Design of the Triton Autonomous Underwater Vehicle for the International RoboSub Competition

Dvir Hilu (Team Captain)*, Kota Chang*, Jacob Cronin*, Kevin Huang*,
Pavitar Kalra*, Viktor Moreno*, Ryan Murch*, Angie Pinto*
**UBC Subbots*
University of British Columbia
Vancouver, Canada
Email: ubc.subbots@gmail.com

***Abstract*—UBC Subbots's submission to RoboSub 2021 is the Triton Autonomous Underwater Vehicle (AUV). Novel elements designed in-house include mechanical components, such as our enclosure and pull-out mounting plate, and electronics, such as our battery management system. Our software pipeline, running on a Jetson TX2, takes advantage of ROS2's modular design, introspection tools, and integration with the Gazebo simulator. With the constraints set by the COVID-19 pandemic, we focused heavily on development of realistic and physically-informed software simulations for AUV control, computer vision, and sound localization, laying the groundwork for testing and verification of future iterations of our AUV.**

***Index Terms*—robotics, navigation, autonomous, controls**

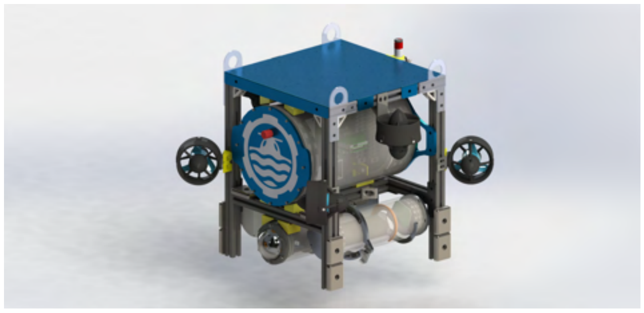


Fig. 1: Full robot CAD render

I. Competition Strategy

Our competition strategy comes from prioritizing adaptability and reliability. As a fairly young team with limited resources, it was challenging to aim for all competition tasks right away. Our strategy is to ensure that the robot can reliably complete tasks that are earlier in the competition course, while being mindful of all the design changes that will be required for future iterations. Although the specific competition tasks are unknown every year, there are consistencies such as path finding, recognizing objects, manipulating objects, etc. With this in mind, we prioritized general functionalities such as object classification underwater, general actuation and sensor systems, propulsion system with 4.5 degrees of freedom, and a sound localization system for all tasks requiring pinger detection. If competition were to be held offline, we would expect our robot to pass the gate and follow the path marks to the first task. Although the actuation system is incomplete, our adaptable design approach will let us improve our robot for future competitions to tackle more and more sophisticated competition tasks.

II. Novel Design Elements

A. Main Enclosure

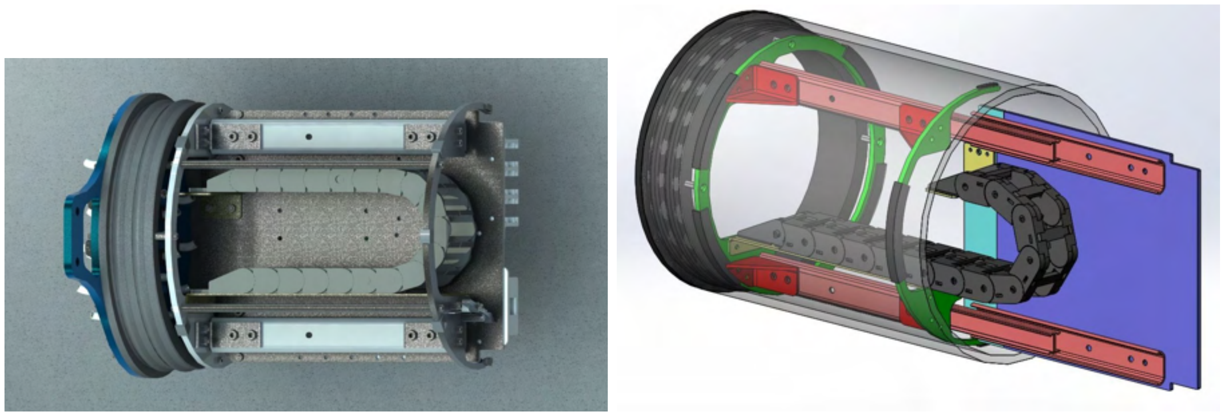
1) Design Goals: The goals of the main enclosure are to protect critical operational components, allow for easy maintenance and to organize and secure cabling, while providing a significant buoyant force to the autonomous underwater vehicle (AUV).

The enclosure measures 8.5” in diameter 2.5” thick with aluminum end caps. Each end has a double seal to prevent leaks. On one end of the enclosure, incoming and outgoing cabling is routed. These are secured with an epoxy resin while also being fitted with O-rings to provide both strain relief and waterproofing. Handles extending about 2” from the enclosure were included for easier manipulation when installing the enclosure. The other end can be opened to access the inside electronics. It includes a pressure valve to equalize air pressure when on the surface

2) Mounting Plate: The mounting plate mechanism was designed to reduce strain from waterproof connectors and ease of access during maintenance in mind. The dual stainless steel rails allow for extending the aluminium mounting plate out from the enclosure, allowing access to components from a more convenient position. The vertical orientation allows for better loading on the slide rails with minimized deflection and sturdy component mounting. The plate can also be removed easily, if needed.

Located directly behind the mounting plate is a cable carrier. Proper strain relief was implemented on both the moving and stationary ends of the cable carrier to ensure that connectors and cables were not unnecessarily strained when moving the plate, as shown in Fig. 2b.

This also reduces the chance for failure of the waterproof connectors on the front of the enclosure both in operation and during maintenance. Securing this assembly are aluminum brackets attached on aluminum rings pressed against the acrylic enclosure, cushioned with rubber. At the access end of the enclosure is a small latch to secure the rails during operation.



(a) Pull-out mounting plate in its stowed position. (b) Pull-out plate when deployed.

Fig. 2: CAD renders of the pull-out mounting plate.

