Autonomous Unmanned Vehicle Systems & Intelligent Robotics (AUVSIR)

T. Giles, C. Robles, M. Cobian, C. Ramirez, C. Nagel, X. Du, S. Harrison
Oregon Institute of Technology (Oregon Tech), 3201 Campus Dr, Klamath Falls, OR 97601

I. Abstract

The goals of OTUS AUVSIR Team at Oregon Institute of Technology (Oregon Tech) are to obtain learning objectives and experience hands-on knowledge while strengthening theoretical aspects together at the 2021 RoboSub online competition, even in restricted situations. The purpose of our project is to design and manufacture a functional autonomous robotic submarine capable of completing a couple of tasks underwater including passing through a mandatory gate. We've done it through 3D models using Blender like virtual reality this year as shown in Sec. 4. We have been fortunate to work at home remotely even in the spring term, July and also came to school labs in the early August so that we were able to build the vehicle through utilizing OIT machine shops and robotics lab while testing in a limited capacity, though, for essential propulsion test under autonomous mode (also see Section 4). We have designed four candidates for the 2021 RoboSub competitions. With thorough design matrix and tough discussion, we selected the final version of hull and frame for the Competition that enables our strengths to focus on navigation to complete as many tasks as possible such as a torpedo to shoot either Bootlegger or G-man after opening via a robotic arm, which is designed for up/down motion to also lift barrel cover and to drop markers or drop bottle into 2 bins while using a new magnetic dropper. Thanks to the participation of computer/embedded/software teams, we were able to develop upon last year's system with a

hardware package consisting of a new Raspberry Pi 4 (SBC) with Raspbian OS, Arduino Mega, and a new 9dof PhidgetSpatial IMU device to replace the Pixhawk. Python and C++ are our main languages, Python being for the main application, and C++ is used both for compiling in the visual studio to incorporate image and vision system and for our GUI that generates the state machine for mission control to develop competition strategies, depending on situation at the competition, and also certain C++/Java libaries to communicate with the IMU and pressure sensor systems. Additionally, we have built a virtual simulation with 3D models to make scenarios and it is going to further develop all tasks for the next year. We look forward to competing with our robot at RoboSub next year on site, TRANSDEC pool.

II. Competition Strategy

2.1 Software State Machine

The software side of the project had vast improvements due to the many contributors to it. Before the RoboSub can start, there is a 'Mission Control' where specified tasks and goals are determined and set to be sequentially accomplished by the RoboSub during its mission. Mission Control encompasses any programs that need to be done on the onboard computer before the vehicle would descend into the water. Any instructions inputted there would then be translated into SQLite then exported to a text file. Once the sub is turned on underwater, the Mission Execution would start when the vehicle is turned on externally and take the text file from Mission Control. Based on the instructions from said file, the code would extract information from our vision recognition system, which calculates any important variables from the camera via OpenCV, such as the distance and vector between the RoboSub and its target. Based on the results of the information, the code will tell the Arduino to make the motors run in intuitively calculated sequences to reach the assigned objective(s). This process is looped until all assigned tasks are complete.

2.2 Mission Control

Most of the Mission Control stage takes place in a graphical user interface (GUI) shown in Figure 1, and like the on-sub software, the GUI is developed in a similar way to promote adaptability and future additions. The GUI has a select then drag and drop interface, simplifying the instructional process for the vehicle. Most of these specific instructions are simple commands such as moving forward and rotating right, but more of the complex ones rely on camera and Arduino during and after mission execution, such as:

