1. Abstract

This paper details the steps taken by Montana State University's Robocats to prepare their vehicle for the 2021 RoboSub Competition. This vehicle was designed years ago and this year they primarily focused on upgrading the capabilities of the Autonomous Underwater Vehicle (AUV), improving its structure and, developing a simulation testing environment.

2. Competition Strategy

With the addition of system complexity, the 2021 RoboCats AUV strategized to further optimize the spatial integrity of the waterproof electrical chamber by designing a new electrical rack for the vehicle. With the addition of a hydrophone system designed and tested by the electrical team, a new electrical rack system was pivotal in creating, and utilizing the limited space available in the preexisting waterproof chamber. The implementation of a new electrical rack within the sub created additional space for new components and provided a greater level of organization to the wiring components throughout the system. To ensure reliability of the submarine, a new CPU was embedded with the goal of improving performance of the operational systems and functions of the submarine. A hydrophone system will be implemented to find a pinger that is located on the bottom of the competition pool. Design considerations were then altered accordingly to ensure reliability in the sub performance.

Using the visual recognition software from MATLAB and existing data collected from GitHub, the robot was instructed to recognize the image of specific competition objects such as gates, torpedoes, buoys, and boxes. With this recognition, the robot was able to identify competition objects, and further complete competition maneuvers such as recognizing, and moving towards objects.

2.1. Pinger Location with Hydrophones

In order to locate a Pinger in the water, a combination of hydrophone system and signal processing are used. The hydrophone system consists of 4 sets of hydrophone, bandpass filter, and amplifiers. Signal processing consist of a computer and Analog Digital Converter (ADC). Once the location of the Pinger has been found, the coordinates are passed to the motor controller in order to move to that location.

2.1.1. Hydrophone System

A Pinger in the water produces a vibration with a set frequency and period. Along with other noise, these vibrations are picked up by hydrophones. Hydrophones then outputs electrical signal that represents the picked up noise and vibrations. The signal is passed to a buffer amplifier which removes any resistance that may come from the wires. The signal that comes out of the buffer amplifier is the same as the signal that comes into the buffer amp. The signal from the buffer amp is then passed to a non-inverting amplifier which boost the signal that's received through the hydrophone. This ensures that the Pinger's vibrations are received. This amplifies that noise received through this amplifier as well. The output from the non-inverting amplifier is then passed to a bandpass filter which removes the noise from the amplified signal. The filter removes any other signals that are outside the set range of frequency centered on the frequency of the Pinger. This filtered signal is then passed to another non-inverting amplifier to be scaled to a preferred signal level. The preferred signal level is where the ADC can convert the signal from Analog voltage levels into digital data. The signals are then processed on a computer.

2.1.2. Signal Processing

Four signals that came after the hydrophone system are then passed to ADC and computer. The ADC is part of a microcontroller or a chip that converts an analog voltage level to a number that represents the voltage. The data from the ADC is then transferred to a computer. The computer then timestamps the arrival of the data from each ADC. These timestamps are then used to calculate the general direction of the Pinger and guessed distance of the Pinger. The distance is guessed since the algorithm chosen can only accurately locate in a certain radius of the submarine. Once the Pinger is within that range, the submarine will know the exact location of the Pinger.

2.1.3 Location Algorithm [Need to keep adding stuff. Jacob can continue the rest if he can]

The key on locating the Pinger are the timestamps of the arrival of the Pinger signal. Using the timestamps, time difference between a chosen reference hydrophone and other three hydrophone times are calculated [Equation 1].

$$\tau_A = T_B - T_A = \frac{1}{c}\left(\sqrt{(x-x_B)^2+(y-y_B)^2+(z-z_B)^2} - \sqrt{x^2+y^2+z^2}\right)$$

$$\tau_B = T_C - T_A = \frac{1}{c}\left(\sqrt{(x-x_C)^2+(y-y_C)^2+(z-z_C)^2} - \sqrt{x^2+y^2+z^2}\right)$$

$$\tau_C = T_D - T_A = \frac{1}{c}\left(\sqrt{(x-x_D)^2+(y-y_D)^2+(z-z_D)^2} - \sqrt{x^2+y^2+z^2}\right)$$

