Amador Valley Robotics Club: Design of Nemo AUV 2021

Suhas Nagar (President); Mechanical: Sri Parasaram (VP), Andrew Delevaux, Ishan Duriseti, Aayush Gupta, Dylan Kwong, Isabelle Lo, Ifran Mohamed, Rohan Panjikkaran; Electrical: Edward Ding (VP), Jessie Chan, Eaton Huang, David Li, Justin Yu; Software: Phoebe Tang (VP), Angad Bhargav, Michael Li, Kush Nayak, Kalyan Sriram, Nikhil Sunkad, Craig Wang, Vincent Wang, Andrew Xiao, Serena Zhou; Business: Athan Yang (VP), Ethan Apalis, Myra Qin, Leo Shao, and Xina Wang

***Abstract*—This year, AVBotz's new submarine, Nemo, was finally designed and assembled despite the challenges that COVID-19 placed on the team. With a design that focuses on stability and modularity, Nemo is now able to accomplish tasks more efficiently than before without the extra weight of the previous submarine, Marlin. During the transition to a new submarine, the electrical division took the opportunity to overhaul the electrical system by organizing the components to make maintenance easier. Furthermore, since social distancing guidelines prevented any effective pool tests from taking place, AVBotz created a fully functional simulator in Gazebo, which allowed the software division to test code and vision functions without needing a physical environment, improving the accuracy of the AUV's navigation software. All of these upgrades have allowed Nemo to become the most optimized AUV that AVBotz has created.**

I. Competition Strategy

The themes we set for ourselves during this turbulent year were simplification and revitalization, and we focused on making Nemo as efficient as possible. Last competition, our team decided to avoid torpedoes and instead focus on the core tasks of simplifying our approach. Our modified targets were:

1) Gate with style points
2) Bin drop
3) Hitting buoys
4) Surfacing in the octagon

To complete the tasks, computer vision is used to detect the locations of the gate, bins, and buoys as we navigate near the objects. Next, our hydrophones algorithm is used to calculate the angle to the octagon, allowing the sub to surface within it. Additionally, we decided to attempt the bins before the buoys, as touching the backside of the buoys would yield more points.

In the latest competition, our accuracy was low, especially for the bins task, and we were unable to reach our goal of entering the finals. Upon analysis, we attributed the low score to a flawed hull, convoluted electrical wiring, and inefficient navigation code. These factors have contributed to the team's drought of competition success, causing us to focus on overhauling old architectures while maintaining the same targets to simplify competition runs.

Our foremost priority this year was to construct our new AUV, Nemo, to fix the shortcomings of our previous sub, Marlin. In past competitions, Marlin consistently displayed structural issues that led to unexpected behaviors. A major problem was the asymmetrical hull, which caused the AUV to veer off course at high speeds, reducing our accuracy for tasks such as gate and buoys. Thus, we aimed to redesign the hull this year, focusing on increasing the stability of our AUV. Furthermore, we designed Nemo to be modularized with standardized panel slots, allowing members to replace any panel and optimize the vehicle's ease of maintenance.

Due to the restrictions set by COVID, our team had to come up with a unique solution to testing our mission code and submarine. This year, our software division directed their resources into creating a simulator to test our mission code instead of an in-person pool test. This data-focused approach of testing allowed us to debug code at a dramatically increased rate, adjust the low-level motor control process for smoother movement, and develop more accurate navigational approaches for the gate, bins, and buoys. In previous competitions, our navigation methods had a high degree of inaccuracy due to the program's inability to have the sub readjust itself. However, because

of the simulator's ability to pinpoint issues and errors, our team has developed new navigational methods that are slower but more precise, while also optimizing different navigational aspects to compensate for the decreased speed.

Sticking to our goal of revitalization, we also focused on overhauling our previous software and replacing it with a new stack. This allowed us to utilize more recent C++ and Python versions, increase data distribution efficiency, and most importantly, simplify the code base so that new members could easily contribute new navigational approaches.

II. Design Creativity

With COVID disrupting the past two years, we were forced to temporarily halt our planned activities and testing. To make progress on our new vehicle during this time, each division came up with unique ways to deal with the unprecedented situation. The mechanical subdivision organized safe meetings to exchange "kits" consisting of parts and tools to construct separate pieces of the vehicle at home while the electrical subdivision adapted by hosting meetings online to discuss possible simplifications to the electrical configurations of our sub. The software subdivision focused more on specialized research, including microcontroller configurations, digital signal processing, and machine learning. Through both individual and group pursuits, we created all-new approaches in terms of both hardware and software.