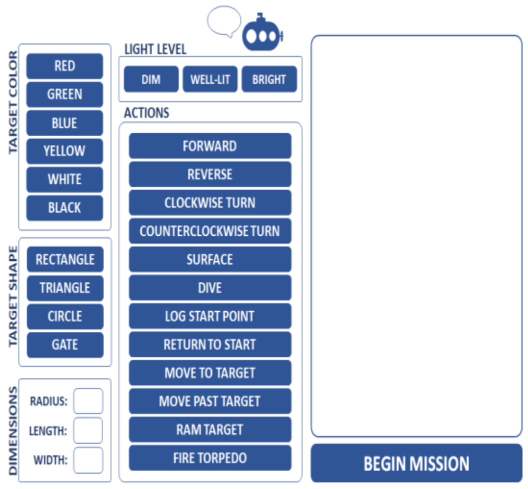


Fig. 1: Mission Control GUI for-AUVSIR

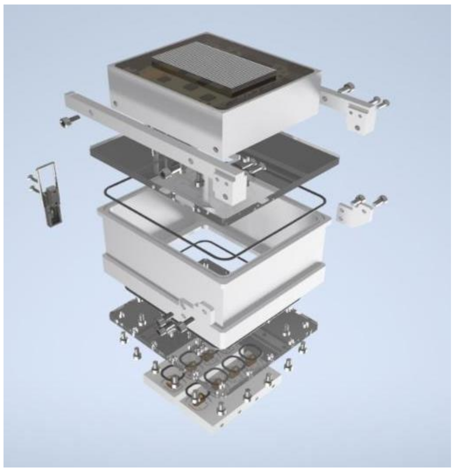
'Log Start Point', 'Return to Start', 'Move to Target', 'Fire Torpedo at Target'.The first two rely on the initial coordinate position data from the IMU, so after it finishes later mission tasks, it could use this data to automatically return to this logged point. The last two commands rely on the vision processing info as well as IMU orientation for calculating the vector necessary to fire a torpedo accurately at the target.

III. Design Creativity

3.1 Main Body

Electronics Housing

Found at the heart of AUVSIR is a central aluminum waterproof containment system designed to house and protect key electronic components. The body and lid are machined from aluminum stock, then supplemented with epoxy, polycarbonate, and rubber gaskets to create a functional, easily accessible housing.



Fig, 2. electronics housing

Now machined from billet aluminum, it serves as a mounting platform for electrical components such as relays, multiple flight controllers and power distribution boards. Previous designs required welding to construct the housing. Creating the housing from asingle billet has allowed us to reduce weight while increasing pressure and depth capabilities. The entire housing system consists of three main components. The lid, the potted ESC's, and the Polycarbonate sheets. The lid construction is dual purpose component machined from billet aluminum bolted to a polycarbonate base, and by using polycarbonate sheets, we can perform visual inspections of the gasket surface. The Electronic Speed Controllers for thruster control are located under the housing, and this year, the ESC's are smaller and lighter in weight.

Stress Analysis:

Aside from competition requirements, we wanted to see the deformation of the main hull frame. The pressure won't be serious since the depth at the competition venue is not deep, though, FEA is a de-facto technology for detailed stress of the hull frame, which has all computing and sensing technologies. Using Creo, we accomplished finite element analysis at 17 psi and get a good prediction of the deformation across the frame shown in the below figure.

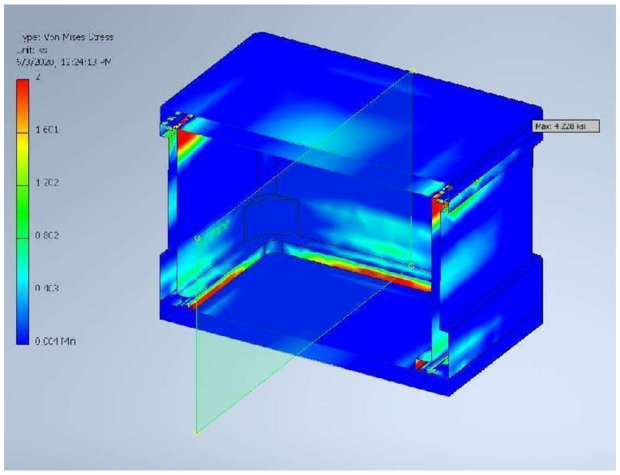
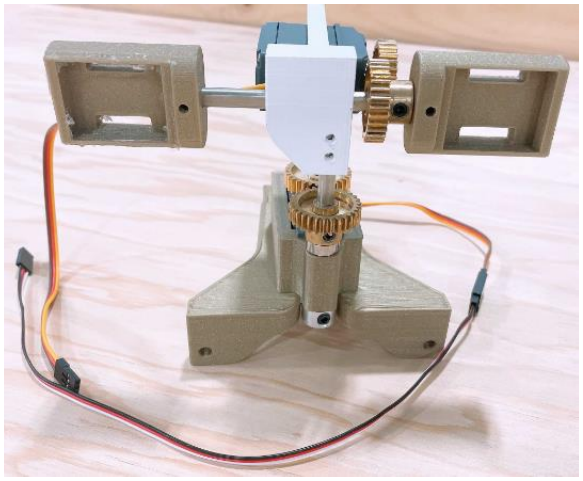