3) Component Layout: Heat was foreseen as an issue from the Electronic Speed Controllers (ESCs) (top left of Fig. 3), motor drivers (top middle of Fig. 3), and main computer (green block on right of Fig. 3). These components were spread out to avoid overheating. The ESCs were also predicted to produce significant electrical noise, which led to another design consideration: signal noise.

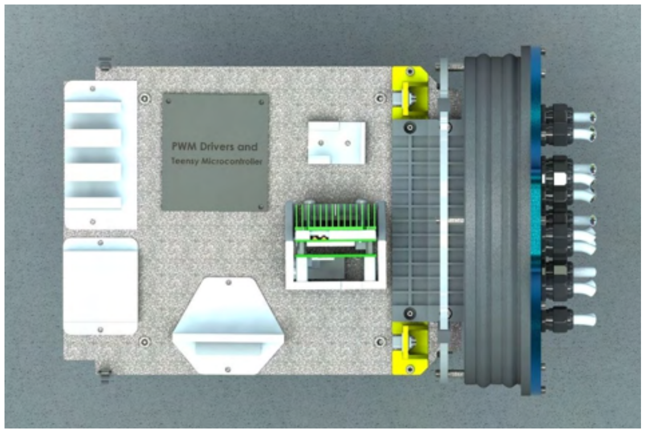


Fig. 3: Overhead view of the mounting plate and component mounts.

To minimize noise, we first made sure that sensitive components such as the Inertial Measurement Unit (IMU) (bottom left of Fig. 3) and the surface communications module (bottom middle of Fig. 3) use data cables that are less susceptible to electronic noise. We also made sure that incoming high power wires stay physically further away from the sensitive wires which reduces the potential for interference in signals, increasing overall reliability. Incoming signals from other enclosures would also be able to pass underneath the main computer to avoid high power cabling.

The volume of cabling was another consideration when configuring the layout. Wide gaps between components and relatively high clearance above the plate allows for flexible cable routing. A separate cable connection plate is used to secure cabling when the rail system is being moved. The center of the plate also provides additional space for USB hubs to be added.

Mounting hardware was 3D printed with Acrylonitrile Butadiene Styrene (ABS) on an Fused Deposition Modelling (FDM) printer to custom fit components. Since the mounting plate was vertically positioned, mounting hardware had to allow for components to be cantilevered. The main computer was the largest component by volume and mass, requiring a longer protrusion from the plate. It required access to multiple connector ports, leading to its current design. The ESC mounting hardware is also taller than necessary to facilitate sufficient heat dissipation. Other components were not as critical in heat dissipation nor had as much mass, so conservative cantilevered designs were sufficient.

B. Thruster Configuration and Buoyancy

1) Thruster Geometry: The AUV consists of 6 Blue Robotics T200 Thrusters to attain 4.5 degrees of freedom (DOFs) (Fig. 4). Four thrusters are positioned in plane with the centre of mass at $45°$ to the diagonals. This provides control over the surge, sway, and yaw allowing the AUV to have full control over movement in the XY subspace. The final minimum degree of freedom required for the competition is control over heave. Two vertical thrusters straddling the main enclosure are placed in plane with the centre of mass to provide the vertical control. This configuration had the added benefit of providing some control over roll. The selected configuration was chosen as thrusters could not be placed in front of the main enclosure flanges without preventing access to the enclosed electronics. The two degrees of freedom for controlling roll and pitch were determined to not be critical to the function of the AUV for the competition tasks. Although some level of control over these degrees of freedom would be beneficial to the overall stability of the robot, they were removed as a tradeoff for reduced cost.

2) Buoyancy: As a direct result of the decision for not having significant control over the roll and pitch, a large moment arm was desirable to prevent unwanted movement in these directions. Enclosures naturally create a large buoyant force while the frame and other heavy components (ie. batteries) are main contributors to the overall centre of mass. These components are positioned such that the larger enclosures are higher up on the AUV frame, while the batteries are near the bottom. The position of these components are limited by various design and space considerations. This results in a centre of mass (COM) and centre of buoyancy (COB) that are not in the desired positions for maintaining level operation in the XY plane. Weights are placed along the aluminium extrusion towards the bottom of the robot to move the COM and increase the righting moment. While foam blocks are placed along the top plate of the AUV to tweak the COB and maintain near neutral buoyant forces.

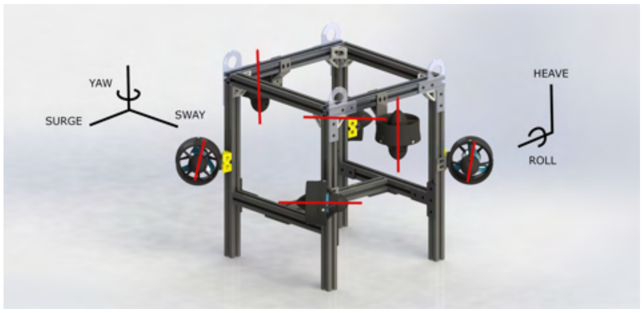


Fig. 4: AUV thruster layout illustrating degrees of freedom

Due to uncertainty in the SOLIDWORKS model, the COM of the robot will ultimately be determined using tension scales attached to four points on the AUV for multiple faces. The weights placed at the bottom of the robot will be adjusted to change the location of the COM. In turn, the thrusters will be adjusted to align with the COM to prevent unwanted pitch and roll. Calculations for COM do not account for the added mass of the water that will be present when submerged. This is an acceptable choice due to the open nature of the frame leaving minimal space for trapped water to accrue. Although some amount of water will be carried forward in the robot during motion, we neglect the head generated as the effect will be negligible at the operational velocity.