A. Mechanical

1) Symmetrical Hull Design: In previous years, one of the largest problems with Marlin was the irregular horizontal cross-section that yielded balance problems. This caused Marlin to roll over and pitch forward when in motion. Changes to the overall shape of the hull were aimed at creating symmetrical cross-sections to increase control, leading to the design of two mounting planes that would evenly distribute the weight of the submarine. Now, Nemo's center of gravity is geometrically centered and allows for easier maneuverability and an increase in the sub's stability.

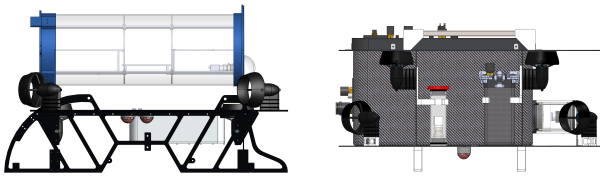

Fig. 1. Compared to Marlin (left), Nemo (right) has a more symmetrical design, allowing the sub to remain stable while moving forward at high speeds.

2) Mounting Plane Modularity: Not only do the mounting planes resolve Marlin's asymmetry, but they also improve the modularity. Modularity, or the flexibility of a system, was limited on Marlin due to its irregular frame. Components had to be redesigned to fit around odd shapes, creating unnecessary constraints. To tackle this problem, we designed six modular panels to be placed around Nemo. These 3D printed panels can be modified to seat any set of components, allowing us to switch out or change any panel while preserving the overall frame of the vehicle. For example, we added a hole to one of the panels to serve as the opening for the down camera. The mounting planes also allowed us to put multiple components outside of the MEB, including the batteries, hydrophones, and cameras. Each enclosure is not only fully independent and waterproof but also transferable. This allows both electrical and mechanical to use or replace these components without much hassle.

3) Carbon Fiber Manufacturing Process: An important aspect of constructing Nemo was manufacturing the physical components of the submarine, but due to the pandemic, we had to adapt and implement handcrafted strategies.

In the process of creating new watertight components for Nemo, the team first needed to decide which material to use. The main candidates were aluminum alloy and carbon fiber, as both could easily withstand the water pressure. For aluminum alloy enclosures, the manufacturing process would be relatively simple and the costs would fit within our budget. Furthermore, because the contour of the enclosures was an open-top box, metal welding could be used to construct them. However, after considering the cons of utilizing metal, we realized that metal enclosures would be difficult

to waterproof and would significantly exceed our target weight. Although requiring a more intensive production process, we ultimately chose carbon fiber because it met our strength and weight standards.

To meet our budget constraints and strict tolerances, the AVBotz team performed all of the carbon fiber enclosure production "in-house." Through discussions with experts in the composite industry and research into the process behind carbon fiber utilization, the team implemented a wet-layup and vacuum-bagging technique and recorded the process with detailed documentation. First, the team built our hotwiring cutter made from a spare power supply, PVC pipe, and nichrome wire. With our custom setup, foam was cut and glued to create the necessary external/internal molds of our components to provide a framework for carbon fiber. In preparation for the wet-layup process, we also cut carbon fiber fabric into specific 2D layouts that were nets of our 3D shapes (the enclosure designs).

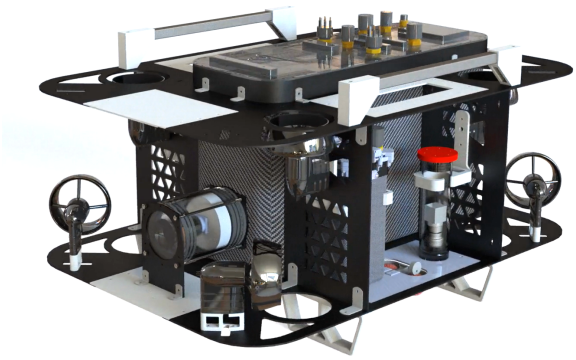

Fig. 2. Full model of Nemo that was used to guide manufacturing.

