Design and Development of the SubjuGator 9 Autonomous Underwater Vehicle

Adam McAleer, Joseph Goodman, Lester Bonilla, Ethan Mitchell, Adrian Fernandez, Carlos Chavez, Ryan Hoburg, Cameron Brown, Lorant Domokos, Alex Johnson, Keith Khadar, Adam Hamdan, Daniel Parra, Eric Schwartz

***Abstract*—SubjuGator 9 is an Autonomous Underwater Vehicle designed to compete in the RoboSub competition. The vehicle is designed to be a platform with strong fundamental capabilities, such as localization and movement, that can be improved upon with competition-specific systems. SubjuGator harmoniously combines mechanical, electrical, and software systems into a robust autonomous system. The mechanical systems provide a configurable base for mounting electronics and mechanisms. The electrical systems power the motion of the vehicle while providing connectivity between systems. Software systems are implemented in ROS2, facilitating asynchronous processes and task completion. The team uses a rigorous schedule to constantly test and debug the systems.**

I. Introduction

SubjuGator 9 (Fig. 1) is the latest iteration of the SubjuGator platform, a family of autonomous underwater vehicles (AUVs) that have been in development for almost 30 years. The RoboSub 2025 competition is the first deployment of SubjuGator 9.

Much of SubjuGator 9 was originally designed before the COVID-19 pandemic, but was delayed due to in-person work restrictions. Further delays resulted from a lack of transfer of knowledge during the pandemic, as new and existing members of the lab were not able to work together. As a result, the electrical and software systems for SubjuGator 9 were completely redesigned during 2024 and 2025. This complete redesign allowed for a flexible approach to competition strategy.

As RoboSub 2025 will be the first deployment of SubjuGator 9, the team's goals for the competition are to have a functioning baseline vehicle capable of movement, localization, and basic task completion. All sensors, actuators, and software packages should be robust to facilitate the later addition of capabilities.

II. Competition Strategy

Our competition strategy revolves around developing robust, well-understood, and well-documented modules that provide a platform that is easy to extend and debug. We use a bottom-up approach to determine what features are necessary to complete each task. We then focus our efforts on features with the widest applicability, where the potential to score points versus the time spent developing is highest.

To manage the growing complexity of the system, we independently test new features before integrating them with the rest of the system. This helps ensure reliability at the subsystem level, reducing errors during integration.

Fig. 1: SubjuGator 9

We focus on guaranteeing points in competition with reliable and functioning systems, rather than spending time implementing new features. As competition draws closer, we prioritize testing and completing tasks that leverage the strengths of our vehicle. For example, quality localization and precise movement will secure points in tasks that use these behaviors, such as the navigation channel and dropper tasks.

Our strategy places emphasis on consistent and structured in-water testing. As the competition approaches in five weeks, over the last six weeks, we have performed over 15 testing sessions each of about five hours in length. Our feedback loop for testing sessions involves writing notes at the end of each session about the successes, failures, discoveries, and hypotheses made during the session and our plan of action before the next session. We use these notes to drive development and adjust expectations for our timeline.

Review of the week's successes and failures motivates us and keeps the next actionable step clear and unobstructed by the growing complexity of the system. This part of the strategy increases the confidence of the team in working with the system in water. This familiarizes the team with problem solving through collaborative debugging in a competition-like environment. Increasing our efficiency during in-water testing sessions will directly increase our efficiency during our limited

pool time at the competition venue.

III. Design Strategy

To support our competition strategy and maximize points earned at the competition, SubjuGator 9's design is extensible and easy to test. Our goal is to maintain a platform that can adapt to changing competition requirements. Using our bottom-up approach, we design and develop a minimum viable product with baseline capabilities of localization and movement control. All additional subsystems are then guaranteed to have a functional base.