C. Control System

1) AUV Model: The dynamics model for our vehicle was formulated based on the work of Fossen [2], Vervoort [3], and Gonzalez [4], and describes the equations of motion, damping, and environmental wrench contributions for low-speed, submersible vehicles. The second-order equation describing the robot's motion in $\mathbb{R}^3$ is presented in Eq. 1. This expression depends on environment forces (τ), gravity (g), buoyancy (η) in the world-frame position, velocity (v) and acceleration ($\dot{v}$) vectors, the mass (M), Coriolis (C) and damping (D) matrices. Using knowledge of our vehicle's behaviour in competition settings, we were able to make practical simplifications based on our vehicle's mechanical characteristics and modify the model accordingly. The usefulness of the AUV mathematical model is twofold: it provides us with an accurate representation on the AUV, such that we can linearize the equations and apply a generic control algorithm to debug and tune our motion control system, and it also provides a means of calculating restoring forces in our chosen simulation environment, Gazebo.

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \tag{1}$$

To make the model more practical, we considered that the AUV was designed to travel at low speeds and is near-symmetrical about the XY, YZ and XZ planes, which allowed us to make assumptions to simplify the model, as was done by Vervoort [3]. We assumed that with a mass matrix calculated with respect to our vehicle's centre of mass, we were able to neglect added mass and cross-product contributions, which allowed us to treat the mass matrix, and by extension the damping matrices, as diagonal matrices. Using these simplifications, we were able to linearize the model with confidence that around certain equilibriums, our model would be accurate.

When designing the control system, we also noted that our vehicle dynamics imposed no restrictions on the "pitch" axis of motion (rotation about the Y-axis), whereas our thruster configuration prohibits us from controlling motion along that axis. As well, the robot's current limitations added design constraints, since our combined thruster configuration can only draw so much power from the batteries at any time. We also determined that we would not be constrained by the computational performance of our on-board computer, the Jetson TX2, which means we can consider real-time compensation algorithms.

2) Control Model: Using the previously mentioned assumptions and considerations, we selected a Proportional-Derivative (PD) controller and a Linear Quadratic Regulator (LQR) to aid in controller tuning. With this approach, we used the cost function to directly address our robot's constraints by applying a moderate cost to each of our 5 degrees of freedom and a large cost on the pitch motion over which we have no control.

The controller implementation uses a path planning algorithm to generate position setpoints and a velocity profile for each step, then uses a generic feedback model to perform PD control on the desired setpoint using the gain values determined by the cost function. Using the velocity profile and a thruster transformation matrix, we are able to calculate the force contributions by each thruster to achieve the desired trajectory in each iteration of our control loop.

3) Architecture: Our software architecture was designed using the open-source ROS2 framework, which allows us to implement our AUV's necessary functions as modular nodes that can run concurrently on our AUV's Jetson TX2. Telemetry and sensor data are passed between nodes as messages, which can be easily monitored for introspection and debugging. ROS2 is also language-agnostic, so we can pass messages between nodes written in different languages. For applications

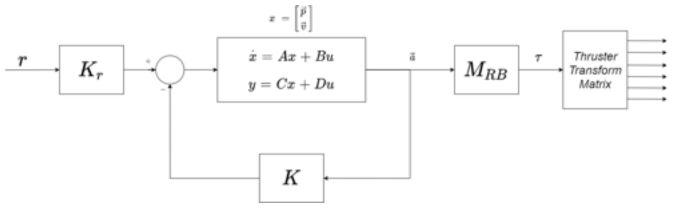

$x = \begin{bmatrix} \vec{p} \\ \vec{v} \end{bmatrix}$
r
K_r
+
−
$\dot{x} = Ax + Bu$
$y = Cx + Du$
$\vec{a}$
M_{RB}
τ
Thruster Transform Matrix
K

Fig. 5: Block diagram of the PD controller used by the Triton AUV.

requiring low-latency processing, we use C++, while Python is used primarily as a high-level interface for managing our pipeline. Our custom pipeline manager can be configured to execute arbitrary sequences of actions, starting and stopping nodes based on published feedback according to criteria we define.

4) Testing and Verification: With little pool access due to the COVID pandemic, our team made the decision to shift our focus to developing our simulation environment. Simulation provides us with a cheaper and safer way to test our AUV, as well as ample synthetic data.

To develop and validate our robots control system, we created a Simulink project to implement the control algorithm using our linearized dynamics model. Once satisfied with our control system's performance under theoretical conditions, we then developed a simulation environment and 3D rendering of our vehicle to further test our control system. This approach allowed us to incrementally develop the system and decouple different stages of the design.

The simulation environment we deployed was developed using the open-source simulation tool Gazebo, which allowed us to create a simulation description format (SDF) file representing our robot. The SDF description allowed us to import an STL-format model of our robot from SOLIDWORKS and apply mechanical properties such as inertia and damping to generate realistic restoring forces as the vehicle moves through space. Using this environment, we developed camera, position, gyroscope and depth sensor emulators, as well as thruster driver emulators in the form of plugins that interact with our control pipeline. We implemented buoyancy and hydrodynamic force plugins that use the second-order equations of motion for the AUV, as well as position, velocity and acceleration values at each iteration of the simulator's update loop. These calculate the environment forces acting on the AUV at any given time.

We took advantage of Gazebo's seamless integration with ROS2 to bring the Simulink project and the simulation environment together. As shown in Fig. 6, our validation process involved implementing our equations in the Simulink project, then implementing the same equations in our real-time control loop and hydrodynamics plugin used by Gazebo. With ROS2, we are then able to launch Gazebo alongside our autonomous control pipeline and communicate with the simulation environment exactly as we would communicate with physical hardware. This approach lets us verify and visualize the vehicle's behaviour with confidence before moving on to physical testing.

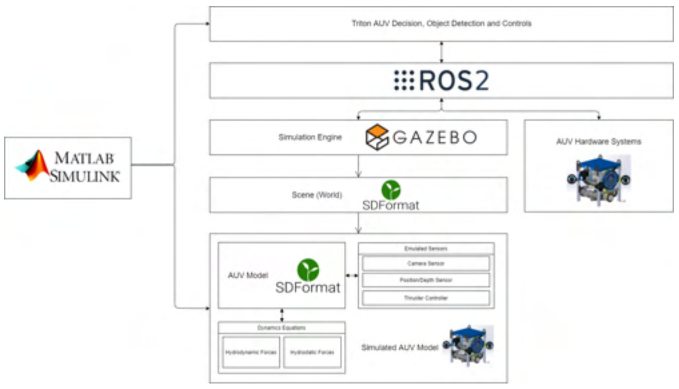


Fig. 6: Software verification framework connecting Simulink control to the Gazebo simulation environment using ROS2 for message-passing.