Fig. 3: Section view of stress analysis of electronics box.

Stereo Vision Emplacement:



Fig, 4. Cameras housing

The design for this year calls for two cameras that have pan-tilt motion with gear boxes, each of which has 180 degrees freedom of rotation, allowing the sub to view objects with better accurate distance using disparity image maps between left/right images and works for the primary direction of navigation. It can fully rotate to the downward direction to find landmarks or objects so that robot can adjust depth before moving to the next mission as well as for manipulating objects by manipulating the gripper arms without changing of whole robot attitude. Using 3D printed housings has allowed a reduction in weight due to the ability to print complex geometries that are not machinable by conventional methods and the use of lighter materials. The new camera boards also offer nearly 180 degrees of views for left-right/up-down compared to the static 60-degree view of the previously used web cameras. The software for image processing, computer vision, and tracking has C++/OpenCV/Tensorflow to detect, identify, classify and extract ROI (region of interest) to navigate and execute the commands (see Sec. 3.2 for details).

Thruster Mounts:

The thruster mount design for the RoboSub had to incorporate four horizontal thrusters and four 45º thrusters. The design approach originated from last year’s design which indicated mounting the thrusters on to the 80/20. Over the fall term and Christmas break we began brainstorming and designing. After coming to some designs that we liked, we began the prototype phase. We went through a few iterations of prototypes because of fitment issues. Some of the first prototypes were not sliding on the rail. After adjusting the dimensions, we came to something we were happy with.

The final prototypes of the two mounts worked well and attach all the thrusters firmly. The first complete revision consisted of 8 separate mounts. While it did its job of holding the thrusters, it turned out to be a challenge removing them. This led us going back to the drawing board. We combined the two different mounts into one solid mount. This reduced the overall mounts from eight to four. The new design also let us widen the sub which was needed for stability and buoyancy. This revision was very sturdy and made it easier to remove them if needed. However, after the water testing, we found that the movement and stability were not up to par. This set us back once again.

The final revision, shown below in Figure 5, had several major differences. The thruster was moved to the inside of the frame to increase stability. Where the thruster was located on the previous rendition placed more of a rotational movement on the frame instead of the desired vertical force. In addition to changing the location of the mount, all of the sharp corners were removed. This greatly decreases the drag coefficient of the submarine as it moves through the water. This final corner mount makes the sub wider than any previous mount. As subsystems started to be installed on the submarine, we needed more space to attach these systems. These corner mounts doubled as attachment points for the arms and even cleaned up some of the wire management issues. These four corner mounts were made on a 3D printer and were made using polyethylene terephthalate glycol, or PETG. This design is sturdy enough to meet the requirement of the submarine.

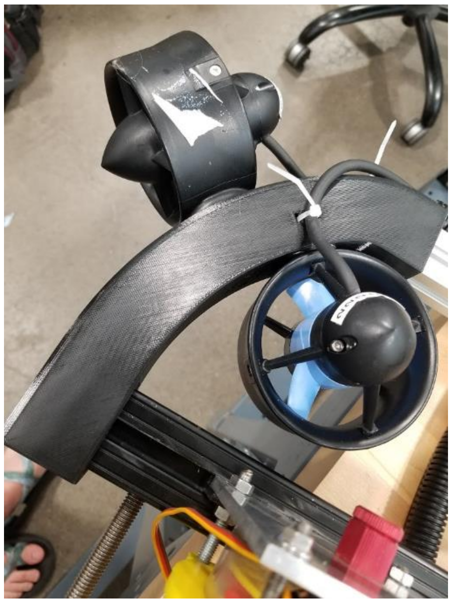


Figure 5: Final corner design with attached thrusters

Torpedoes:

The self-propelled torpedo contains a lot of electronic in a small size form. There are two propellers, which counter rotating to each other, attached to two brushless motors at each end of the torpedo to power the torpedo’s traveling and make sure it travels straight to the target. The water-cooled electronic speed controllers are soldered to brushless motors inside of the torpedo to satisfy the speed limitation inquiry. The light sensor inside could be activated by the simple flashing light to launch to torpedo through the nano Arduino board. A battery is attached inside to power everything. The figure below is the torpedo containing all electronics required to drive it.

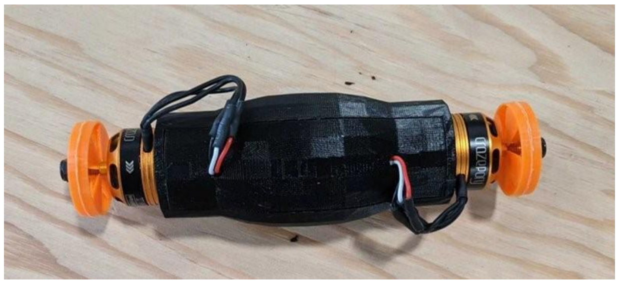


Fig. 6: 3D printed assembled torpedo

Marker droppers:

We have elected to use two permanent magnets capable of holding 1.5” x 1” x 0.5” rectangular steel objects as markers. Each system is composed of an inner and outer cylinder where the inner cylinder can rotate from 0-90° via the help of an underwater servo. Each cylinder has a total of four ¼” x ¼” spherical neodymium permanent magnets placed radially and normal to the center of the inner cylinder. Another four steel rods are placed radial like the magnets, but these are placed parallel to the system. With a larger steel rod running through both cylinders, it creates a large magnetic field when the inner cylinder is in the appropriate position and will nearly stop the magnetic field when the poles are reversed.

Using this method, the marker is dropped by using the cameras on-board to signal when the submarine is in position. Once the submarine drops, it will then retract back to the on position to retrieve any markers by getting close to it.

The system is 3D printed out of PLA for testing purposes but will be machined out of 6061 aluminum for its lightweight but strong properties. Each system is located on each battery pack.

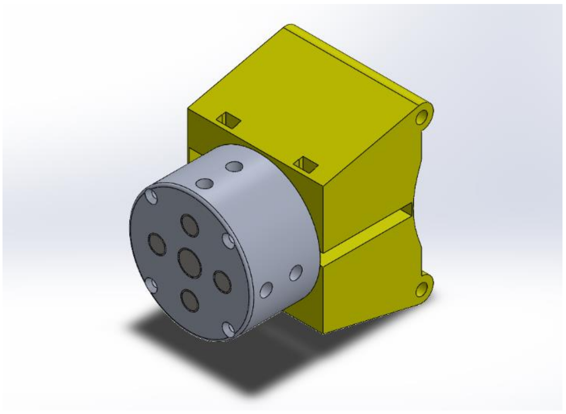


Fig. 7: Assembled view of marker dropper

Arms:

The arms portion of the RoboSub, for use in a lever manipulation challenge, are installed on the foremost motor mounts of the autonomous vehicle. The arms will function similar to the human arm: a set of two double-jointed arms with gripping implements at their end, all driven by servomotors. The fore and upper arms approximately the same length of 10 inches and move laterally only in the XY-axis of movement. This is considered optimal, given the ease of the sub in its entirety to rotate and orient the arms for more finite movement using its primary propulsion system so that the Z-plane location of the target will be lined up. The total range of motion will be in a 1.5-foot half circle radius in a vertical plane, centered on the forward 80/20 mounts.

The bulk of the assembly consists of 3D printed parts (i.e. gripper, forearm and upper arm) The inside gripping surface of the vice is texture to ease in target capture and reduce the need for accuracy when deploying it to pick up an object. The individual arm and jaws are manipulated using a system powered by three stepper motors per arm at the shoulder, elbow and gripping joints.