A. Mechanical

1) Structure: SubjuGator's frame is made from four aluminum sheets welded together. Hardware, pressure vessels, and thrusters attach to the frame via machined slots and mounting holes. Eight carbon fiber tubes provide structural support and the ability to easily mount hardware. The computer box, a hollowed-out block of aluminum stock, and the frame are anodized blue for both corrosion resistance and visual appeal. A waterjet-cut aluminum bottom plate is mounted below the computer box to support additional hardware.

The computer box houses the main electronic systems for the vehicle, including the main computer (a Nvidia Jetson Orin NX), power delivery systems, and networking devices. The minimum wall thickness of the computer box was calculated to withstand a vehicle depth of 35 m, with a safety factor of 3.

2) Kill Wand: SubjuGator's kill wand mounts to the carbon fiber rods. The mount is re-positionable by sliding it along the rods. A magnet at the end of the kill wand that interacts with a hall effect sensor on the inside of the computer box lid. The kill system is easy to trigger by pulling the wand.

3) Thruster Configuration: Eight thrusters move the vehicle with 6 degrees of freedom. The configuration allows for the loss of a thruster without losing a controllable degree of freedom.

4) Depth rating: An important consideration when designing the vehicle was determining a maximum depth rating, based on the depth of the shallowest rated component. The components susceptible to mechanical failure due to high pressure at depth are the computer box, navigation tube, camera tubes, hydrophones, and doppler velocity logger (DVL). The component to fail at the shallowest depth would most likely be the camera tubes, rated for a depth of 33 m with a safety factor of 3. This maximum depth was determined to be acceptable, as use cases of the vehicle are the RoboSub competition (with a maximum depth of 2.1 m) and the deepest testing environment the team has access to has a maximum depth of 5 m.

5) Mounting: SubjuGator 9 is designed to be easily mechanically configurable. All pressure vessels are secured with custom 3D-printed brackets which are closed using clevis or quick release pins, allowing tool-less maintenance to the vehicle. Large blocks of non-compressible foam cam be attached to the carbon fiber rails using snap-on clips, allowing for quick removal during transport of the vehicle.

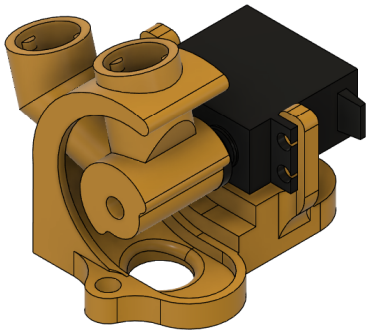

Fig. 2: Dropper

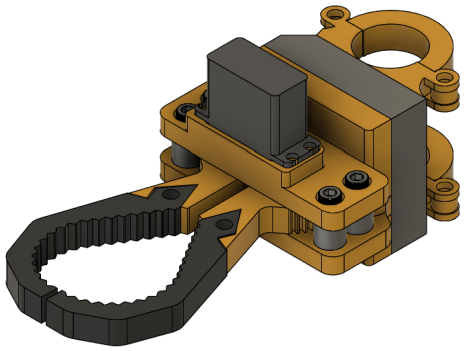

Fig. 3: Gripper

6) Buoyancy: SubjuGator 9 uses non-compressible foam to maintain a positive buoyancy of more than 1% of its weight (above the 0.5% required). Exceeding the requirement reduces the need to reconfigure buoyancy as new components are added to the vehicle. Non-compressible foam is superior to cheaper and more readily available alternatives, such as pool noodles, because its flotation changes minimally as a function of depth and time deployed.

7) Mechanisms: SubjuGator 9 has a dropper (Fig. 2), gripper, and torpedo launcher (Fig. 3). These mechanisms are simple and quickly replaceable since they use 3D-printed parts and the same waterproof servo. They are strategically mounted near cameras to improve task accuracy.

B. Electrical

1) Electrical Compartments: SubjuGator has two main electrical systems: the computer box and the navigation tube (NavTube). The computer box houses the main computer, an Ethernet switch, and power systems. The NavTube houses three sensors: the inertial measurement unit (IMU), pressure sensor, and hydrophone electronics. One cable connects the NavTube and computer box, which provides power and enables data transfer.