D. Computer Vision System

1) Underwater Synthesis: Synthetic data has become increasingly used in robotics and machine learning, as it offers a solution to the issues of data collection and data variety. However, for synthetic data to be useful, it must be similar enough to reality. Measures must be implemented to reduce the issues of sim-to-real domain transfer.

In underwater images, the farther away an object is, the greener it looks. This phenomenon is caused by light behaving differently underwater due to scattering and absorption. If we want our simulation to look realistic, we need to model the behaviour of light. A key component of our simulation is underwater image synthesis, which takes an RGBD image

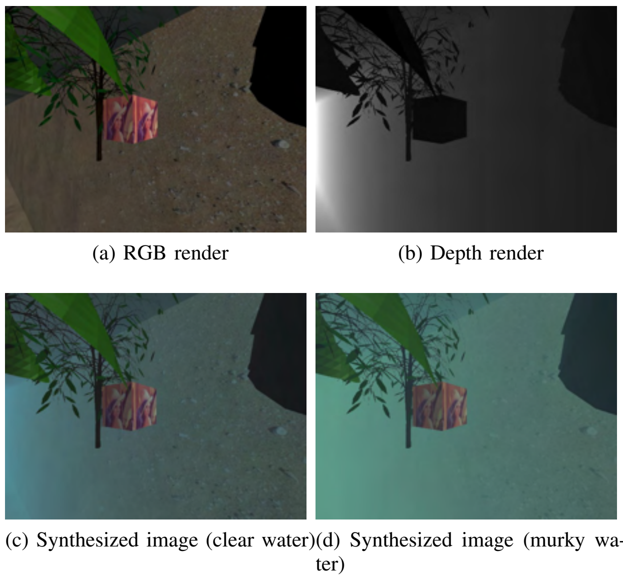
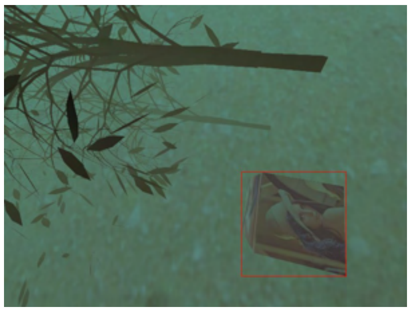

(a) RGB render
(b) Depth render
(c) Synthesized image (clear water)
(d) Synthesized image (murky water)

Fig. 7: Results of underwater synthesis from a) RGB and b) depth renders, using parameters of c) clear and d) murky water.

(colour and depth) rendered in our simulation and generates an RGB image of an underwater scene. Our implementation, based on the work of Ueda et al. [1], models the physical properties of light, allowing us to simulate a variety of underwater environments by adjusting the attenuation coefficients over the visible spectrum of light (ten different water types are currently implemented). This physically-informed model ensures our synthetic data covers a range of different conditions which will help our models generalize to differences in water quality. Fig. 7 demonstrates the synthesis of two different water types from the same RGB and depth images.

The gate and marker tasks require detection of orange objects. For these tasks, we extract features in the image based on colour information in the HSV and LAB colour spaces, which better model perceptual changes in colour than RGB. This is tested against real and synthetic underwater images to ensure our segmentation is accurate and robust against noise.

Many of the tasks involve recognition of printed pictures. For this, we decided to use a YOLO object detection model, which can not only detect multiple classes of objects, but give their bounding boxes as well, allowing us to localize the object relative to our AUV. This is where underwater image synthesis becomes invaluable, as we want to be confident the effects of water won't negatively affect the reliability of our detection. In tandem with our underwater image synthesis, we designed a custom plugin for Gazebo that stores bounding boxes of objects to automatically label data; together they allow us to generate large datasets for training object recognition models. Fig. 8 shows examples of generated images with bounding boxes.


(a) Object without occlusion

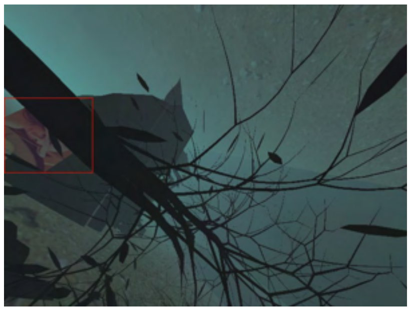

(b) Object with occlusion

Fig. 8: Generated underwater images with automatically labelled bounding boxes (visualized here in red).

E. Battery Management System

Our focus in power management was toward battery and system safety. By focusing on these, we could mitigate battery failure, giving us more potential to succeed in the water. Due to the power requirements of the thrusters and the noise they generate, we designated a 4S LiPo battery (14.8V) specifically for them. All other components in our system are powered by a smaller 3S LiPo battery (12V). This eliminates noise from the thruster system impacting our more sensitive low-power components.

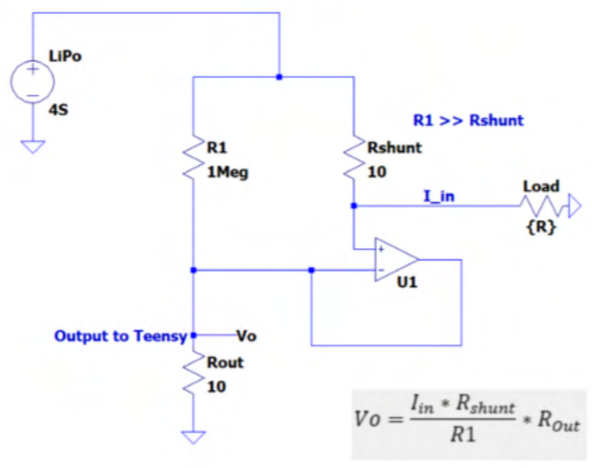
Our design process started with determining the areas most susceptible to failure. We decided to focus on three main areas. First, cell voltage monitoring, to ensure no cell in the battery drops below the safe limit voltage, protecting and increasing the lifespan of our batteries. Second, current monitoring, to ensure currents higher than that rated for our wires and connectors is not surpassed. Lastly, temperature monitoring, to ensure unsafe temperatures are not reached within our battery enclosures, increasing the lifespan and mitigating potential for battery failures.