τ_A, τ_B, τ_C = *Time difference between the time arrival on the hydrophone to a chosen hydrophone*

x, y, z = Location of the Pinger

$(x_B, y_B, z_B), (x_C, y_C, z_C), (x_D, y_D, z_D)$ = *Location of each hydrophones*

Equation 1: Time of Arrival Calculation [2][3][4]

2.2. Software in the Loop (SIL) Testing

Using a Software in the Loop (SIL) development pipeline allows for cleaner transitions between the various stages of software and hardware development. The pipeline will consist of a five-step process: local development, integration testing, revision, review, and deployment. The core of the SIL pipeline is the integration testing step which uses simulations of a 1:1 scale model of our robot to verify and examine the effects of our codebase without having to configure and submerge the bot. This will result in an improved development timetable and the resolution of issues in a timelier manner.

2.2.1 Local Development

This stage of the development pipeline will encompass code development, including the addition, removal, and refactor of features. These changes will be made locally on personal machines, utilizing Git and GitHub for version control and code sharing. Changes to the codebase will reside in their own branch, and upon complete of that feature a new merge request will be made. These merge requests will then require review and approval from at least two other team members before the feature can be merged into the production branch (see 2.2.4). This, along with frequent commits, will allow for easy tracking of our codebase as well as foster a lively and continuous review process.

2.2.2 Integration Testing

This stage of the development pipeline is focused on virtually testing our codebase in a simulated pool. This allows us to constantly verify that changes to our code work as intended and do not interfere with other features without having to upload the code to, configure, and submerge the robot in a pool of water. A great deal of time was spent deciding which simulation platform to use for this stage. The software development team considered several factors while researching simulation software, including portability, ease of integration, simulation processing time, compatibility with the current codebase and development environment, as well as the fidelity of the simulation itself. Some Frameworks considered included Matlab Simulink, an inhouse solution, Webots, Gazebo, and many more. After much consideration and discussion, Matlab Simulink and Gazebo appeared to be best suited for our application. Simulink offers many desired features including intuitive and well documented control design process with which most members were already familiar with. Gazebo offers a means of verifying the physical behavior of the robot in a simulated environment. Combined, these two solutions allow for a more accurate and seamless development process – Simulink offers an intuitive way to verify logic and expected behavior on the software level while Gazebo offers a means of verifying the physical actions of the robot. As such, we decided to implement both solutions through the method of co-simulation using the Gazebo plugin for Simulink[1] which allows both solutions to interact with each other during the simulation.

1 https://www.mathworks.com/help/robotics/ug/perform-co-simulation-between-simulink-and-gazebo.html#responsive_offcanvas

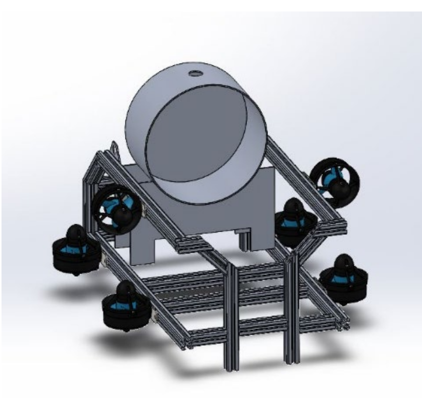

Fig 1. A work in progress model of the robot used for simulation in the SIL pipeline

2.2.3 Review

The review stage will be an important feature or our development workflow, helping identify and resolve feature issues before code is pushed to production. This stage will occur after a feature has been pushed to a merge request and has completed preliminary integration tests [1], and the stage will consist of at least two other team members pulling and reviewing the code [2]. The review process will involve looking at the feature's accuracy in achieving its intended functionality as well as the quality and efficiency of the written code. Comments and suggestions for improvements and changes will be exchanged on the merge request in GitHub and can be elaborated upon in person at developer or club meetings.