The NavTube separates the IMU from high current components that cause interference. The NavTube is easily removable and serviceable. Electrically, the only requirement to interface with the NavTube is a power over Ethernet (POE) switch.

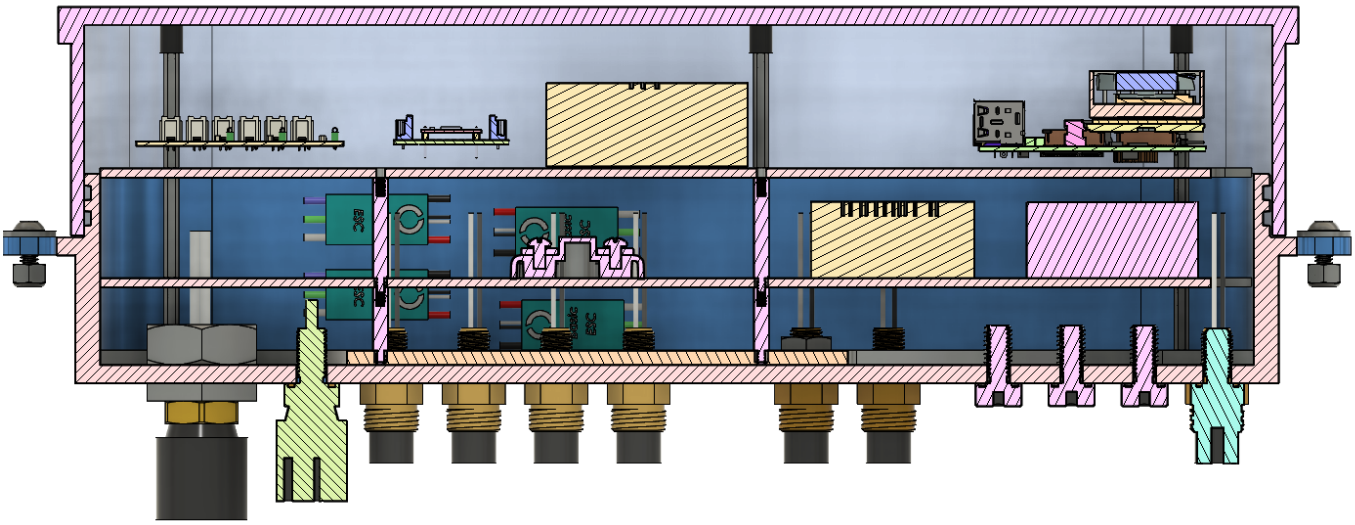

Fig. 4: computer box section view showing layered architecture.

2) Layered Architecture: The computer box is organized into three stacked layers (Fig. 4). The layers trade vertical space for added surface area. The placement of components within these layers is based on the expected frequency of access. The added difficulty of servicing the bottom layer is offset by the added ease of organization and cable management.

The bottom layer contains and separates all input cables from the underside of the sub. The cables are routed to the necessary layers through channels along the sides. The number and type of ports available aim to be flexible and not prescriptive. Unused input cables are tucked within the bottom layer and are retrievable from the second layer. Service to the bottom layer is only necessary in the case of failure of a wet-mate connector.

The middle layer holds the power system. Input from the battery is passed and split into two bus terminals, both nominally at 14.8 V. One bus provides power to the thrusters and the second bus provides power to the main computer, POE switch, thruster control board, and relay control board. The space cost of having two busses is offset by the advantage of controlling power to all thrusters with one relay. Additionally, we avoid the complexity of extra relays.

Power to thrusters is independent of power to all other components, allowing SubjuGator's computer system to operate normally when the vehicle is killed. The thruster bus has a higher current requirement, with large ring terminals. The second bus requires lower current with smaller and easy-to-crimp ferrule terminals. When a new component needs power, it is simple to remove the top layer and add the connection to the terminal.

The upper layer contains the components most frequently accessed: the main computer, POE switch, thruster control board, relay control board, and Hall-effect board. These components must be regularly accessed to flash firmware, view status lights, make new Ethernet connections, and debug hardware issues.