1) Cell Voltage Monitoring: To monitor voltage, we needed to isolate each battery's cell voltage and ensure it never dropped below the safety requirement. Our design isolated each cell voltage by using differential op-amps and comparing the resulting output to a reference voltage. Should any cell be below it, a relay leading to the rest of our robot would open and act as a kill switch, preventing any damage to the batteries and other system components.

To test our design, we simulated battery cells using a power supply and a voltage divider circuit. Adjusting these voltages let us mimic a single cell's voltage decreasing, and monitor the output to the relay. Testing of our initial design was promising, as the output to our relay was as expected. At high voltages, the relay was closed, and once any cell dropped below the reference voltage, the relay would open. However, due to a limited current output of the AND gate supplying the signal to the relay, we found that the relay could not reliably close. To resolve this issue, we added a BJT amplifier to increase the current to the relay so it could remain in its steady state. We also added a fly-back diode as a layer of safety between the relay and BJT to prevent any back EMF entering our circuit should the kill switch be triggered.

2) Current Monitoring: To monitor current, we wanted a simple design that would act as a risk mitigator, pictured in Fig. 9. To design the circuit, we leveraged a negative feedback op-amp to keep the voltage on two nodes identical. Then, using a shunt resistor in parallel with a larger reference resistance, we could monitor changes in voltage and measure its relationship to current to get a reading of how much current was entering our system.

Similar to the voltage monitor, our first design entailed monitoring these values electrically. However, we soon determined that a kill switch would not be ideal since triggering that would immediately put us out of the competition. Instead,



$$Vo = \frac{I_{in} * R_{shunt}}{R1} * R_{Out}$$

Fig. 9: Circuit diagram for the current monitor circuit

we wanted some way to monitor the current and adjust robot performance accordingly. To handle this, we decided to include a small microcontroller in the battery enclosures that could communicate with the Jetson TX2. The TX2 could then adjust thruster operation based on the operating current. As an extra safety precaution, we are also adding a 12A fuse into our system to act as a last line of defense kill switch. In addition, we are going to implement an LED indicating if this fuse blows, letting us quickly swap it out in competition.

3) Temperature Monitoring: The low power system's dedicated 3S battery is extremely unlikely to overheat so it is not necessary to monitor temperature for that battery. We decided to solely focus only on the high power 4S battery. The maximum safe operating temperature of the battery is 70° Celsius. To monitor temperature, we are using the integrated 3435K NTC thermistor. Using this thermistor is ideal since it's positioned between the battery cells, thus giving us the most accurate temperature reading. We will be sending this thermistor data to the TX2 via the microcontroller in the battery enclosure. To mitigate chances of damage or overheating, at 55° Celsius we will begin adjusting the operation of thrusters to give the batteries a chance to cool down. At 70^circ Celsius we will cut off operation.

F. Pinger Localization

In order to complete tasks marked by pingers in the competition, we require a system that allows us to accurately detect the location of the pingers at the desired frequency and navigate to them. We created an in-house Python simulator for this sound localization system to enable sub-system level testing while offline testing is not accessible.

The overall architecture for the sub-system is shown in Fig. 10. Specific models or part numbers for the components are yet to be determined as this system is still under development, however, the general outline has been designed with an analog interface and a digital processor which would allow us to conduct all necessary calculations for localizing the sound pingers.

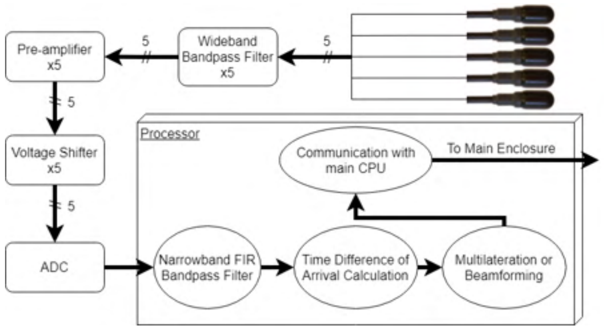

Pre-amplifier x5
Wideband Bandpass Filter x5
5
5
5
Voltage Shifter x5
5
ADC
Processor
Communication with main CPU
To Main Enclosure
Narrowband FIR Bandpass Filter
Time Difference of Arrival Calculation
Multilateration or Beamforming

Fig. 10: Pinger localization system architecture.

1) Analog Interface: The system is designed to work in isolation, and will be inside its own enclosure to allow it to be integrated with either Triton or future revisions. As a result, the enclosure must be quite small and one of the main requirements for the analog interface circuitry is a small form factor.

Once detected by five AS-1 hydrophones, the signals pass through a fixed frequency, 15-45kHz bandpass filter. The purpose of this filter is to eliminate most non-pinger frequencies before the pre-amplifier, thus avoiding interference from intermodulation products stemming from the amplifier's nonlinearity. Since the filter encapsulates all possible pinger frequencies, there is no need to make the filter frequency adjustable, saving a significant amount of space. A frequency adjustable, narrow-band FIR bandpass filter will be implemented in software to take care of distinguishing between the different pinger frequencies.

Most teams typically use a gain-adjustable amplifier, but our team is planning to implement a fixed-gain low noise amplifier. Though this would result in the amplifier saturating as Triton moves closer to the pinger, All of the localization methods we are currently exploring only require to maintain the integrity of the zero-crossing of the wave. Hence, saturation would not pose a significant issue.

Lastly, the system will include a voltage shifter that will shift the AC wave to a level that could be accepted by our analog to digital converter (ADC).