[1] In the case of major changes to the feature during revision, multiple integration tests will be necessary to ensure the feature maintains its function throughout the review and revision process (see 2.2.4).

[2] It is encouraged that more than two team members will be allowed to review merge requestions.

2.2.4 Revision and Deployment

After receiving feedback, the feature will then receive enter the revision stage. This stage will be an opportunity to respond to feedback and suggestions offered in the review stage (see 2.2.3) After revisions have been completed, the new commits will be pushed back to the merge request where the entire feature will be reviewed once more. More revision will occur if new issues are found, or new suggestions are made. Of all comments and suggestions are resolved, the merge request will require approval from at least two team members. Upon approval, the code will be deployed to the appropriate on-board device on the robot (see 4.3 for details on upload and deployment processes).

3. Design Creativity

Implementing a new electrical rack was essential for regulating and optimizing the heating, cooling, and overall space capacity of the watertight capsule on the submarine. The switch from a triangular to rectangular frame was a creative decision to maximize simplicity and organization of the electrical components. The first model, as seen in Figure 1, was constructed from 1’’x 1’’ aluminum 80/20 beams, to allow for easy modifications and design alterations. Later, with the implementation of the Shuttle CPU, the frame was altered to accommodate for the greater space of the CPU. Figure 2 displays the most current addition of the electrical rack design.

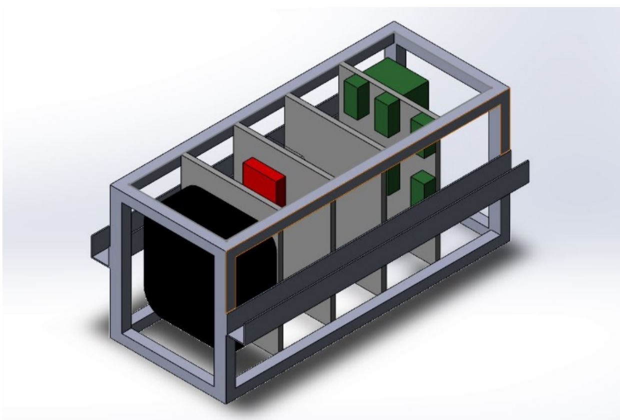

Figure 1: First electrical rack design

Figure 2: Final electrical rack design

As seen above, the electrical rack accommodates the size of the newly implemented Shuttle CPU and has additional space to install the hydrophone system for the 2022 competition. This new model also supports the heating and cooling considerations that the CPU generates, as the CPU is positioned closed to the water flow around the watertight chamber, and away from converging wires and cables.

3.1. Hydrophone System and Algorithm Design

For the Hydrophone System, two versions were implemented. The difference between the two versions is the bandpass filter. The bandpass filter for the first version is a 2^{nd} order active low pass filter in series with a 2^{nd} order active high pass filter. The bandpass filter for the second version is a programmable bandpass filter.

3.1.1. First Version of Hydrophone System

A 2^{nd} Sallen-Key order bandpass filter [Figure 3] was designed using an online calculator to produce the required component values [5]. Each stages of the filter were then analyzed to see what the expected results is from the using the filter. The first version lacked a stepper drop-off, so some noise could be present after filtering. This led to the second version.

