SubjGator 9: Iteration & Improvement of a Modular AUV

Russell MacGregor, Daniel McAleer, Ethan Mitchell, Carlos Chavez, Ziyin Liu, Erik Neff-Flahan, Ali Skarshinski-Fred, Dr. Eric M Schwartz

***Abstract*—SubjuGator9 (Sub9) is the latest AUV from the University of Florida's Machine Intelligence Lab. Sub9's design emphasizes modularity and improves upon previous years' work to achieve the best results at the RoboSub 2026 competition. This year's design introduces new mechanical capabilities, an improved electrical architecture, and a rewritten mission planner. Rapid prototyping and testing were achieved through the modular design of each component of the sub. Through these design decisions, SubjuGator9 has proven to be capable of completing all 6 of the RoboSub competition tasks with core capabilities, 5 of which with advanced or disruptive capabilities, while also being able to easily adapt to new circumstances.**

I. Competition Strategy

A. Core Strategy

We aimed to complete all six tasks with the highest possible capabilities (disruptive) and only scaled back our goals as obstacles arose. This was only the case with Task 5 (octagon) due to time constraints. Sub9′s platform enables the pursuit of all tasks independently, as it is designed for modularity in both hardware and software. We selected this approach to allow for Sub9 to adapt to future changes to the rule set and changes to the point values assigned to tasks. We did not opt for a dual-vehicle strategy because the lengthy construction time is better spent on improving the reliability of our existing vehicle.

B. Capabilities

Task 1 (start gate) depends on computer vision and submarine control (style points). The vision model and yaw style points were adapted from RoboSub 2025, but the roll style points required turning off our Doppler Velocity Log (DVL) because the DVL would lose track of the pool floor. Still, we attempt this task because the resulting odometry drift is minimal. However, the buoyancy of the sub prevents us from attempting *pitch* style points. We decided to keep our buoyancy consistent and disallow pitching in order to not compromise our overall reliability. Task 2 (slalom) depends on vision and was completed second, being adapted from RoboSub 2025. Task 3 (bins) was similar to the bins task from RoboSub 2025, which we did not complete at the time. We designed a new dropper mechanism using metal marbles to allow for straight drops. The addition of a down-facing camera allows Sub9 to better locate the bins, which are more complex than in RoboSub 2025. Task 4 (torpedoes) uses a new spring-loaded torpedo mechanism which possesses just enough power to score maximum points for firing at a distance. We assessed an alternative mechanism using powered torpedos, but the immense complexity was undesirable. In Task 5 (octagon), Sub9 is limited to locating the pinger and surfacing, due to time constraints on creating a gripper. This is still under development and may be ready by the time of the competition, but we will only attempt higher capabilities if we can make the gripper sufficiently reliable. Task 6 (return home) depends on good localization which Sub9 possesses with its DVL, allowing it to reliably retrace its steps.

C. System Changes

Sub9 has undergone major mechanical changes in the form of 3 new mechanisms, which are enabled by the switch to more reliable underwater servos. The modular design of the Sub9 platform allowed these mechanisms to be separately developed and easily integrated. We have also revamped the software stack to use a behavior-tree based mission planner, allowing for missions to be

Fig. 1. Sub9 full CAD assembly

expressed as a composition of simple operations, ultimately enabling faster iteration on missions.

II. Design Strategy

A. Sub9 returns to competition

Sub9 made its first appearance at RoboSub 2025. It will return to the competition again this year with significant upgrades to its subsystems. From revitalized mechanisms to reinvented software, Sub9, as shown in Figure 1, has never been stronger.

Our design strategy for 2025 centered around entering the competition with a new vehicle. SubjuGator 8 had just been retired, and Sub9 was new and untested [1]. We prioritized basic navigation and vision capability, while mechanisms remained largely underdeveloped. RoboSub 2025 was very informative at revealing Sub9′s strengths and its many areas needing improvement. This year, our design strategy involved bringing full functionality to Sub9, cementing in place its capabilities and subsystems.