In the carbon fiber wet-layup process, the team brushed resin and layered resin-saturated carbon fiber sheets on top of the mold. The thick carbon fiber sheets supplied structural rigidity for our base layer while the thin carbon fiber sheets were used to fill gaps and provide a smoother mounting surface for other parts (camera enclosure, conduit panel, etc.). Then, we placed the mold and wet carbon in a vacuum bag to compress the carbon onto the mold. The vacuum additionally squeezed out any excess resin, making our final enclosure even lighter. To prevent galvanic corrosion due to contact between aluminum and carbon fiber, we anodized our parts with a Type - 3 hard coat at A&E Anodizing. Maintaining resiliency, ensuring cost-efficiency, and sustaining proficient strength-to-weight ratios, our modular components are attached to the sub through 3D-printed clamps and modular panels. Similarly printed handles are attached to the top mounting panel to enable simple portability of the entire AUV.

B. Electrical

1) Vertical Rack: Two key problems for the electrical subdivision that we aimed to solve with our new submarine were component accessibility and maintenance. The old electronics rack on Marlin was cramped and poorly laid out – electrical systems and components were generally hard to access, which compounded our difficulties in changing the batteries after a pool test or at the competition. We tackled this problem by 1) moving some components outside of the submarine, which will be touched on later, and 2) switching to a vertically-mounted, double-sided rack design. Instead of being mounted as a series of horizontal shelves like in Marlin, Nemo's rack is essentially a wooden frame, laser-cut and manufactured in-house, suspended vertically from the lid at the top of the MEB. This new organization drastically reduces the complexity of the wiring for the electrical team, provides more room to build on, and increases accessibility in an open layout from both sides.

Fig. 3. Compared to Marlin's rack, Nemo's new vertical rack design allows for easier accessibility to components for electrical members.

2) Components Outside the MEB: Another solution to fixing our limited access to components was to transfer parts out of the main electronics bay (MEB). In Nemo's design, key components are placed outside of our MEB, allowing for easy access. These include the batteries, hydrophones, and cameras, which all have their own enclosures. The independent boxes made moving the internal components much more convenient, allowing mechanical and electrical to easily access them. For example, swapping batteries in their own enclosure became a breeze compared to swapping batteries in Marlin, resulting in increased productivity in maintaining our new AUV.

3) Microcontroller Upgrades: Another pressing issue during the building of Nemo was the aging, legacy microcontroller and sensor system. Our old Marlin ran on a single underpowered Atmega2560 microcontroller with 256KB of flash and 8KB of RAM at 16Mhz, which placed a severe constraint on the low-level system. This limitation restrained the rate at which sensors could be sampled and the fusion and estimation algorithms that could be run on the microcontroller.

To remedy this issue, the team planned a complete overhaul of the low-level sensor and control system based on a philosophy of distribution. The new system consists of several distributed, much more powerful STM32G4 microcontrollers that communicate via the UAVCAN communication protocol over a central CAN-FD data network. An SLCAN adapter directly connects the main computer with the CAN network, and a ROS bridge node allows the publishing and subscribing of UAVCAN subjects directly in ROS, simultaneously reducing complexity by eliminating the need for a central board and more closely integrating the sensor and control system with the main computer. The resulting system solves the problems of the legacy system by adding much-needed processing power while increasing modularity through the distributed architecture of the UAVCAN/CAN network.

4) IMU Replacement: Another legacy component we identified as problematic was our TRAX AHRS, which, although accurate, had a high replacement cost if damaged. We determined that modern, inexpensive 6-axis MEMS IMUs and 3-axis magnetometers, combined with an open-source Madgwick [1] software fusion algorithm, would be able to provide similar performance at a much lower cost. Our approach combines a common, inexpensive ICM20689 MEMS accelerometer+gyroscope with a RM3100 magnetometer. We initially tested with an MPU9250 9-axis IMU; however, we determined that the IMU could not match the accuracy of the AHRS system it was replacing, and thus we chose the ICM20689 and RM3100 instead. Ultimately, the new AHRS system performs with the same accuracy while outputting data at a faster rate, improving Nemo's ability to calculate its orientation in the pool.

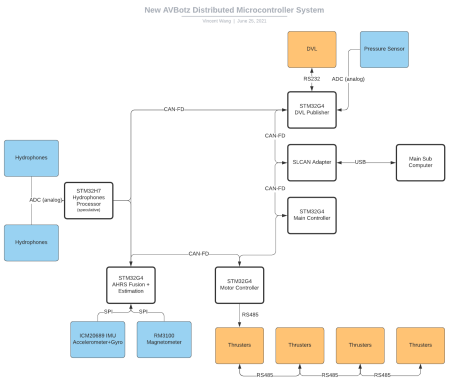

Fig. 4. New UAVCAN-based microcontroller system architecture with new IMUs integrated offers increased modularity, allowing team members to easily replace different parts and make upgrades.

