SDSU Mechatronics 2022 Autonomous Underwater Vehicle (AUV): Perseverance

Tristan Richmond, Bhuvan Bhardwaj, Tran Ly, Cayton Larmer, Ian Reichard, Ken Ishio, Cameron Zamora, Brad Dela Llana, Jean Michel Vives, Yulianna Izaguirre

1

ABSTRACT - Perseverance is the Mechatronics AUV for the 2022 Robosub competition. The main goal for the team was to make a modular design that would be able to complete all obstacles but still let us focus on certain tasks. This strategic decision guided the design efforts, leading to a mechanical design with swappable components, a generalized electrical system capable of accepting any printed circuit board with a common interface, and ample opportunity for future expansion.

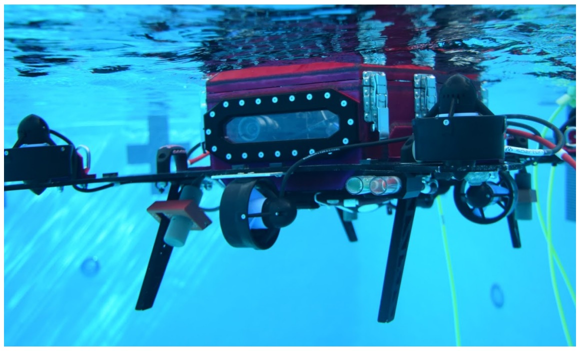

Fig. 1: 2022 RoboSub vehicle "Perseverance".

I. COMPETITION STRATEGY

Mechatronics' competition strategy for 2022 is to design an AUV that could be easily modified for various tasks. As a baseline, the main focus was to complete a few tasks with high confidence, with stretch goals down the line.

This year, our team prioritized the goal of reliably going through the competition gate, completing the Make the Grade (buoy) task, and rising in the octagon for the Cash or Smash task.

Mechatronics decided to use the side of Bootlegger for the gate task. For the Make the Grade task, we expect that our vehicle would be able to bump either buoy, as we focused on the vehicle being able to hit either for the highest chance of success during the competition.

Perseverance was designed as a baseline AUV with the intention of future modifications such that multiple AUVs are deployed in the RoboSub competition.

II. VEHICLE DESIGN

For the overall design of the AUV vehicle and its systems, the focus was on simplicity, accessibility, and modularity. We found that focusing on these aspects made the design more efficient for testing and debugging purposes.

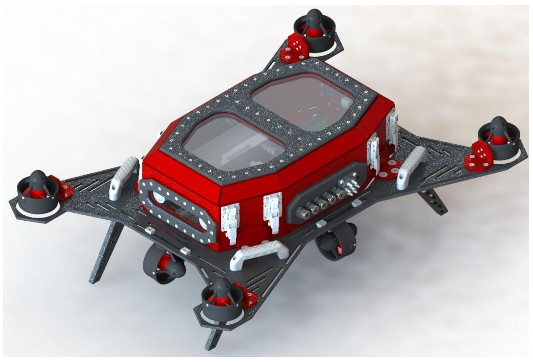

Fig. 2: CAD model of Perseverance.

A. Mechanical Systems

Perseverance's original design was influenced by criteria of accessibility and modularity. These foci led to the modular I/O port design, the clamshell hull and the external frame as being defining features of Perseverance in the 2017 - 2019 competition cycles. With the onset of the Covid-19 pandemic and the subsequent 2 year pause in club operations, a great deal of knowledge was lost within the club. More so than ever, the flexibility and ease of access allowed by the above design features have been critical to the success of the team this year.

Because of the loss of experience within the club, this year's rebuild of Perseverance has focused on themes of simplicity, consistency and modifiability. While this imposes additional limits on what the team is able to accomplish from both a technical and strategic perspective, it offers a number of critical advantages. The simpler nature of designs has greatly reduced the manufacturing requirements, with many components now being made of acrylic and 3-D printed PLA. These materials are both much cheaper than metals and are able to be manufactured in-house, saving time and money spent on outsourced components. Furthermore, this has allowed for less technically experienced team members to offer real contributions to the design and manufacture of the submarine. Perseverance's modular nature has also allowed for many components to be bought off-the-shelf, allowing further time saved.