B. Mechanical Design and Systems Integration

a) *Depth Rating:* The pool at the Woollett Aquatic Center for RoboSub has a maximum depth of 13 ft, which is approximately 4 m [2]. The shallowest rated components are the plastic tubes used to hold electronics, like the navigation tube and downcam tube. The navigation tube houses the inertial measurement unit (IMU), pressure sensor, and hydrophone electronics, and the downcam improves accuracy for dropper tasks. These tubes were selected with a depth rating of 33 m and a safety factor of 3 [3], which is sufficient for the competition. Also, the computer box, which is a hollowed-out block of aluminum stock, was designed with a wall thickness calculated to withstand a vehicle depth of 35 m with a safety factor of 3 [3].

b) *Mechanisms:* Mechanical updates have focused on mechanisms for task completion, which were non-functional or incomplete last competition season. A major problem from last competition was waterproofing the servos used for the mechanisms. Therefore, we replaced all servos with SER-2020 underwater servos. We designed new mounts to accommodate this geometry, strategically attaching them close to cameras to improve task accuracy. The ball dropper design, shown in Figure 2, had its mounting patterns updated to accommodate the new downplate and the new servo to complete Task 3.

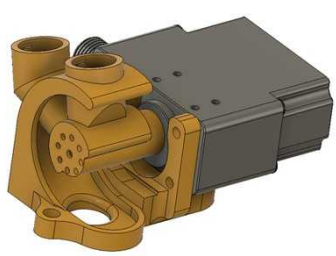

Fig. 2. Ball Dropper CAD Assembly

To further aid with computer vision implementation,the dropper was placed as close to the downcam as possible to help aim the markers when dropped.

The torpedoes previously showed vertical instability due to a lack of fins, making them incapable of completing Task 4. This led to a complete redesign of both the launcher and the torpedoes since the original launcher was unable to accommodate fins. In the updated design, shown in Figure 3, the torpedoes enter slots, compressing a 45lb_f/in spring. They are rotated into place and secured by friction and a detent bump. A servo-driven switch triggers the torpedoes by releasing the fins from a detent bump. Previous iterations of this switch used 3D-printed interfaces, which often deformed under the force of contact. Therefore, the switch contains a shaft collar which uses a shoulder screw as the set screw to distribute

the force of contact with the torpedo fins over a greater surface area.

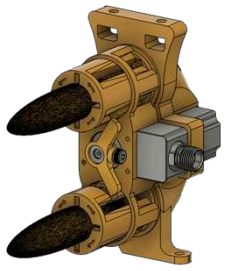

Fig. 3. Torpedo Launcher CAD Assembly

We fine-tuned the torpedoes to be neutrally buoyant with 78% infill density in order to aid with straight-and-level motion. Upon release, the spring applies 39.375 lb of force to the torpedo, resulting in 17.23 lb*in of kinetic energy. This energy is more than enough to launch the torpedo 1.5 m (60 in) for the disruptive tier of Task 4.

C. Cameras

Sub9 incorporates two cameras. One camera faces forwards, the second faces downwards. The forward-facing camera is used thoroughly for vision-related purposes across all tasks. The downward-facing camera is primarily used for the dropper and grabber-related tasks.

a) *2025 Cameras:* In 2025, our team consistently encountered challenges with the cameras. We used Dell UltraSharp Webcams for both cameras. We selected these cameras for their high resolution (4k @ 30Hz refresh) and field of view (65, 78, 90°) [4]. These cameras come with supporting structures meant to support them on a tabletop. We removed these supports, left the camera body intact, and mounted them within Sub9′s camera tubes. These cameras were too large in diameter to mount flush against the camera tube's lid. Consequently, a large black ring appeared in all camera frames. We could not take advantage of the camera's high FOV, as the camera tube obscured it. Additionally, visibility underwater was poor and inconsistent. The camera included automatically-adjusted HDR, light correction, and auto-focus settings that could not be adjusted externally. While helpful for video conferencing applications, these features were difficult to control.

b) *2026 Cameras:* This year, we replaced the Dell UltraSharp Webcams with Blue Robotics Low-Light USB Cameras. These cameras are purpose-built for use on a vehicle and for visibility below water. We selected these cameras since their resolution and field of view were similar in quality to the previous cameras (1080p @ 30Hz and 80°) [5]. The Blue Robotics cameras[1] have a significantly smaller footprint than the Dell UltraSharp cameras[2], occupying only 5.33% of the volume of the Dell UltraSharp cameras. This allowed us to use smaller camera tubes for Sub9, providing more space for cable management and mechanisms. Additionally, the Blue Robotics cameras incorporate onboard H.264 compression technology, reducing processor load on Sub9′s primary computer. The cameras' lower pixel count (2MP) and slightly lower resolution (1080p) also reduced processing time for vision models, allowing Sub9 to capture and process images with lower latency.