C. Software

1) Simulator: In prior years, our main method of testing software was running pool tests; however, they were extremely time-consuming, prone to hardware failures, and limited to small pools. Thus, we opted to build an end-to-end simulator to evaluate the stack, which allowed us to test motor control and navigation in a competition setting, saving countless hours of debugging.

Using the UUV Simulator [2] plugin, the simulator is implemented in Gazebo. All sensors, including the Doppler-Velocity-Log (DVL) and inertial-measurement unit (IMU), are included in the simulator while the angle to the pinger is calculated manually. To mimic real-life settings, virtual cameras are calibrated with the same field

of view as their real-life counterparts, and noise is added to sensor data to create turbulent submarine movement. The sensor data is then published onto ROS topics, which the motor control process subscribes to in order to estimate the vehicle's position. To move the AUV, the control loop directs eight virtual thrusters, allowing the sub to yaw and translate on the horizontal plane to complete tasks.

We also needed to create a realistic visual scenario to test the vision system. In order to do so, green fog had to be added into the virtual world, and competition props constructed in Blender were exported into Gazebo. Because competition props in the real TRANSDEC shift slightly during each run, we mimicked the effect by using a Python script that randomizes the locations of props in each simulation, guiding software members to avoid brute-force navigational methods.

However, we discovered that despite the sensor readings with error, the simulator still appeared too idealistic, as the AUV moved in perfect lines. To fix this issue, a water current was implemented with continuously shifting angles and speeds, resulting in even more turbulence than a real-life current. With this, we could better judge the submarine's ability to handle turbulent conditions. We reasoned that if Nemo could perform tasks successfully in an extremely chaotic simulator, it would perform them well in a more peaceful pool.

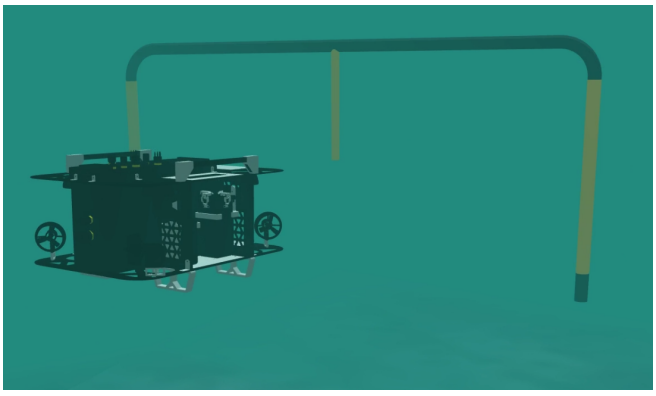

Fig. 5. Nemo attempting the gate task in the simulator.

2) Motor Tuning GUI: In the past, adjusting the aggressiveness of the motor controller was tedious and time-consuming, as it required recompiling and restarting the AUV to implement the adjustments. As a result, the motor gains of our submarine were not ideal, causing the AUV to move with inaccuracy and instability. To fix this issue, we created a graphical user interface that allows the user to tune the motors in real-time while remote controlling the submarine. Thus, when pool tests resume, this method is expected to save hours of tuning and allow the AUV to move with more stability.

3) Low-Level Control Upgrades: Previously, our state estimation only implemented a simple filter to fuse sensor data, and our microcontroller required a significant portion of the codebase to be written in AVR macros, making the control system gibberish to many of our newer members. Thus, our main priorities were to implement a Kalman filter to obtain a fused estimate of the submarine's position and to rewrite the code in readable C. These changes are aimed at providing more stable sensor data to Nemo while allowing new members to contribute easily to the low-level control process. Furthermore, the addition of more powerful microcontrollers enabled our team to begin utilizing the open-source Zephyr real-time operating system (RTOS) to provide hardware abstraction and threading, boosting the capabilities of the system.

The PX4 [3] autopilot was briefly considered for the new sub's control system. Its easily modifiable mixer system, which allowed quick configuration of frame types, was appealing to our team, and the implementation of advanced control algorithms, a direct ROS2 bridge, a working simulator, and a well-optimized extended Kalman filter made it a tempting choice. However, the team eventually decided against adopting PX4; the custom nature of Nemo's sensors made it difficult to integrate into PX4, and the system required a large amount of configuration. The Ardusub system was also considered and rejected for similar reasons.