3) Ethernet Interconnect: The NavTube system communicates as a network device through a gigabit POE switch. The POE switch supports the addition of powered devices to be attached as part of the network. The devices act as discrete modules that can be accessed and tested without relying on other components.

4) Kill System: Power to the thrusters is controlled by a relay in series with the electronic speed controllers (ESCs). When killed, power to each thruster is cut. A relay control board implements logic to accept a kill signal from five sources and control up to six relays. The system currently accepts signals from two sources: the Hall effect board and the thruster control board. The Hall effect board detects the presence of the kill wand. When the wand is removed, the Hall effect board sends a signal to the relay control board to cut power to the thrusters. The thruster control board's microcontroller raises a kill signal when a heartbeat connection to the main computer is lost or when a kill request is sent from the main computer.

The design of the kill system facilitates robust testing of each kill source and each kill scenario. It also allows for easy modification and addition of new sources and different relay configurations.

C. Software

1) ROS2 Architecture: All of SubjuGator's systems are designed with ROS2 Jazzy. ROS2 (Robot Operating System 2) is a framework for handling complex concurrent systems. It also provides helpful libraries for localization, control, and frame of reference transformation capabilities. ROS2 enables thread-safe concurrency between threads (called nodes) through subscriber-publisher and server-client architectures.

2) Mission Planner: High-level autonomous behaviors are created using a custom mission planner system built in ROS2.