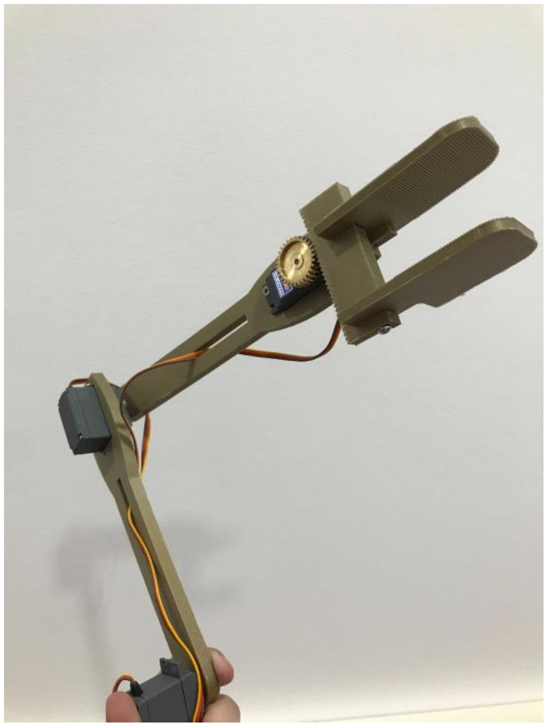


Fig. 8: Gripper assembly.

Each arm extends from the forward motor mounts of the sub providing a stable mount for the 2 assemblies, which have a 1.4-foot arc of movement in 180 degrees. Moving primarily in a vertical plane perpendicular to the front of the Robosub, the sub will be positioned so that the target is within this plane, reducing the targeting for the arms from 3D to 2D. This data will be fed from the cameras in the form of 2D grid coordinates within the plane of movement for the arm assembly to capture.

3.2 Embedded System Design

The purpose of this system is to control an autonomous robotic submarine that will be submitted in the RoboSub competition, but also to create a variable autonomous robotics library for students at the Oregon Institute of Technology for development on future subs and club projects. This autonomous submarine must be able to perform various advanced tasks requiring streamlined communication between both the hardware emplacements and software objects without human interference.

Hardware:

The main electronics housing contains a primary Raspberry Pi 4B+ for main control and computation USB connected to a PhidgetSpatial 9dof IMU for independent, no-interference vector and positioning data, with an Arduino Mega for controlling 8x T200 BlueRobotics thrusters, arm servos, as well as extra sensor hookups if necessary. Every peripheral sensor or board is integrated into the system so that it could be removed, and the rest of the system could still run at varying strengths. This was implemented in order to help us in certain situations, such as if we did not have the board at the time or wanted to debug consolidated parts of the navigation system. This is a hardware-software symbiotic relationship system that can be tested and developed quicker and easier on other future robotic platforms. Although the embedded system is not custom integrated into one another through a PCB or all-in-one board, the communication between our multiple boards/sensors is very steady, with no problems having arisen from the directly connected serial communication lines. The Raspberry Pi is able to efficiently decode the data sent out and in of itself, whether it be to and from the Arduino Mega, or from the Phidget22 IMU over USB. When parsing data from a gyro connected to the Arduino Mega, there is no noticeable data loss when passing it onto the Raspberry Pi.

Software:

The basic navigation system is built around allowing our RoboSub to travel to any orientation matrix or position using coordinates and waypoints. The central class for this is referred to as our Movement Commander, containing every subsystem, most importantly including our gyro classes, thruster hardpoint controllers and vision recognition scripts. We do not use many external libraries, as we strive for as much customization and full understanding of our systems as possible. All of the navigation code is our own, and we only use some Python libraries for certain sensors, peripheral communication, and vision recognition. The code is also written in such a way to ease testing and development because of our limited software team. As mentioned before, the code is able to intuitively check if certain sensors are disconnected, but we can also go in and manually change the initialization values of the application to only run specified subsystems. This has been extremely useful for debugging certain parts of the RoboSub when introducing new, possibly application-breaking hardware or software implementations, as we can consolidate errors into ‘bite-sized-pieces' of our code instead of trying to sift through our entire application to find bugs. When this practice is fully employed, we are able to easily delegate development while also lessening the time it takes to debug.