4) Rewrite of Stack with ROS2: One major problem with our software stack was the accumulation of legacy, confusing code which made it difficult for new members to contribute. To address this productivity issue, the high-level guidance system was rewritten using Robot Operating System 2 (ROS2), C++11, and Python 3, allowing us to leverage more modern C++11 and Python

3 APIs. The rewrite also allowed us to redesign legacy sections of the codebase and improve code readability. In addition, using ROS2 has allowed us to better integrate with popular Python-based machine learning, robotics, and computer vision frameworks while also providing backward compatibility with legacy ROS1 nodes.

5) Computer Vision: This year, we focused on expanding our vision system's capabilities, while increasing the speed of our machine learning models. Thus, we explored the novel YOLOv5 [4] vision algorithm, which offers state-of-the-art accuracy at low hardware costs.

Previously, we utilized the EfficientDet-D0 [5] model for detecting objects in images because it was lightweight, easy to train, and extremely fast compared to old models. However, due to the quick-changing nature of the field of AI and computer vision, we quickly began searching for an improved model to replace EfficientDet with. After extensive comparison tests with other contemporary vision models, our team chose to implement a new vision system based on the YOLOv5 algorithm. Our new YOLOv5s based object detection model offers comparable accuracy to the old EfficientDet algorithm while running multiple times faster, at 87 FPS - a speedup of almost 3x. The lightweight nature of the new model allows easy simulation of the vision system even for inexpensive laptops and removes the need for a heavy, expensive, and power-hungry GPU onboard the AUV. Its high update rate also enables us to more quickly and accurately track objects in our camera feeds, improving the accuracy of our angle calculations and control code.

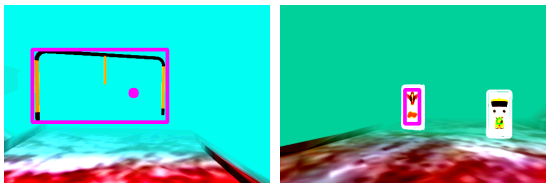

Fig. 6. Machine learning models detect the gate and buoys inside of the simulator, allowing the sub to calculate the angle to the tasks and turn towards them.

6) Spatial Awareness: In previous years, our method of calculating the angle to an object was inaccurate, as it assumed that pixel lengths were proportionate to degrees. To solve this problem, we developed a new method of calculating the angle which uses trigonometry calculations, specifically the inverse tangent of differences between coordinates, to arrive at the exact angle to the object. This allows us to turn the AUV with more precision, increasing success rates in tasks such as the gate and buoys.

Another major problem was detecting the distances to objects. Because the AUV does not have sonar sensors or depth cameras, we are unable to directly access distances to competition objects. However, we do have access to camera properties such as field of view, sensor size, and focal length. These attributes are then plugged into an equation [6] that calculates the distance to the object. This is a game-changer for our navigation system; instead of using a brute force approach to guess the distance to an object, the new instant calculation allows the AUV to deliberately approach an object, increasing the intelligence of the AUV and its overall efficiency.

7) Hydrophones: This year, we improved the accuracy of our hydrophones by migrating the approach from a basic Discrete Fourier Transform to the Multiple Signal Classification (MUSIC) [7] algorithm. This was a necessary change due to the around 95% accuracy of our previous algorithm in calculating the correct angle to the octagon. Our previous code contained quadratics to assist with angle-determination, and the math used to determine usable roots caused the 5% chance of error. After weighing the pros and cons of different methods, we decided on using the standard MUSIC algorithm because it was an efficient, fairly simplistic, and well-documented method. First, we take the amplitudes of the pinger's soundwaves from our uniform linear array of four hydrophones, spaced apart at 0.0127 meters. Our sampling rate is 286,000 samples per second, and we process these signals with a butter bandpass filter with a frequency of 25 kHz. Next, we calculate the time delay between the first element in the hydrophone array and each subsequent hydrophone, using this delay to find the steering vectors of the hydrophones and then adding back the original noise subspace. We use this to recreate the signal of the pingers and come up with

a covariance matrix, which we then pass into the MUSIC algorithm. The algorithm determines a specific value for a spectrum between 0 and 180 degrees. The peak of the spectrum is the most probable angle of arrival, allowing us to turn towards the pinger.

III. Experimental Results

A. Simulation

Because the pandemic prevented us from gathering for pool tests, we instead polished the software stack through simulations. In the simulator, the sub has access to virtual thrusters, and it can translate in six degrees of freedom akin to a real-life AUV.