2) Localization Method: At its current iteration, the system is designed to locate the relative angle of the pinger on the xy plane. The pressure sensor on board provides accurate measurements of the AUV's height (z-axis), so with only the angle information we can simply steer at the detected angle and rely on our other sub-systems to detect the competition objective once we are close.

At the moment, we are considering two techniques: multilateration and beamforming. Multilateration involves calculating the time difference of arrival (TDOA) between the signals and then triangulating the location of the pinger. Since the waves are in the far-field regime and the hydrophones must be quite close together for the TDOA calculation, this method does not always produce accurate distance readings. At the moment, we calculate the TDOA via cross correlation and

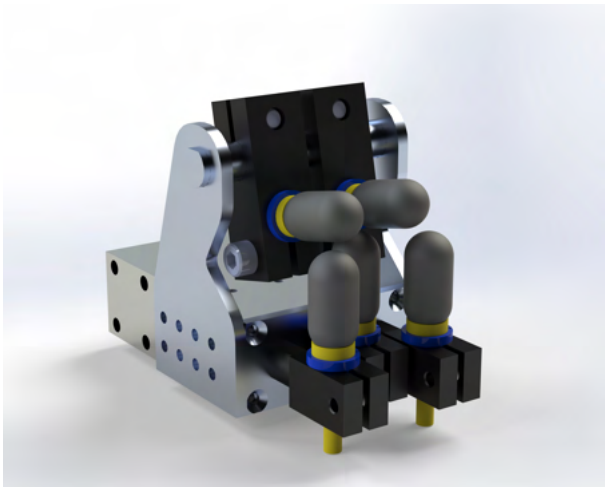
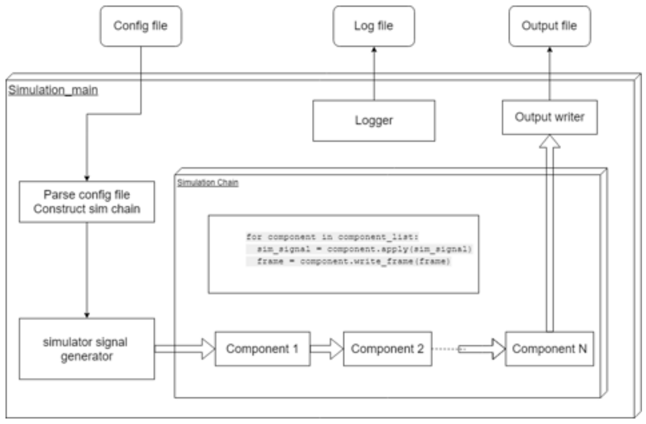


Fig. 11: CAD Render of the hydrophone mount.

then use a nonlinear least squares (NLS) reconstruction to find the position of the pinger. This method does not yet account for signal reflections, and we are currently researching into methods that will allow our localization to perform robustly under multipath.

In the case we decide to move forward with multilateration as our localization method, we have designed an adjustable mount (Fig. 11) for the hydrophones that will allow us some freedom in tuning their positions to the ideal geometry. The design consists of several telescoping mounts which allow for three hydrophones to translate in plane, with two additional headphones that can be rotated to vary the distance from the in plane hydrophones. This design additionally allows for the side panels to be waterjet cut, which provides short turnaround for modification in the acoustic centre positions.

Since we are only interested in the direction of the pinger, we are also considering beamforming, which involves placing all hydrophones in an evenly spaced array and finding the direction of maximum superposition of all hydrophone signals.



```
for component in component_list:
  sim_signal = component.apply(sim_signal)
  frame = component.write_frame(frame)
```

Fig. 12: Pinger localization simulator architecture.

3) Simulator Architecture: In order to test our initial designs of the system, we have built a custom Python-based simulator. The simulator architecture, pictured in Fig. 12, emphasis modularity and flexibility. With our current architecture, we design each system module as its own individual “component”. These components are abstractions of different portions of the system (certain circuit designs, algorithms, etc.), and could be chained together or swapped to form different simulation chains. In its current structure, the simulator allows us to create different configurations of this simulation chain which allow us to test different configurations of the system as a whole. Aside from directly implementing various components in Python, we are using LTSpice (integrated with Python) to simulate all of our circuits and Bellhops to simulate sound propagation in the pool.

III. Experimental Results

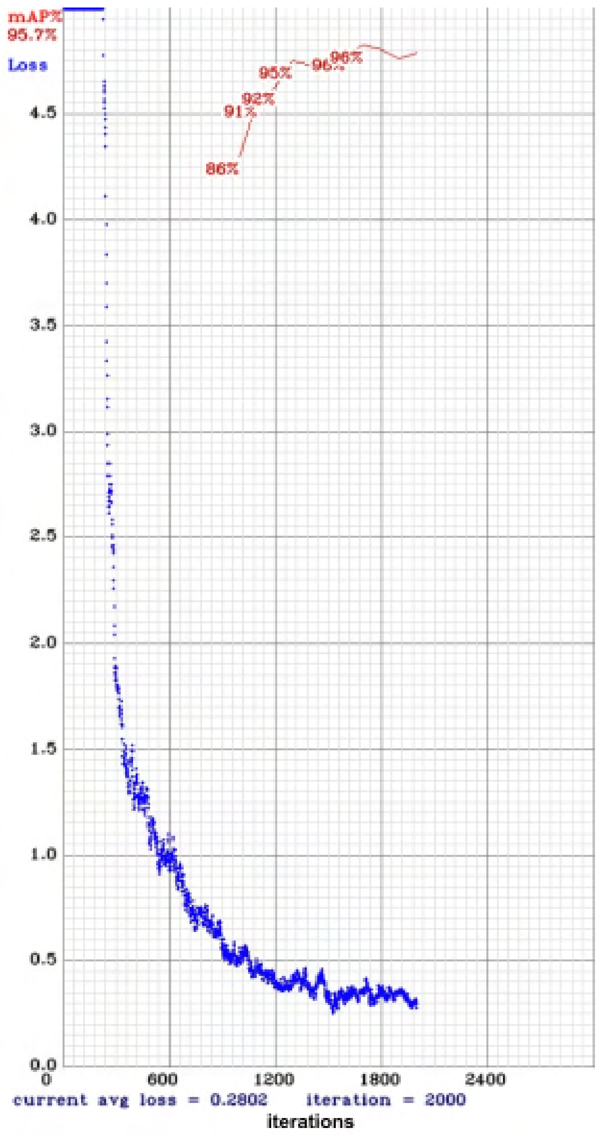
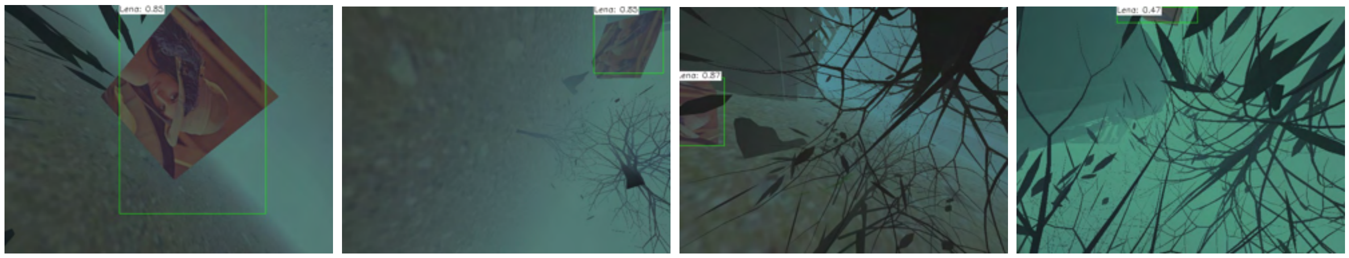