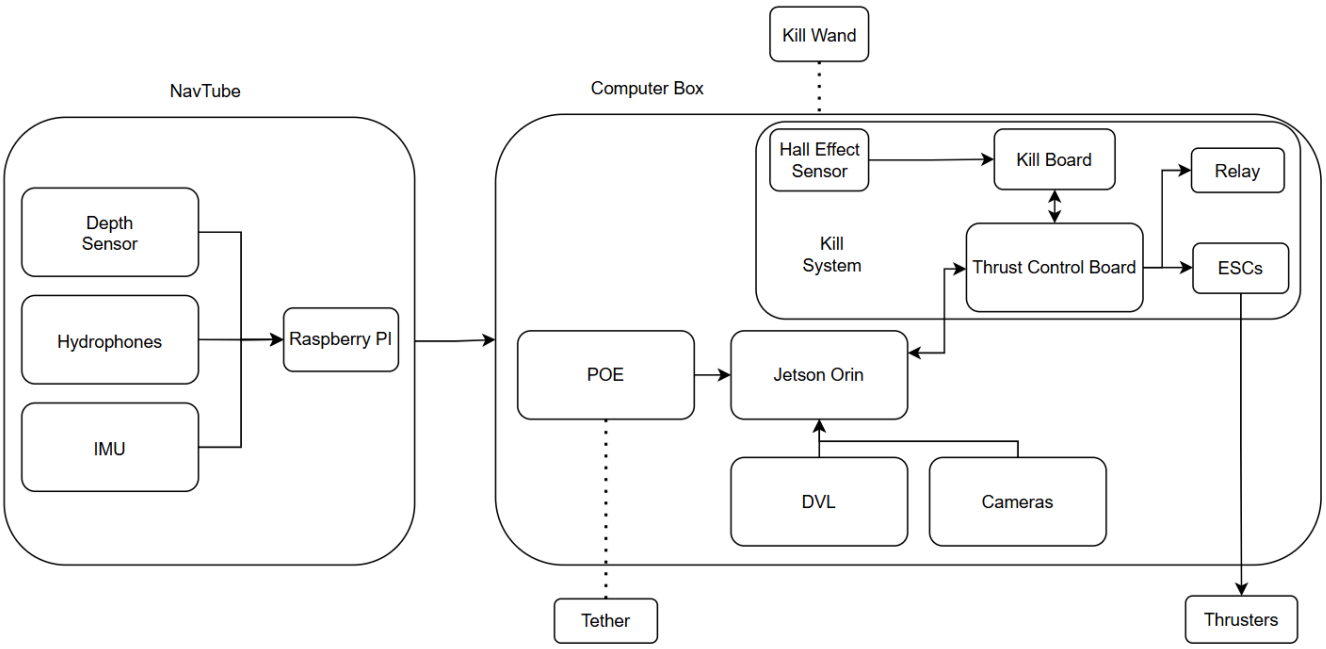

Fig. 5: Electrical Functional Block Diagram

The mission planner is implemented using ROS2 actions, a core communication type in ROS2 that facilitates the transfer of goals and feedback between nodes and servers. The system is composed of one central mission planner node, mission servers, and a configuration YAML file.

The mission planner node parses a mission from the YAML file and extracts tasks and relevant parameters. The node then sends goals along with any parameters as a ROS2 action goal to the corresponding mission server. The node then awaits feedback from the mission server to determine whether the goal is complete or if the task needs to be retried.

Mission servers are ROS2 action clients that receive a goal (a task and parameters) and execute predefined autonomous behaviors. The action clients can send feedback to the mission planner node, such as the distance remaining until reaching a target, whether a task is successfully completed, or reasons for not finishing a task (such as a sensor error, losing track of an object visually, or timing out).

A mission file is a YAML file that can be quickly modified, dictating tasks such as ”navigate around an object” or ”find an object with cameras.” Parameters such as task timeout, camera target, or orbit radius can be easily specified in the YAML file.

3) Localization: The DVL, IMU, and pressure sensor are fused into a position estimate using an extended Kalman Filter provided in the ROS2 robot localization package. This position estimate informs the PID controller.

Localization uses two separate reference frames called *odom* and *base_link*. The *odom* reference frame is fixed and defined by the vehicle’s starting orientation. *Base_link* is the vehicle’s reference, which rotates and translates with the vehicle.

4) Vision: SubjuGator used two HD Dell webcams for vision - one forward facing and one down facing. YOLOv11 nano provides computer vision for detecting objects and their position in the camera frame. This facilitates identifying tasks, moving through and around objects, and navigating using path markers.

5) Controller: The SubjuGator 9 vehicle uses a 6-degree-of-freedom PID controller implemented with the ROS2 controller toolbox. The gains for the controller were determined through extensive experimentation. The PID controller uses the error between the current pose and a goal pose to publish a command wrench.

6) Closed-Loop Control: Localization, the controller, and the thrusters come together to form closed-loop control (Fig. 6). The thruster manager decomposes a wrench into individual forces needed to be generated by the vehicle’s thrusters to achieve the desired wrench. This resulting pose is measured by the sensors to re-inform the PID controller, which will correct for any error.

7) Debugging with rqt and Rviz: ROS2 has a suite of debugging and visualization tools contained in the rqt and rviz packages. All numerical topics can be plotted using the rqt plot tool, allowing easier analysis of the vehicle’s performance. Rviz (Fig. 7) is used to debug issues with reference frames, transforms, and relative sensor positions.

8) Simulation: Simulation is used to test autonomous behavior without physically running the vehicle. The simulator is implemented using Gazebo, and the model is implemented to closely resemble the physics acting on the real vehicle. The primary benefit is the ease of testing mission planning algorithms that can be transferred to the real submarine.

