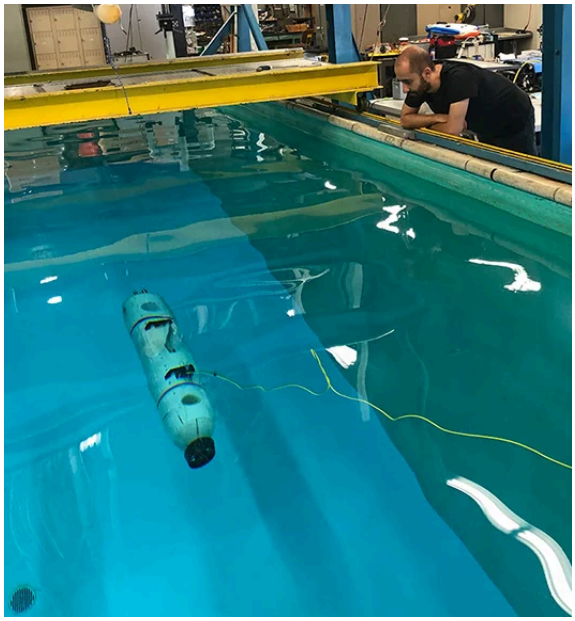
C. Testing Strategy

At this time, most of our testing takes place in the Stonefish Simulator, specifically the Marine Vehicle Package (MVP) wrapper for Stonefish. In Stonefish we have taken the time to make an accurate environment of the Woollett Aquatics Center main pool, such as its length, width, depth, as well as water clarity. Stonefish makes use of your own ROS environment for “realistic” data acquisition such as positional, video, sonar, and accurate hydrodynamic physics. Stonefish makes use of OpenGL and Qt for the front-end, and the Bullet Real-Time Physics Engine for back-end processes.



We also have access to the URI Bay Campus’ testing pool. (pictured below with

the URI SOSLab's AlphaAUV).



Stonefish-MVP makes usage of your current ROS environment so basically any code written works directly within the simulation environment. For example; you can add noise to sonars if you wish to accurately represent the actual on-board sonar. This is all taken care of using MVP and Stonefish's extensive middle-ware.

III. Acknowledgements

We would like to thank our main sponsor Cadence for their irreplaceable industry standard training on PCB design.

We would like to thank NIUVT for their funding which allowed us to travel to the 2024 RoboSub and acquire electrical hardware.

We would like to thank the College of Engineering for their funding which allowed us to travel to the 2024 RoboSub competition.

We would finally like to thank the College of Ocean Engineering for their funding which allowed us to acquire hardware to build our AUV.

IV. References

- E. C. Gezer, M. Zhou, L. Zhao and W. McConnell, "Working toward the development of a generic marine vehicle framework: ROS-MVP," *OCEANS 2022, Hampton Roads*, Hampton Roads, VA, USA, 2022, pp. 1-5 .
- P. Cieślak, "Stonefish: An Advanced Open-Source Simulation Tool Designed for Marine Robotics, With a ROS Interface," *OCEANS 2019 - Marseille*, Marseille, France, 2019, pp. 1-6