The RoboSub application is completely autonomous within a state-like system, where it reads commands generated from GUI that enables anyone, experienced programmer or not, to navigate an intuitive GUI to generate a list of commands or “Mission” for our RoboSub to process and turn into iterative logical navigation. The sub starts every mission through either pressing a start button on the top of the sub, or manually running a specific script in the SSH console of the Raspberry Pi host computer.

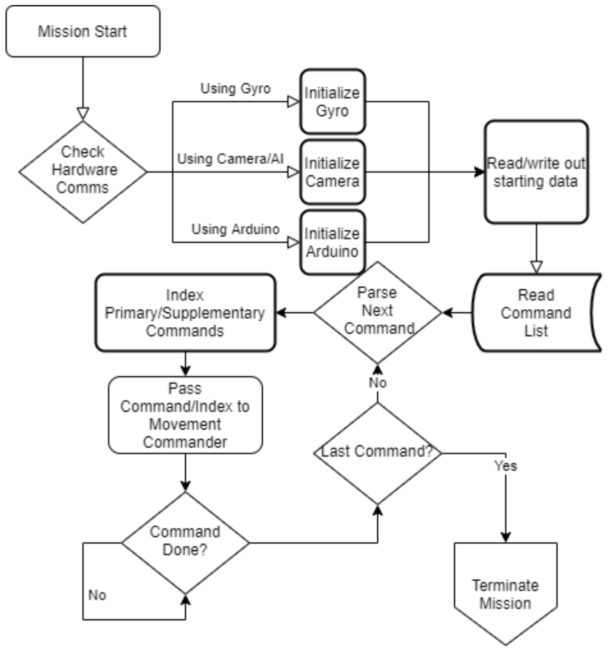


Figure 9: Simple Software Flowchart

For each state, or command read in this system, the sub is able to analyze the command and run specific functions and algorithms depending on the command as well as it’s supplemental data. Almost every command has supplemental data, such as the describing name of a visual target to locate, or gyroscopic/positional data to navigate towards. Each command is categorized in order to limit parsing time and code complexity, so more commonly used command categories have priority for parsing through and comparing to the input command.

IV. Experimental Results

Simulation:

Since we had no access to a physical water environment for testing the Robosub last summer, we worked on possible alternatives that could continue to be useful in the years to come. We were able to test certain parts of the software application through tests in simulated environments, but there is also a simple framework for generating a coordinate map of the RoboSub's physical environment in real time during in-water testing and active missions. We also have environments developed in Unity to accurately recreate what in-water vision would look like that are then used to train the vision AI and OpenCV scripts, and the stored coordinate maps can be used to improve positioning data in future missions. The rendered environment has proven useful for vision testing so far, but the coordinate mapping is only conceptual so far.

Waterproofing:

As far as testing the Robosub's waterproofing, we were able to test its three main components individually and again all together. These components being the central hub and the two battery packs. The central hub is machined/milled down from 6061 aluminum with no welding involved therefore decreasing the chance of failure due to poor welds. On the bottom and top we have added plexiglass to allow visual inspection of the hub. The bottom being fixed, we added the electronic speed controllers (8) on the outside of the hub for natural water cooling and decreased heat dissipation on the inside hub. To waterproof the controllers, an added base of epoxy was added to the inside of each individual controller and added a rubber gasket which was bolted down to the main hub. Two adjustable latches were added to the lid for easy accessibility into the hub. There have been holes drilled into the hub for power and signal, these also have been lathered in marine epoxy.

After successful visual examination, the battery cases and central hub were tested in the water trough. After continuously checking for visual leaks, the submarine was left in the trough for two days. It passed the test, showing no signs of leakage into the hub or the battery case.