D. Electrical Architecture

Sub9′s underlying electrical structure underwent several changes to improve overall reliability and performance.

a) *13.9V 10A Buck/Boost Converter:* At RoboSub 2025, Sub9 did not incorporate any major regulator between the battery and downstream electronics. Batteries delivered power to a passive PoE ethernet switch, which then powered the Jetson Orin and Raspberry Pi computers responsible for normal operation. During testing, our team noticed occasional loss of signal from several sensors onboard Sub9, particularly when making contact with pool walls or the floor. Our team also noticed that sensor data became increasingly inconsistent as battery voltage decreased. We determined that a change was needed to prevent sensor brown-outs and other power-related issues caused by battery movement. We added a 13.9V 10A buck/boost regulator between the batteries and all downstream electronics, aside from thrusters. With this regulator, all sensitive

[1]Blue Robotics camera volume: ~65.92 cm^3 [5]

[2]Dell UltraSharp camera volume: ~1237.68 cm^3 [4]

electronics receive constant voltage regardless of battery voltage.

b) *Battery Monitoring, ORing, Power Path Management:* Sub9 previously operated on a single 9000mAh 14.8V 4S2P battery. From testing, Sub9 could operate for approximately one hour before the battery voltage dropped from its fully charged voltage (16.8V) to 13.8V. To prevent overdischarging the battery, we replaced batteries before their voltage dropped below 13.8V. Sub9 had no system to prevent the battery from being overdrawn and damaged. The only method of checking battery voltage was manually checking the cell monitor located in the battery tube. This monitor would audibly chirp when battery voltage reached the cutoff voltage. In 2025, one of our batteries was retired due to damage from overdraw. Although the retired battery was equipped with the cell monitor, it was insufficient to alert the team of the battery's low voltage. To address this, we replaced the cell monitor with a custom board stack that oversees battery monitoring and ORing. With the new system, Sub9 can be run off of two 4S2P batteries. Using power ORing, Sub9 draws power from the battery with the higher voltage. This doubles Sub9′s effective battery life and provides the team with more time to conduct tasks without worrying about battery replacement. Additionally, the new system reports battery and individual cell voltages directly to Sub9′s computer. If cell voltages drop below accepted parameters, the computers onboard Sub9 will receive a warning from the battery monitors. If the batteries are not replaced and Sub9 continues to draw from a depleted battery, the new monitor electrically isolates the battery via high-impedance back-to-back N-channel MOSFETs.

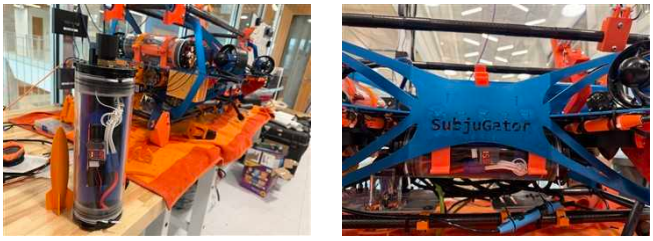

Fig. 4. Battery Tube with old cell monitor

c) *Improvements to Thruster PWM Control Board:*

Sub9 utilizes a NVIDIA Jetson Orin NX on a SeeedStudio J4012 carrier board as its primary computer. In this configuration, only one hardware PWM channel is available [6]. Sub9 incorporates eight brushless thrusters for movement. These thrusters are controlled via electronic speed controllers which require a PWM input. Sub9 has an intermediary PCB between the Jetson Orin and the thruster speed controllers: the Thrust Control Board. This PCB is connected to the Jetson over USB. We have improved on this PCB since RoboSub 2025. Primarily, this first iteration incorporated a boot jumper that, if connected, would inadvertently short VCC to GND. The new iteration of this PCB no longer has this hazard and is considerably smaller in footprint. Additionally, the new iteration of this PCB features extra I2C connections. We will add temperature sensors to Sub9′s computer enclosure to monitor for overheating; these sensors will communicate with the Jetson using these exposed connections.

d) *Relocated depth sensor:* Sub9 uses two sensors to gauge depth when submerged: a Doppler Velocity Logger and a pressure sensor. The latter is a simple device, communicating depth information to the Jetson over a single analog voltage pin [7]. At RoboSub 2025, we located this depth sensor on Sub9′s Navigation Tube. The Nav Tube was not present in Sub9′s original design; it was added shortly before RoboSub 2025 to provide accessible connections to Sub9′s hydrophones and primary IMU. Since the original design and placement of the Nav Tube were rushed, this placed the depth sensor near the wake of the front-left thruster. When this thruster spun up, readings from this sensor became inaccurate. At sufficient velocity, the thruster's wake would overpower the sensor to the point where it could not return any data. Since RoboSub 2025, we have relocated this sensor to the underside of Sub9′s primary computer enclosure, away from thruster wakes. Readings from this sensor are now significantly more consistent and accurate.