Among the new components included in the Perseverance rebuild is the redesigned torpedo system. While the old torpedo system operated off of a pneumatic system housed inside the body of the submarine, the new torpedo system uses compression springs and a 3-D printed nylon housing. The firing mechanism consists of two servos which are controlled from a printed circuit board inside the submarine. This design was heavily influenced by the simplicity criterion; while the previous pneumatic system may have allowed for greater torpedo velocity, experience has taught the club that a point-blank approach provides a higher success rate for the torpedo challenge. Furthermore, the solenoids and pneumatic hoses needed for the old system required much more space and offered more points of potential catastrophic failure.

The orientation of the T200 thrusters has also been modified from the 2019 iteration of Perseverance. While the old orientation had the thrusters either normal or parallel to the forward-backward axis of the submarine, the new orientation has the thrusters offset from this axis by ±30°. By engaging all 4 vectored thrusters at once we can more efficiently make quick turns along with more efficient strafing, but this also adds more complexity to our thruster controller.

In order to test the effectiveness of Perseverance's aging sealing system, the mechanical and electrical teams developed a simple vacuum testing. Informed by the previous critical failure of an integrated valve, the new vacuum test takes advantage of Perseverance's modularity by attaching to one of the removable pneumatic ports, which is subsequently sealed with a screw before pool testing. While experience has demonstrated that vacuum tests cannot be used to strictly replace more rigorous underwater mechanical tests for find leaks, they do allow for a much faster pass/fail test of the sealing system that does not require valuable pool time.

B. Electrical Systems

Our electrical system is made up of four custom boards designed by our members: Weapons Gripper Board (WGB), Sensor Interface Board (SIB), Kill Auto Board (KAB), and Hydras.

All of our custom boards communicate via USB. We decided to use USB because of its ability for fast development and common interface on hardware devices. The main processor, the Jetson AGX Xavier, then communicates with all the boards through a USB Hub, simplifying our interfacing with the embedded systems.

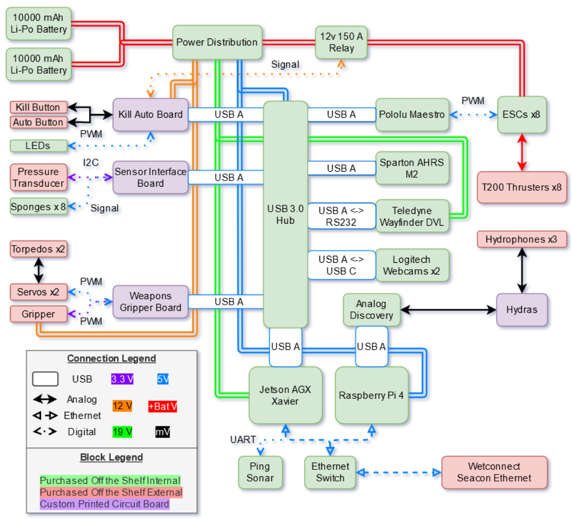

Fig. 3: Perseverance's electrical system block diagram.

Power system of Perseverance has been designed to supply a pure, uninterrupted power to the whole vehicle's peripherals. The power system and the distribution elements in the vehicle are designed in a way that it consumes less space and allow proper wire management. Two 10000 mAh lithium polymer(Li-Po) batteries are connected in parallel and used as the main power house of Perseverance. A parallel configuration is chosen to improve the endurance and the run time of the vehicle. The supply from the battery is then buck/boosted and distributed amongst the peripherals in the vehicle. Different voltage rails of 5V, 12V, and 19V are branched out from the power distribution board which are then provided to the motherboard, DVL, hydrophones, cameras, and the suite of sensors. A kill mechanism is also incorporated in the vechicle to safely halt the run of the vehicle in case of any emergency.

The kill switch is an integral part of any electrically operated vehicle, a kill switch ensures of a proper stop of a vehicle by cutting the power to the propellers or any actuation system to bring the vehicle to a complete halt. Perseverance is equipped with a kill mechanism based over a electromechanical switch(relay). When the relay is actuated the power supply to the thrusters is cut and the vehicle comes to a complete stop; allowing the vehicle to come to the surface, due to its slight positive buoyancy. A indication LED strip is also placed along the vehicles hull to ensure a proper signaling of the operating state of Perseverance.