Through the simulator, we realized that the old alignment method of using the vision system, calculating the distance that the sub needs to move to reach the object, and only moving once was inaccurate because calculations could be off by 1-2 meters, creating a clunky movement system that only had one chance to succeed. This year, the simulator allowed us to test new alignment methods that continuously readjusted the position of the sub, executing a rapid loop of calculating position offsets and adjusting the sub's position accordingly. Testing in the simulator indicated that this method elevated the success rate of the bins drop from 35% to 85%.

In addition, the simulator allowed us to test novel angle and distance calculations. Using the aforementioned updated approaches, the simulator proved that the angle calculation was accurate within +/- 5 degrees while the distance calculation was within +/- 1 meter. These were satisfactory for our purposes, and they transformed the way our AUV can navigate by utilizing distance measurements to accurately translate through the gate and buoys.

Finally, the simulator also allowed us to upgrade our method of obtaining style points; in the past, the AUV turned in 90-degree blocks. However, this was extremely time-consuming as the sub needed to slow down as it approached each waypoint, draining precious run time. But, through the simulator, a new method of spinning was implemented, which instead continuously set new waypoints 90 degrees from the current position in a rapid loop to prevent the sub from slowing down. This ultimately allows the sub to continuously spin, which saves time in competition and looks extra stylish.

Fig. 7. Nemo utilizes its vision system to continuously readjust and align itself to the bin in the simulator.

B. Water Testing Enclosures

To finalize the new submarine, we have been water testing the MEB, the two battery boxes, the hydrophones box, and the front camera enclosure. We started by filling the enclosures with water to detect any obvious leaks and then submerging

the components in members' pools at home while placing a GoPro inside each enclosure to help spot any points of leakage.

Through water testing, we discovered that the battery box initially had leaks through the screw-holes and the corners of the lid. To fix these leaks, additional sealing washers were used for the screws, and 3D printed pieces were added to each corner, increasing the clamping force in those areas. These changes greatly reduced leakage, but additional testing is required to ensure that no water gets in.

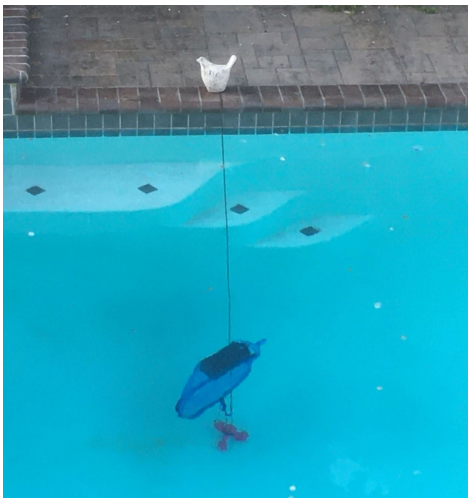

Fig. 8. Water testing the battery box to make sure that the enclosure is waterproof.

C. GPU Optimization

The Geforce GTX 1080TI is one of the most power-hungry parts of our AUV. With the implementation of the new vision systems, we explored the possibility of replacing the GTX 1080TI with a weaker GPU that can inference without drawing as much power. With the help of software members, we were able to test a variety of CPU/GPU configurations. With the benchmarks, we learned that a leaner GTX 1070 or an AMD Ryzen 7 3700x would be more than enough for inference. This meant that we would not have to utilize a bulky GPU in the AUV anymore, allowing us to scale down the electrical system and make the sub easier to construct and maintain.

ACKNOWLEDGMENTS

We would like to thank our sponsors for giving us the opportunity to create Nemo. We especially thank Datron, who has helped us machine countless parts for our submarines. We would also like to thank our club advisor, Mrs. Barnett-Dreyfuss, for guiding us throughout this project and helping us become the club we are today. We are grateful for members allowing the club to test in their backyard pools, as well as Amador Valley High School for letting us use their pool. The club is indebted to the dedication of our members, who have all contributed countless hours to construct our submarine. We also cannot forget the organizers of Robosub, who have allowed us high school students to gain valuable experience in a robotics team. Last but not least, we are all extremely indebted to our parents who have always supported us, both emotionally and financially. Our club would not be possible without the help of everyone above.

REFERENCES

[1] Sebastian Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 25:113–118, 2010.

[2] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. Uuv simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–8, 2016.

[3] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6235–6240. IEEE, 2015.

[4] Do Thuan. Evolution of yolo algorithm and yolov5: the state-of-the-art object detection algorithm. 2021.

[5] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] Matt Grum. How do i calculate the distance of an object in a photo?, 2011. https://photo.stackexchange.com/questions/12434/how-do-i-calculate-the-distance-of-an-object-in-a-photo.