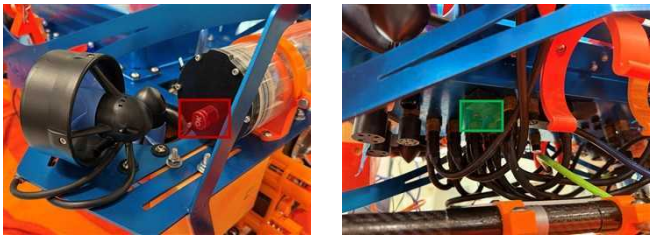

Fig. 5. Previous vs. current placement of pressure sensor

E. Mission Planner

Sub9 features a new Mission Planner, rewritten for improved reliability. In RoboSub 2025, Sub9 used a mission planner written in Python and designed around modularity. Unfortunately, the Python interpreter used for this mission planner failed to identify runtime errors before execution. We rewrote Sub9′s mission planner in C++ to take advantage of its more robust and comprehensive error checking during compilation.

Additionally, C++ executes significantly faster than Python in terms of real-time data processing and computation, which is critical for a mission planner [8]. Likewise, a C++-based mission planner integrates better with Sub9′s underlying Robot Operating System (ROS) architecture. In the context of ROS, nodes written in C++ perform better than nodes written in Python since C++ nodes can be compiled directly into machine code. Python nodes consistently require more computational resources and overhead, leading to measurably higher power consumption and longer processing times [9]. Critically, Python's Global Interpreter Lock (GIL) prevents Python-based threads from running in true parallel, a documented source of latency in ROS2′s `rclpy` MultiThreadedExecutor library [10]. Threads written in C++ use the `rclcpp` library which does not have this limitation. Using C++ instead of Python also allowed for use of C++'s widely used and well-documented behavior tree library. The previous mission planner used a custom-made framework to emulate the functionality of the C++ library. While this framework functioned adequately, the C++ library is more optimized and feature-rich.

The new mission planner performs much better than the previous planner in terms of reliability. Interestingly, we have not yet noticed a major improvement in terms of execution and performance as expected. We may need to further optimize our software to achieve these improvements. The new mission planner also has a somewhat higher barrier to entry. Newer software team members needed more time to adjust to the C++-based mission planner compared to its Python predecessor.

III. Testing Strategy

A. Software Testing

To ensure the reliability of our software systems, our team emphasizes an incremental testing approach. We divide up our tasks into comfortably small components, which we thoroughly test before continuing. This allows us to create small sanity checks that verify expected behavior early. These tests help confirm which parts of the system are functioning correctly and narrow down the sources of problems when unexpected behaviors arise.

This same testing philosophy is brought into our mission planning. We break down larger autonomous behaviors into smaller discrete mission components that we can independently test before combining them into more complicated missions. This modular structure makes missions easier to debug, improves reliability of components in the behavior tree, and allows the team to reuse validated components across multiple missions.

One software component that required individual testing was evaluating new YOLO models. Because perception results directly influence mission-planning decisions, we found it important to test each model independently before integrating them into the full autonomy stack. This allowed us to determine whether unexpected behavior during a mission was caused by the vision model itself or by the surrounding mission logic.

A vital part of testing our software components is Gazebo Simulation. Simulation allows us to evaluate the validity of our mission logic in a controlled environment. To maximize robustness, we run missions under multiple simulated conditions, including different vehicle starting positions and object placements. By identifying software issues in simulation, we can iterate more quickly and reduce the amount of in-water testing time spent on preventable software bugs.

B. Electrical Testing

The modular design of Sub9 allowed for rapid and efficient integration of new electrical hardware. Each team member assigned a task for the upgraded Sub9 completed each step of the design process, with final integration and testing overseen by a lead team member.

When integrating new hardware onto Sub9, after completing any wiring or soldering tasks, we tested the board in isolation. In this stage, we analyzed printed circuit boards for faults and unintended behavior. We also characterized the power consumption of the boards in this phase of testing. We then compared the board's behavior against simulation results and breadboard measurements.

Upon determining the board or piece of hardware behaved as intended in isolation, we integrated it into its intended subsystem. We took care to ensure proper connections were made and there was no potential for reverse-biasing a component or causing a short circuit when connecting the different devices present on Sub9.