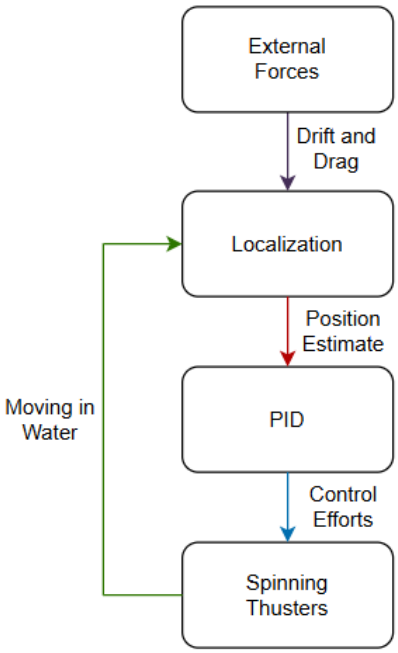

Fig. 6: Closed loop control

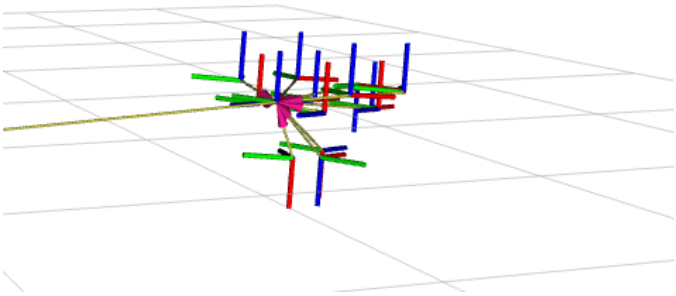

Fig. 7: The vehicle visualized in RViz

IV. Testing Strategy

A. Testing Schedule

We test SubjuGator in-water two or three times a week. We use GitHub issues to form a testing schedule (Fig. 8), which allocates time for localization, PID, and camera.

B. Pre-Flight

Pre-flight is a software procedure that mimics what the vehicle will do in the pool. Pre-flight spins the thrusters, checks sensors, pings the computers, and checks the camera connections. Pre-flight ensures the vehicle is in a working state before we take it to the pool.

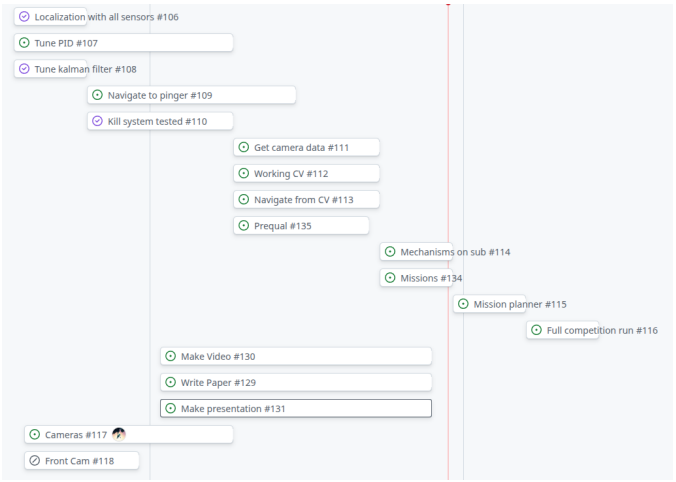

Fig. 8: Testing schedule for SubjuGator

C. At the Pool

Once the vehicle arrives at the pool, the team sets up and re-runs Pre-flight. A modified Pelican case (called the network box) connects the vehicle to poolside computers. The vehicle goes into the pool with a designated swimmer. Common swimmer tasks include: orienting the sub, monitoring battery voltages, acting as a reference point, and killing the sub.

D. Bagging Data

ROS2's bag functionality can record and playback data. All of the vehicle's sensor data, thruster efforts, and position estimates can be recorded. Any test that the vehicle fails at the pool is recorded and re-run in-lab. This means software changes to systems such as localization and the PID controller can be tested before the vehicle goes in the pool.

Acknowledgment

Team SubjuGator AUV, a project of the Machine Intelligence Laboratory (MIL) would like to extend our sincerest gratitude to everyone who has supported the team. We extend our thanks to the University of Florida's departments of Electrical & Computer Engineering and Mechanical & Aerospace Engineering for their continued support. We would also like to thank the CIMAR lab for providing the essential resources and facilities for our work. We are also deeply appreciative of our major industry sponsors: Sylphase, L3Harris Corporation, JD-Squared, and Texas Instruments. A special thank you to our alumni donors and advisers, Dr. Eric Schwartz, Dr. Carl Crane, and Matt Griessler (a MIL alumnus), for their invaluable guidance and mentorship throughout this project.