[7] Lan-yue Zhang and De-sen Yang. Doa estimation based on music algorithm using an array of vector hydrophones [j]. *Journal of Harbin Engineering University*, 1, 2004.

APPENDIX A: COMPONENT SPECIFICATIONS

Component	Vendor	Model/Type	Specs	Cost (if new)	Status
Frame	Custom	Aluminum 6061 - T6	90.50cm x 63.50cm x 33.34cm	Sponsored	Manufactured
Main Waterproof Enclosure	In-House	Carbon Fiber Enclosure sealed with 2 EPDM O - Rings	62.23cm x 22.76cm x 35.24cm	$200	Manufactured
Battery Enclosures (2)	In-House	Carbon Fiber Enclosure sealed with single EPDM O - Ring	10.50cm x 10.50cm x 23.95cm	$120	Manufactured
Hydrophones Enclosure	In-House	Carbon Fiber Enclosure sealed with single EPDM O - Ring	13.00cm x 10.00cm x 19.62cm	$120	Manufactured
Waterproof Connectors	SubConn	Circular Series	(Varies Based on Series) Micro-Circular Series Power Series Coax Series	Reused	Installed
Thrusters	VideoRay	M5 Thrusters	90mm Length	Reused	Installed
Propellers	VideoRay	Standard Propellers	90mm	Reused	Installed
Motor Control	Rugged Circuits	Rugged MEGA	ATmega 2560 microcontroller, Arduino Protoshield	Reused	Installed
High Level Control	In-House	PID Control	Reused	Free	Installed
Batteries	ZEEE	6s	6000mAh, 22.2V, 260Wh	Reused	Installed
DC Converter	Cincon	Brick CFB600-300S	600W, 180-425V, 48V to 24V	Reused	Installed
CPU	Intel	i7-4790T	4 Cores, 8 Threads, Base Freq: 2.7 GHz, Turbo Freq: 3.9 GHz, Cache 8 MB	Reused	Installed
GPU	Nvidia	GTX 1080ti	Memory Type: DDR5x, 11 GB RAM, CUDA Cores: 3584, Max Power: 250W, Min Power: 200W	Reused	Installed
Motherboard	Jetway	NG9J-Q87 Mini ITX	4 USB 2.0 Ports, 2 USB 3.0 Ports	Reused	Installed

RAM	Corsair	Vengeance 16GB	2x8GB DDR3 SODIMM RAM	Reused	Installed
Storage	Samsung	ITB mSATA 860 EVO SSD	Max Seq Read/Write Speed: 550 Mb/s	Reused	Installed
Internal Comm Network	ROS	ROS2 (Eloquent)	-	Free	Installed
External Comm Interface	-	Ethernet	-	Reused	Installed
Doppler Velocity Log (DVL)	Teledyne Marine	Explorer DVL OEM	Type: Phased Array Transducer, Frequency: 600 kHz, Max Depth: 1000m	Reused	Installed
Altitude Heading and Reference System (AHRS)	PNI Sensor	TRAX AHRS	Static Heading Accuracy: 0.3°	Reused	Installed
Pressure Sensor	Ashcroft	K-17	Accuracy: ±1%, Range: Vacuum to 20000psi, Gauge Range: 15 psig, Input: 10-36V (DC), Output: 1-5V (DC)	Reused	Installed
Front Camera	FLIR	BFS-U3-200S6	Frame Rate: 30 fps, Resolution: 5472x3645, Megapixel: 20MP, Sensor Type: CMOS	Reused	Installed
Front Camera Lens	Computar	VO828-MPY	8mm fixed lens, Resolution: 12MP, Horizontal Angle: 77.3°, Vertical Angle: 61.7°	Reused	Installed
Down Camera	FLIR	BFS-U3-13Y3C-C	Resolution: 1280x1024, Megapixel: 1.3MP, Frame Rate: 170FPS, Sensor Type: CMOS	Reused	Installed
Down Camera Lens	Theia	SY125M	Focal Length: 1.3mm, Resolution: 5MP, Horizontal Angle: 125°, Vertical Angle: 119°	Reused	Installed
Hydrophones	Teledyne Reson	TC4013	Frequency Range: 1Hz to 170kHz, Depth: 700m max, -211 d13 ± dB receiving frequency	Reused	Installed
Signal Processing	Diligent	Nexys 4 DDR Artix-7	Block RAM: 4,860 Kbits	Reused	Installed