C. Interdisciplinary Testing

a) *In-Lab Testing:* Before Sub9 was taken to a pool to be tested in the water, each component would often do a "dry run", where the electrical, mechanical, and software components of a task were tested before they were attempted in the pool. This allowed for the tasks to be fine-tuned in a controlled environment before they were deployed in the pool, and saved time detecting issues before the sub travelled to an off-campus location to be tested. While not every element of Sub9 was able to be tested in this manner, as many features require the sub to be in the water, testing ahead of time helped pinpoint different modes of failure and saved time.

b) *In-Pool Testing:* From August of 2025 through March of 2026, testing was conducted once per week in a local pool. This testing schedule was ideal, as it allowed for continual testing of added features and matched the pace of development. As the spring semester ended and the summer semester began, the pace of development hastened and the need for more testing sessions arose, the number of testing sessions was increased to three times a week. To prevent weather from interfering with testing, the time of day was changed if the forecast was unfavorable for testing.

Prior to each testing session, a list of objectives for the day was created. Frequently this included testing new missions and models, and determining which were more useful. Prior to the testing session, props related to each of the tasks were created based on the specifications in the rule book. When Sub9 was tested with the props, the team was successfully able to use the props to train the models and practice full runs of the competition.

Acknowledgements

Our work getting Sub9 ready for RoboSub would not be possible without the help of our faculty leaders and sponsors. Dr. Schwartz has led the Machine Intelligence Lab since 2002. Under his oversight, the lab has grown from a small group of students to a preeminent research team on campus. Dr. Schwartz's leadership has been instrumental in bringing our lab to where it is today.

Many of our alumni have also been foundational in getting Sub9 ready. Cameron Brown, who wrote much of Sub9′s early codebase, has been a regular point of contact for questions pertaining to ROS2 and Sub9′s architecture. Adrian Fernandez, who designed the thrust control board, continues to provide guidance for Sub9′s electronics. Forrest Voight, who built Sub9′s hydrophone system, is the founder and CEO of Sylphase LLC, which provides our lab with generous financial and technical contributions [11]. Israelle Widjaja, a cofounder to Sylphase LLC, who also assisted with the purchase and procurement of all lab-related custom printed circuit boards. Other notable alumni include Adam McAleer, Daniel Parra, Lester Bonilla, Sophie Lanahan, Lorant Domokos, Yovany Molina, and Adam Hamdan. These alumni were instrumental in building the early stages of Sub9′s electrical, mechanical, and software systems. Without them, Sub9 would not have been possible.

References

[1] L. Bjellos *et al.*, "SubjuGator 2024: Design and Implementation of a Modular, High-Performance AUV." [Online]. Available: https://drive.google.com/file/d/1DFxqyUuK7DbCt2yr4aE37FovDm_E7mxs/view?usp=sharing

[2] "How deep is the pool?." [Online]. Available: https://www.yelp.ca/questions/william-woollett-jr-aquatics-center-how-deep-is-the-pool/yGDEg_TWbvkDXAc2TP0gwg

[3] A. McAleer *et al.*, "Design and Development of the SubjuGator 9 Autonomous Underwater Vehicle." [Online]. Available: https://robonation.org/app/uploads/sites/4/2025/07/RS25_TDR_University-of-Florida-MIL-Subjugator.pdf

[4] Dell Technologies, "Dell UltraSharp Webcam WB7022." [Online]. Available: https://www.delltechnologies.com/asset/en-us/products/electronics-and-accessories/technical-support/dell-ultrasharp-webcam-wb7022-data-sheet.pdf

[5] Blue Robotics, "Low-Light HD USB Camera." [Online]. Available: https://bluerobotics.com/store/sensors-cameras/cameras/cam-usb-low-light-r1/

[6] Seeed Studio, "reComputer J401 Schematics." [Online]. Available: https://files.seeedstudio.com/wiki/J401/reComputer_J401_SCH_V1.0.pdf

[7] SSI Technologies, "P51 Pressure Sensor." [Online]. Available: https://www.ssi-sensors.com/perch/resources/documents/p51-pressure-sensor-p51-ds.pdf

[8] Debian, "The Computer Language Benchmarks Game." [Online]. Available: https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

[9] M. Albonico, M. B. Cannizza, and A. Wortmann, "Energy efficiency in ROS communication: a comparison across programming languages and workloads," Apr. 01, 2025. [Online]. Available: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2025.1548250/full?trk=article-ssr-frontend-pulse_little-text-block

[10] Open Robotics, "Executors." [Online]. Available: https://docs.ros.org/en/foxy/Concepts/About-Executors.html

[11] Forrest Voight, [Online]. Available: https://sylphase.com/