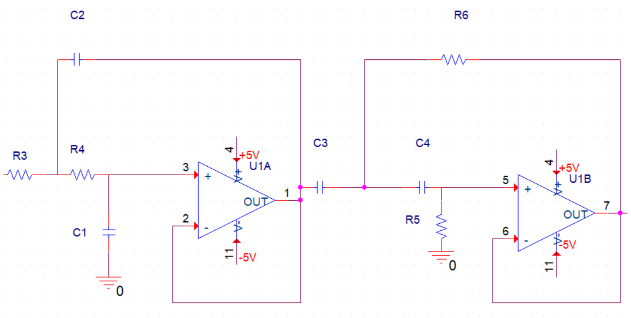

Figure 3: 2nd Order Sallen-Key Bandpass Filter

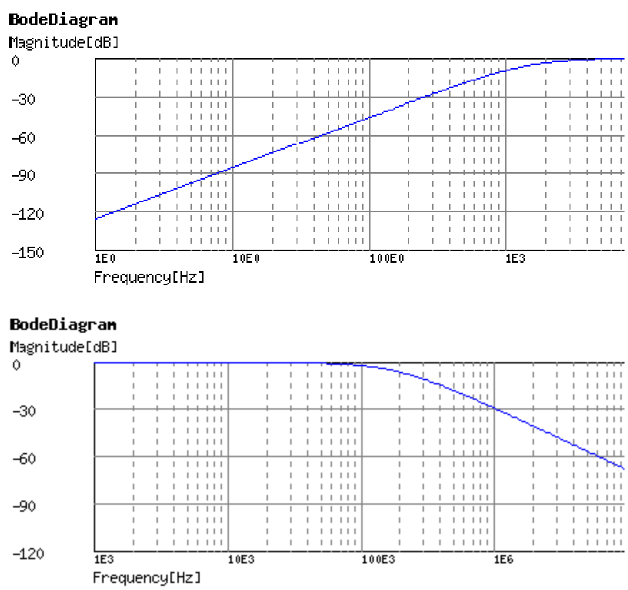

Figure 4: Analysis of Bandpass Filter

3.1.2. Second Version of Hydrophone System

A MAX267 Pin Programmable Bandpass filter was used for the second version. The bandpass filter was configured according to the datasheet of MAX267 [4] [7]. The configuration was able to filter 25kHz, 30kHz, 35kHz, and 40kHz. This version wasn’t fully implemented due to missing clock. An alternative of using a crystal oscillator to produce a frequency required but was not sufficient.

4. Experimental Results

Due to ongoing restrictions imposed by the COVID-19 pandemic our team has only recently gained extremely limited access to a public pool for testing purposes. We also are continuing to face hardware and software complications that have resulted in an inability to test in-water such as the failure of our in-water tether, insufficient solder joints for various communication wires, inability to communicate with the sub via SSH and, nonfunctioning legacy scripts. As of late June, we have rectified all but one of these issues and hope to get our sub back in the water in the coming weeks.

4.1. Hydrophone System Experiment Results

Two versions of the Hydrophone System were tested both in lab and in pool. Two conditions can make a difference to how large the amplitude the signal will get the more distance between the sensor and Pinger.

4.1.1. Hydrophone System First Version

The first version of the hydrophone system was tested on lab. This version successfully filtered 25kHz and 30kHz. However, the 35kHz and 40kHz were distorted. This is due to the circuit being designed to only filter all 4 frequencies, but the drop-off was around 40kHz.

The same version was then tested again in a pool. The filter did not pick up the signal from the Pinger, but the non-inverting amplifier at the second stage was adjusted to pick up signal from 30 ft away. 25kHz signal was filtered and amplified to a chosen amplitude. This version of hydrophone was not further developed due to COVID-19 pandemic and the drop-off result was not steep enough that some noise could be picked up.

4.1.2. Hydrophone System Second Version

While on lab, the second version was tested. It can filter 25kHz, but the crystal oscillator created noise in the power supply which is not desired since it creates a feedback to the filter and adds to the output signal. The circuit was not further developed due to COVID-19 pandemic. For the same reason, the second version was not further tested in pool.