Algorithms: vision	Open source	EfficientDet-D0, YOLOv5s, RGB equalizing filter	87 FPS	Free	Installed
Algorithms: acoustics	In-House	MUSIC	Hydrophones	Free	Selected
Algorithms: localization, mapping	In-House	DVL data, image calculations	DVL, IMU, CV	Free	Installed
Algorithms: autonomy	In-House	Linear instructions	ROS2 nodes	Free	Installed
Open source software	Open source	ROS, ROS2, EfficientDet-D0, YOLOv5s, OpenCV, MUSIC	Node management, computer vision, digital signal processing	Free	Installed
Team Size (number of people)	29				
Expertise ratio (HW vs. SW)	13:10 (8 Mechanical, 5 Electrical, 10 Software, 5 Business)				
Testing time: simulation	50 hours				
Testing time: in-water	75 hours				
Programming Languages	C++, Python 3				

APPENDIX B: OUTREACH ACTIVITIES

During this past school year, the first main community outreach activity we organized was our newsletter. *AVBotz Monthly* was sent to more than 350 people and provided insightful updates on each division's monthly progression.

For the software division, we organized the AVBotz GitHub Repository, which provides a set of resources that are available for anyone to use. We have open-sourced 35,000+ lines of code on GitHub for other teams to access, and we encourage collaboration between developers across different teams to further the capabilities of combined knowledge.

In the future, we hope to host an AVBotz Open House, which our Business Team is currently preparing for. At this open house, elementary school students from our district will learn skills from the multiple divisions in our team. We plan to provide submarine-related coding workshops led by the Software Team and various hands-on activities led by the Electrical and Mechanical Team.

However, our most important activity was a community event geared towards middle schools. Last spring, the ACE Coding club and AVBotz held a PPIE Computer Building Workshop to combat the lack of hardware education within our school district's curriculum, and AVBotz helped by preparing components for the workshop and also passing on knowledge that they might have learned during their time in the club. During the workshop, local middle-school students were able to choose from 15+ courses, ranging from training machine-learning-based object detection algorithms to building their very own computers. To the teachers helping in the classes, it was an eye-opening experience to witness kids being excited about the field of STEM during our display. A middle school boy excitedly showing off an object detection model he had trained to his little sister made us satisfied that we had done our jobs well. Ultimately, we hope our workshop opened the children up to the possibilities of STEM, and that one day, they too, will build their own impressive computers or solve the mysteries of the universe.

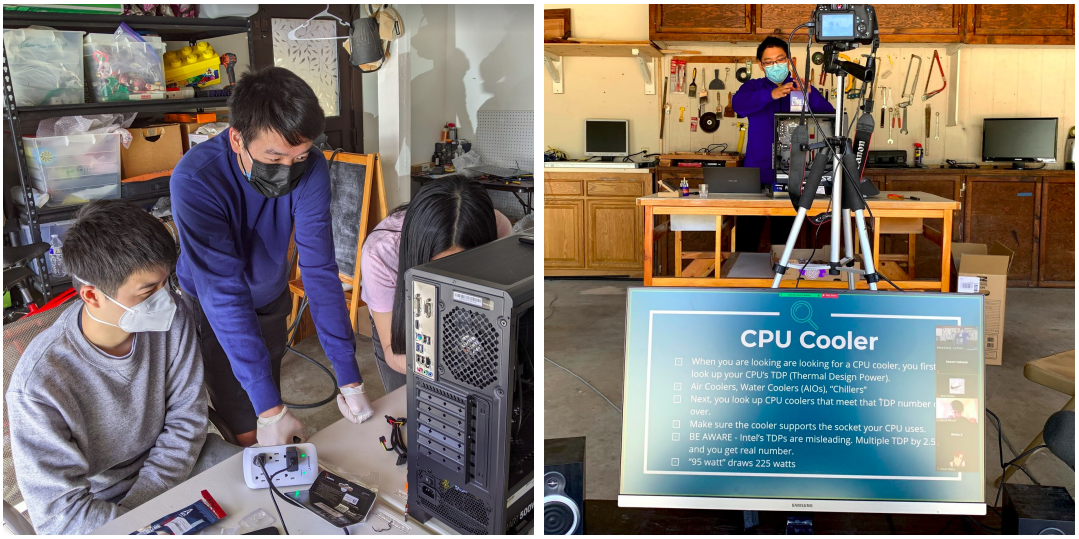

Fig. 9. Preparing and teaching Bay Area kids about tech.