Perseverance is equipped with a variety of sensors which allows the vehicle to sense the surroundings and provide feedback to the motherboard. Suite of sensors include pressure sensor, IMU (AHRS-8), DVL, Dual Cameras, Sonar Sensor, Hydrophones, and Leak sensor.

C. *Software Systems*

The software for this year is written in Python 3 for most processes, C is being used for our “master” process in charge of starting up other processes, and some bash scripts are used to set up our environment. C++ libraries with Python wrappers such as OpenCV are being used in our vision system. Using the AGX Xavier with Ubuntu 18, there is significantly more processing power on our AUV than previous iterations in other years.

Our programming team has gone through many leadership and member changes since the last in person competition, and the lessons previous members have learned have not fallen on deaf ears. We have developed a new software system from the ground up with our new team with the following design goals in mind: Reliability, Time to Develop, Modularity and Ease of Integration, Thorough Documentation, and Obstacle Detection.

Reliability is the most important metric for improvement for us as opposed to previous years. Our team has switched between various systems for interprocess communication and has struggled with implementing features such as validating working processes during autonomous runs. This year we have deviated from previous implementations using our in-house built mechos and in-house built docker containers with RPC. This marks our return to ROS (Robot Operating System), which is a reliable suite of tools for interprocess communication and industry standard in robotics. In addition to switching to ROS, we are implementing a feature-rich watchdog program which can restart failed processes while the AUV is fully autonomous. This helps improve reliability significantly compared to previous years.

Modularity and Ease of Integration was another important concern of ours. Having less "hard coded" components of our system and allowing things to be flexible allows us to work in real-world scenarios. An example of our code being modular is our USB device identifier, discussed in the operating system section. Since we are using ROS, we needed easy to integrate APIs that can listen to specific ROS topics as well.

Thorough Documentation is mandatory for team cohesiveness and legacy support. We want components of our code to be used for years to come, rather than for one year and needing to recode everything. We enforced our own style guide this year, modeled after the pip3 and Linux kernel standards, to go along with this design goal. In order for code to be merged to our beta branch on git, being checked and signed off on by the project manager was mandatory, as they were also responsible for approving any pull request.

Obstacle Detection has always been a difficult task for autonomous robots and is the autonomous task we will find ourselves working on the most. We plan on integrating various sensors into our inference system to precisely and accurately as possible determine what we are looking at. These include cameras with a machine learning model (see below in computer vision), a clustering or contouring algorithm, a sonar sensor to judge distance, and a sophisticated mission planner.

Possibly the biggest design change this year was our commitment to using ROS (Robot Operating System) for interprocess communication so our programs could reliably talk to each other. Using the wisdom from previous leads on the team we made the decision to switch to ROS because it was preferable to maintaining our own ROS alternatives in addition to actually building our AUV's code. As stated earlier, our biggest concern was reliability, and ROS provides that to us for little setup cost compared to writing our own alternative.

The Operating System (OS) of choice was Nvidia's build of Ubuntu 18 LTS for their ARM-based AGX Xavier computer. We integrated our code more into the operating system than previous years. While we share data with ROS, it is important to share data outside of ROS for reliability reasons, in case ROS itself may have an issue. Our watchdog program uses System V style shared memory blocks provided to us by the shared memory module in Python's builtin multiprocessing library. This provides a significant performance benefit compared to our last sub, Pico, which used RPC channels and Docker containers. We use UNIX pipes as well to forward data between processes such as frame data from the cameras.

Our electrical team has put together a wide suite of sensors for us to use in our control system. Our initial plan was to have a CAN bus early on but that was changed due to feasibility reasons. We had to change from that original plan as a result and use various USB connections to these microcontrollers instead. Accordingly, we've developed a modular system of being able to detect which USB device is which by getting hardware serial numbers and then solving for where the OS mounted these devices. This way when we open serial communications with each device, they are not hard coded to be on certain ports. This not only allows for rerouting of the electrical system without compromising our code, but also allows for newer devices to be added easily, once we get their serial numbers.