4.2 Motor and Maneuver Testing

During our first in-water test since before the COVID-19 lockdown we cane to two conclusions. Primarily the legacy python and Arduino scripts necessary to control our sub were both outdated and virtually non-functional with our current configuration. Secondly, two or more of our thrusters were not receiving power. Luckily the latter was only caused by the afore mentioned deteriorating solder joints. This resulted in a very short test which did not allow us to diagnose any further issues or fully test our maneuvers.

4.3 Software Testing

Since our first in-water test our legacy control scripts have been refactored to receive command line commands via a python script running on our Intel NUC running a Linux distribution and send them via serial communication to an Arduino Mega using an acknowledgement (ACK) based protocol. This allows for much more control than our previous testing scripts which did not provide meaningful feedback while running, could not easily be halted in software and, did not allow the user to directly call individual maneuvers.

In the last year, the software team has devoted most of its time and resources to developing a simulation environment for out sub. As mentioned in Competition Strategy we are implementing a form of SIL testing that should allow us to accurately simulate our sub without the need for a pool. Currently, we have not fully functional, but we plan to have it ready to use in the coming months.

5. Acknowledgements

Montana State University's Robocats would like to thank all the people and organizations who were involved in the continued development of this project. Our university and, the Norm Asbjornson College of Engineering for their support, our club advisor Dr. Bradley Whitaker and our business partners NAVSEA and BlueRobotics.

6. References

[1] Montana State University's Dr. Sophie

[2] Rudolf Rabenstein, Paolo Annibale, "Acoustic Source Localization under Variable Speed of Sound Conditions", Wireless Communications and Mobile Computing, vol. 2017, Article ID 9524943, 17 pages, 2017. https://doi.org/10.1155/2017/9524943

[3] https://robosub.eecs.wsu.edu/wiki/cs/hydrophones/multilateration/start

[4] https://www.ece.ucf.edu/seniordesign/fa2009sp2010/g19/hydro.pdf

[5] http://sim.okawa-denshi.jp/en/Fkeisan.htm

[6] https://blog.bliley.com/filter-typology-face-off-a-closer-look-at-the-top-4-filter-types

[7]
https://datasheets.maximintegrated.com/en/ds/M
AX263-MAX268.pdf

Appendix A: Component Specifications

Component	Vendor	Model/Type	Specs	Cost (If New)	Status
Buoyancy Control					
Frame	McMasterCarr (Student Designed)	extruded T-slot	6061 Aluminum, 1"x1"		
Waterproof Housing	Student Designed				
Waterproof Connectors	Seacon	All-Wet Underwater Electrical Wet-Mate Connectors	8pin or 10pin, water tight		
Thrusters	Blue Robotics	T200	390 watts, waterproof,		
Motor Control	Blue Robotics	Basic 30A ESC	30A,		
High Level Control	Arduino Mega				
Actuators					
Propellers	Blue Robotics	Included with Thrusters			
Battery	Blue Robotics	Lithium-ion Battery	14.8V, 18Ah		
Converter	Amazon	Yeeco DC DC Buck Voltage Regulator	Input voltage:DC7-36V; Output voltage: DC1.25-32V		
Regulator					
CPU	Amazon	Intel NUC7i7BNHX1	i7 7th Gen, 16GB RAM		
Internal Comm Network			I2C, Serial		
External Comm Network			Ethernet (SSH)		
Compass	Included in AHRS				
Inertial Measurement Unit (IMU)	Sparton	AHRS-8	9 axis		
Doppler Velocity Log (DVL)					
Vision	Microsoft	Lifecam Cinema: H5D-00013	720p, 30 fps, 5 MP		
Acoustics					
Manipulator					
Algorithms: vision	Student Designed				
Algorithms: acoustics					
Algorithms: localization and mapping					
Algorithms: autonomy	Student Designed				
Open source software	Arduino	Arduino			
Team Size (number of people)	15				
Expertise ratio (harware vs. software)	1				
Testing time: simulation	0				In Development
Testing time: in-water	1 hr				
Inter-vehical communication					
Programming Language(s)		C++, Python 3			