Fig. 13: Loss and mAP (mean average precision) score over training iterations



(a) Object close to camera (b) Object far from camera (c) Object far from camera with occlusion (d) Object partially in view

Fig. 14: Various configurations of objects successfully detected by trained model.

A. Computer Vision

We created scenes in Gazebo with a sea floor texture and various other objects. In each world, we placed a cube textured with a picture (we chose a cube because each render from a non-orthogonal angle provides instances of the desired image from multiple angles). With the scene prepared, we randomly sample camera positions and orientations, then render the camera's view for each pose to generate a large dataset of synthetic underwater images. Depending on the camera pose, the cube may be occluded by other objects, which increases the robustness of the model.

We generated a dataset of 800 underwater images using our pipeline, with 10% reserved as a validation dataset and the rest used for training. On our dataset, we trained a YOLOv3 model to recognize instances of the common test image Lenna.

The score we use to judge the robustness of the model is mean average precision (mAP), which accounts for the precision (proportion of detected positives that are true positives) and the recall (proportion of positives detected as positives) of the classifier, as well as intersection-over-union of the bounding boxes. Fig. 13 shows training over 2000 iterations resulted in a mAP score of 95% on our validation dataset, so we are confident about the model's performance. Once pool access is readily available, we plan on collecting in-water data for use as a validation dataset, as real-world performance will be the true test for our model.

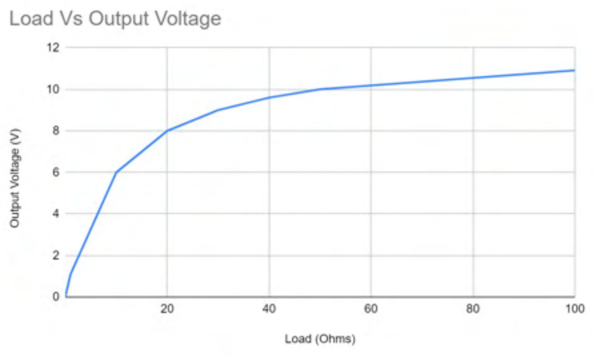


Fig. 15: Load vs Output Voltage Current Monitor Circuit

B. Current Monitoring Circuit

Due to COVID-19 safety limitations, we have yet to test this circuit physically. However, we ran some initial tests in a simulation software to ensure the basis of our design would work.

First, we tested how the circuit would behave under different loads. This was necessary as the exact load on the circuit is unknown due to in-person testing limitations.

As seen in Fig. 15, we found that the variation of the voltage output for different loads is significant. Ideally, the load shouldn't affect V_o so drastically, so this is something we are working on improving. We also tested how the output voltage changed with the current into the robot. We found that the relationship was linear as expected.

C. Pinger Localization

Although the system is not yet complete, we are using our simulator to continuously get new insight for how our system performs and what we can do to improve it. One such example is shown when testing out our multilateration algorithm. Since multilateration requires inversely solving a system of nonlinear equations, most techniques are iterative and require some form of initial guess. To start off, we decided to implement a simple nonlinear least squares (NLS) algorithm to find our position.

The NLS algorithm is inputted some initial guess for the position of the pinger, and then interates that guess to minimize the sum of the squared difference between the measured TDOA and the calculated TDOA based on estimated pinger position. While there are more complex algorithms that we are planning to explore in the future, this provided a good starting place for us to test the feasibility of our multilateration approach.

To start off, we simulated an angular sweep of the pinger at a distance of 10m relative to the AUV, but inputted a fixed initial guess at XY coordinates (-10, 0). Note that the height of the pinger is set to 5m but is assumed to be detected with full accuracy by the pressure sensor. The results of the angular sweep are shown in Fig. 16. Despite the method being simplistic, the simulation shows a worst case average angular error of 3°. With more research into more complex method and as we tighten our strategy and hydrophone geometry, we expect this number to fall bellow our goal of 1°.

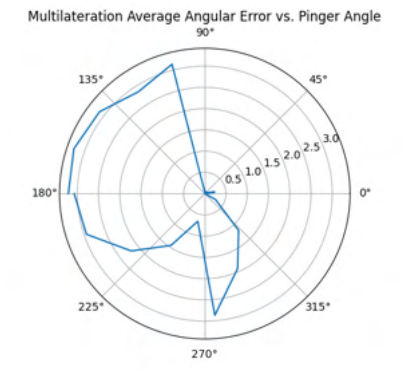


Fig. 16: Average angular error for the NLS multilateration algorithm. The pinger in this trial is located at a distance of 10m and a height of 5m with a sweeping angle in the XY plane. The pinger is initially estimated to be at an XY coordinate of (10, 0). 10 Trials per angle were deemed to be sufficient for each angle since the simulation results were relatively consistent.