Control-wise, we wrote code for 14 possible movement configurations and hope to take advantage of the new vectored thruster design as much as possible. Our planned control system is to have multiple PIDs to balance each of our 6 degrees of freedom depending on our tasks, one dedicated for moving backward, and two dedicated to strafing left and right with our new vectored thruster configuration. We have some

Our computer vision system has not been altered too significantly from our last AUV. The main consideration we have now is adding a second camera, allowing us to see below our AUV. Our cv system uses common libraries such as OpenCV. The largest change is negotiation in our AI between different forms of vision rather than just relying on a machine learning model. In addition to the YOLO object detection system we have used in previous years, we plan on adding another inference algorithm for boundary detection, likely K-nearest neighbor clustering. We also plan to tie in various sensors such as our sonar sensor to judge distance to objects, along with our mission planner for making further inferences on what we should be seeing. These will allow us to better gather where we are and how to proceed legacy code we can use and port to be ROS compatible for this purpose. In addition, our wide variety of sensors can tie in to our control systems and allow for us to have better decision making and movement while in autonomous mode at competition.

Our graphical user interface (GUI) has been rebuilt from our last AUV, Pico. Many lessons were learned from the last GUI and far better design principles were used in the new GUI. We are still using the tkinter library with the Pillow library so we can convert OpenCV images and easily display them in a GUI from our cameras. Instead of opening UNIX pipes to share information, we now use System V style shared memory blocks, just like we do for our watchdog program. This allows for far more efficient communication between different threads and the main thread for displaying windows. We also support 2 cameras now in the GUI so we can see what is going on, as opposed to previous years where the camera display was not built into the GUI directly.

IV. ACKNOWLEDGMENTS

The Mechatronics team would like to thank the SDSU Engineering Department as well as our corporate sponsors: Leidos, Northrop Grumman, Altium, Solidworks, Nordson, and BrainCorp. We would also like to thank our faculty advisors Theresa Garcia, and Donovan Geiger for supporting the team.

V. REFERENCES

[1] J. Redmon and A. Farhadi, *YOLOv3: An Incremental Improvement*. Washington: University of Washington, 2018.

Appendix A: Component Specifications

Component	**Vendor**	**Model/Type**	**Specs**	**Cost(if New)**
Buoyancy Control	PVC Pods			
Frame	Custom 6061 T6 anodized aluminum, 0.25” Thickness			
Waterproof Housing	Custom 6061 T6 anodized aluminum, 0.25” and 0.5” Thickness			
Waterproof Connectors	Seacon	WET-CON		$300
Thrusters	Blue Robotics	T200		$169
Speed controller	Basic ESC	Bluerobotics Blue ESC	30 amp	$36
High Level Control	NVIDIA	Jetson AGX Xavier		$1499
Actuators	None			
Propellers	Blue Robotics	T200		$200
Battery	Hoovo	Li-Po Battery	10000 mAh 4s 100c	$160
Regulator	Mini-Box	DC-DC USB		$54.95
CPU	NVIDIA	Jetson AGX Xavier		$1499
Internal Comm Network	Custom			
External Comm Interface	Seacon	Seacon Cable		$1000
Programming Language 1	Python			
Programming Language 2	C++			
Compass	Sparton	AHRS M2		$1500
Inertial Measurement Unit (IMU)	Sparton	AHRS M2		$1500
Doppler Velocity Log	Teledyne	Wayfinder		$7500

Cameras	Logitech	Streamcam	Resolution: 640x480	$290
Sonar	Blue Robotics	Ping Sonar		$360
Hydrophones	Aquarian	AS-1		$1185
Manipulator	Blue Robotics	Netwon Gripper		$59
Algorithms: vision	You Only Look Once V3			
Algorithms: acoustics	Phase Difference			
Algorithms: localization and mapping	DVL, AHRS, Sonar			
Algorithms: autonomy	PID			
Open source software	OpenCV, ROS, PySerial			
Team size	32			
HW/SW expertise ratio	1.5			
Testing time: simulation	0			
Testing time: in-water	98 Hours			