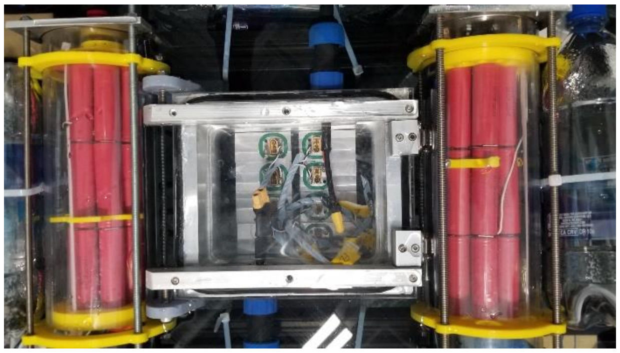


Figure 10: Waterproofed central hub and batteries.

V. Acknowledgements

Thank NASA Oregon Consortium for sponsoring the project partially by NASA Oregon Space Grant (80NSSC20M0035). Special thanks to Oregon Tech MMET Sr. as well as the Oregon Tech AUVSIR Club members.

Student Players at Oregon Tech (OIT)
- César Robles: Camera System & Mech Team Lead
- Theodor Giles: Embedded & Software Lead
- Moisés Cobian: Marker Drop Lead
- Caleb Nagel: Robotic Arms Lead

- Sean Harrison: Mounts & Waterproofing Lead
- Xuanhe Du: Torpedo Lead
- Christopher Ramirez: Mounts Design & Support
- Mauricio Huntoon DeRoche:

Camera/Sensor/Torpedo Support Member

- Jake Mitchell: Club Member
- JayCe Leonard: Club Member
- Austin Morris: Club Member
- Dongbin “Don” Lee: Faculty advisor and mentor

We would like to thank OIT for partially sponsoring the project through RBC, NASA, Oregon Space Grant Consortium Cooperative Agreement 80NSSC20M0035 for their support and encouragement. Our advisor Don Lee has been an excellent source of information and support throughout this process.

IV. References

Oregon Tech RoboSub Team, OTUS AUVSIR,

https://sites.google.com/view/otusrobosub/home

Appendix A: component specifications

Component	Vendor	Model/Type	Specs	Cost-New
Buoyancy Control	OIT	Motor Control	Programmed	
Frame	OIT	AL6160, ABS	Machine/3D print	
Waterproof Housing	O-ring, design	Polycarbonate tube, Resin, Wax	A bulk	
Waterproof Connectors	Blue Robotics/Amazon	XT30/60 (IP-68)	3-/9-pin connectors	
Thrusters	Blue Robotics	T200		
Motor Control	Speed Control	Basic ESC	30A	$27
High Level Control	Outer-loop PID control	Students' Programming		
Actuators	Hi-Tech	Servo/stepper brushless	Electric motors	
Motor Driver	H-bridge	Stepper motors		
Propellers	Blue Robotics	Fwd/Bwd set	Left/Right blades	

Battery	Panasonic Lipo	36EA (18,650mA)	2 * 18/stack	$15
Regulator	Built-in			
CPU	Raspberry Pi 4, 1.5GHz	ARM Cortex-A72	Q-core 64bit/4Gb	$56
Int comm network	Ethernet	Gigabit		
Ext comm interface	Wired for testing	RJ-45 wired	Fiber optic wires	
Program Lang. 1	Python/Pytorch lib			
Program Lang. 2	C++			
Compass	Pixhawk	magnetometer		
IMU	Pixhawk	Accel'/Gyros		
DVL	NA			
Camera	Logitech	C270		
Depth Sensor	Pressure			

Hydrophones	Teledyne Reson	T4019		
Pinger	JW Fisher	Mid-Freq range	20-50KHz/2k-3k feet	
Manipulators	OIT	Custom-made	Front & Bottom Foldable arms	
Algorithm: vision	VIGRA lib/OpenCV			
Acoustics	NA			
Local' & mapping	3D image building			
Autonomy	OIT			
Open Source SW	Python,C++,OpenCV,	Raspbian,Pytorch	Blender (3D)	
Team Size	Approx. 10			
HW/SW ratio	1:1 (5:5)			
simulation time	3 Months			

Appendix B: outreach activities

Theodor: Welcoming newcomers at club fair and introducing them to the Discord. 1/15/2021
César & Moises: Hanging posters around campus and introducing new members to the Slack social site. 2/7/2021