Acknowledgment

UBC Subbots is supported as a student design team by the Faculty of Applied Science at the University of British Columbia. We would like to thank the UBC Applied Science Professional Development team for supporting us and allowing the team to continue functioning throughout the COVID-19 Pandemic. A special thanks to our faculty advisors: Ioan (Miti) Isbasescu, Professor Lutz Lampe, and Professor Joseph Yan. Finally, we would like to acknowledge our gold level sponsors for 2021: UBC Applied Science, Shell, and SOLIDWORKS.

References

[1] T. Ueda, K. Yamada and Y. Tanaka, “Underwater Image Synthesis from RGB-D Images and its Application to Deep Underwater Image Restoration,” 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 2115-2119, doi: 10.1109/ICIP.2019.8803195.

[2] T. I. Fossen, Handbook of marine craft hydrodynamics and motion control. Hoboken N.J.: Wiley, 2021.

[3] J. H. A. M. Vervoort, Modeling and Control of an Unmanned Underwater Vehicle. Christchurch, New Zealand: University of Canterbury, 2008.

[4] L. A. Gonzalez, DESIGN, MODELLING AND CONTROL OF AN AUTONOMOUS UNDERWATER VEHICLE. The University of Western Australia, 2004.

Appendix A: Triton AUV Component Specification

Component	Vendor	Model/Type	Spec	Cost (if new)	Status
Foam Ballast	Salvaged	closed-cell polyurethane foam	Unknown	Legacy	Installed
Stainless Steel dive Weights	Blue Robotics	SS Ballast Weight	https://bluerobotics.com/store/watertight-enclosures/ballast/ballast-200g-r2-rp/	12x$9.00	Installed
Frame	Rockey Mountain Motion Control	1”x1” aluminum extrusion + connectors	https://www.rmmc.net/8020/	Legacy	Installed
Waterproof Housing: Main	Blue Robotics	8” watertight enclosure	https://bluerobotic s.com/store/watertight-enclosures/8-series/wte8-asm-r1/	$343.00	Installed
Waterproof Housing: Battery	Blue Robotics	3” watertight enclosure	https://bluerobotics.com/store/watertight-enclosures/3-series/wte3-asm-r1/	Legacy	Installed
Waterproof Housing: Cameras	Blue Robotics	3” watertight enclosure	https://bluerobotics.com/store/watertight-enclosures/3-series/wte3-asm-r1/	Legacy	Selected
Waterproof Housing: Hydrophone	Blue Robotics	3” watertight enclosure	https://bluerobotics.com/store/watertight-enclosures/3-series/wte3-asm-r1/	$184.00	Selected
Waterproof Connectors	Digikey	EN3 series Connectors	https://www.switchcraft.com/Documents/Switchcraft_EN3_NPB_602.pdf	$500.00	Purchased
Thrusters	Blue Robotics	T200 Thruster	https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/	2x$179 + Legacy	installed
Motor Control	Blue Robotics	Basic ESC	https://bluerobotics.com/store/thrusters/speed-controllers/besc30-r3/	2x$27 + Legacy	Installed
High Level Control	Teensy	Teensy 4.0	https://www.pjrc.com/store/teensy40.html	$19.95	Purchased
Propellers	Blue Robotics	T200 Thruster Propellers		Included with thursters	Installed
Battery 1	Blue Robotics	Lithium-ion Battery (14.8V, 18Ah)	https://bluerobotics.com/store/comm-control-power/powersupplies-batteries/battery-li-4s-18ah-r3/	Legacy	Installed
Battery 2	Venom	3S Drone Pro Battery 12V	Deprecated, link unavailable	Legacy	Installed
CPU	NVIDIA	Jetson TX2	https://developer.nvidia.com/embedded/jetson-tx2	Legacy	Installed
CPU Carrier Board	Connect Tech	Orbitty Carrier for NVIDIA® Jetson™ TX2/TX2i	https://connecttech.com/ftp/pdf/ASG003.pdf	Legacy	Installed
Internal Measurement Units (IMU)	Fidget	PhidgetSpatial Precision 3/3/3 High Resolution	https://www.phidgets.com/?&prodid=32	Legacy	Installed
Camera	Blue Robotics	Low-Light HD USB Camera	https://bluerobotics.com/store/sensors-sonars-cameras/cameras/cam-usb-low-light-r1/	2x$99.99	Selected
Hydrophones	Aquarian	AS-1 Hydrophones	https://www.aquarianaudio.com/as-1-hydrophone.html	5x$395	Purchased
Depth Sensor	Blue Robotics	Bar30 High-Resolution 300m Depth/Pressure Sensor	https://bluerobotics.com/store/sensors-sonars-cameras/sensors/bar30-sensor-r1/	Legacy	Installed
Programming Language 1	C++			Free	Installed
Programming Language 2	Python			Free	Installed
Open Source Software	ROS2	Foxy Fitzroy		Free	Installed
Algorithms: Vision	In-house	Underwater Image Synthesis, Gate/Marker Detection		Free	Installed
Algorithms: Object Detection	pjreddie/AlexeyAB	Darknet	https://github.com/AlexeyAB/darknet	Free	Installed
Algorithms: Acoustics	In-house	Bandpass FIR Filter, Cross Correlation, Time Difference of Arrival, Multilateration, Beamforming		Free	Selected
Algorithms: Navigation/Control	MATLAB R2019b	PD position, controller, LQR controller, optimization, Linearization,		UBC License	Installed
Battery Management System	In-house		circuit designed from scratch from basic components	$100 (PCB + components, for both batteries)	Installed
Team Size			26		
Expertise Ratio (Hardware:Software)			19:7		
Testing Time: Simulation			30h		
Testing Time: In-water			0h (COVID restrictions)		