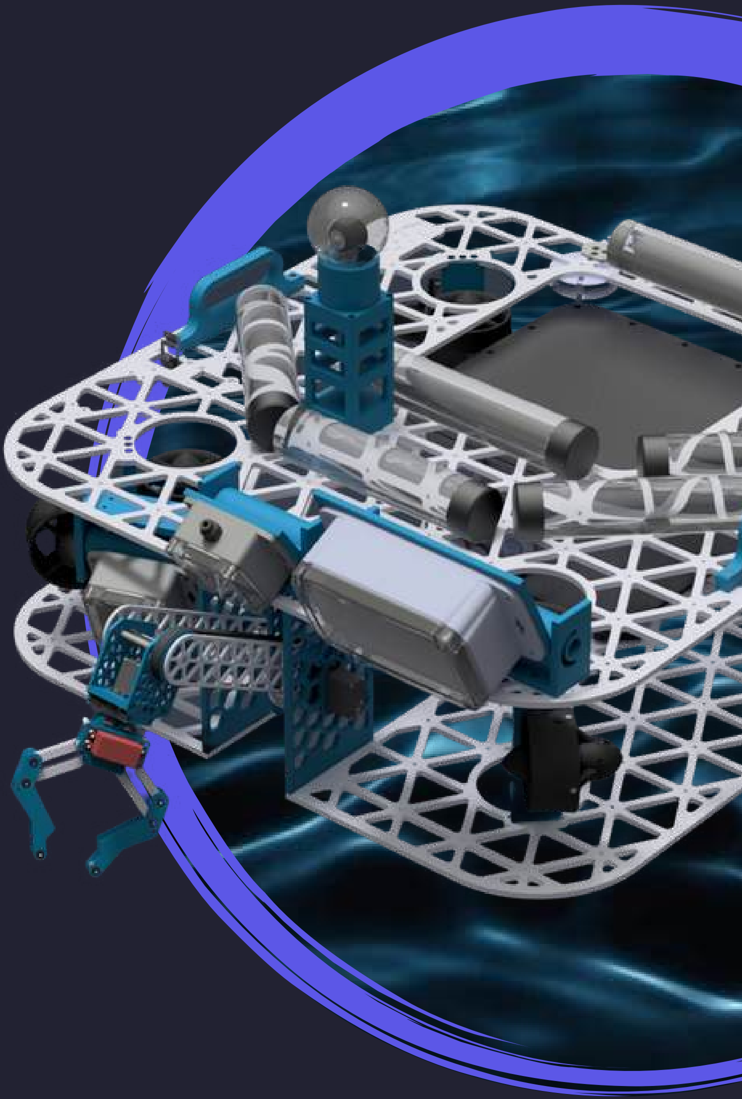
STONY BROOK
ROBOTICS TEAM

STONY BROOK UNIVERSITY
STONY BROOK, NY, USA

AETHER

TECHNICAL REPORT



TEAM ADVISOR

William Stewart

CHIEF EXECUTIVE OFFICER

Daniel Langer

MECHANICAL

Arash Khan *(Team Lead)*
Calvin Wu
Charlene Chen
Daniel Lu
Enkhlen Duinkharjav
Kazi Abthahi
Viven Jiang

ELECTRICAL

Daniel Kapriyelov *(Team Lead)*
Logan Diep *(Subteam Lead)*
Adam Roccanova
Andy Lam
Arhaam Hossain
Dylan Li
Eric Yang
Vincent Tan

SOFTWARE

Ruthvick Bandaru *(Team Lead)*
Mandy Tang *(Team Lead)*
Adam Feng *(Subteam Lead)*
Giovanni Terefi *(Subteam Lead)*
Brady Wynn
Derrick Ma
Karamat Hassan
Manasvi Bhattnagar

Table of Contents

Abstract

The Stony Brook Robotics Team presents Aether, a remotely operated vehicle designed to address the challenges outlined in the 2025 MATE ROV Competition. Developed through our team's collaborative efforts, Aether embodies our focus on robustness and ease of piloting. The chassis features an aluminum two-plate design with a modular hole pattern, providing structural integrity while allowing for design flexibility. Our 8-thruster configuration enables full 6 degrees of freedom movement, while our 3-DOF manipulator arm with concentric packaging maximizes operational efficiency. The electrical system emphasizes modularity and reliability through custom PCBs that minimize internal cabling, while our software implements PID stabilization and inverse kinematics for precise control. Extensive testing in various environments, from cardboard prototypes to pool trials, has refined Aether's capabilities to effectively complete mission tasks including environmental monitoring, marine renewable energy operations, and vertical profiling. This technical report documents our design process, manufacturing decisions, and system integration that have produced a vehicle optimized for the challenges of underwater exploration and intervention.

Figure 1: Team Picture

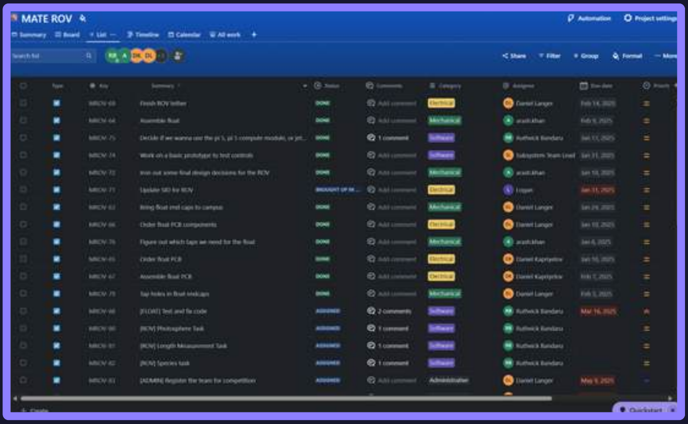
Teamwork

The Aether ROV project was executed under the banner of the Stony Brook Robotics Team (SBRT), a student-led organization at Stony Brook University that brings together aspiring engineers and roboticists across disciplines. SBRT's mission is to foster innovation through real-world robotics challenges, while building a community centered on mentorship, learning, and engineering excellence.

The team includes over 30 undergraduate students, from freshmen to seniors, organized into three primary technical subteams: Mechanical, Electrical, and Software. Each subteam was led by a Team Lead, with additional Subsystem Leads responsible for specific modules such as computer vision, GUI, propulsion, or PCB design. Overarching project coordination was led by a Project Manager, while logistics, funding, and procurement were handled by the Treasurer. External engagement, including sponsorships and industry relations, was overseen by a Partnerships Director.

Personnel and Organizational Roles:

- Project Manager: Oversaw the full project timeline, organized MATE-specific weekly meetings, created weekly action items, and ensured cross-team alignment.
- Team Leads (Mechanical, Electrical, Software): Managed day-to-day progress, hosted subteam meetings, broke down complex deliverables into actionable tasks, and coordinated integration across subsystems.
- Treasurer: Responsible for purchasing all components, managing the budget, submitting reimbursement forms, and maintaining inventory tracking.
- Partnership Director: Secured funding and in-kind sponsorships from industry partners such as Blue Robotics, Hitec, Sigma AeroSpace, Public Metals, PCBWay, and AVS Signage.



Type	Key	Summary	Status	Comments	Category	Assignee	Due date	Priority
✓	MROV-68	Finish ROV tether	DONE	Add comment	Electrical	Daniel Langer	Feb 14, 2025	=
✓	MROV-64	Assemble float	DONE	Add comment	Mechanical	arashkhan	Feb 9, 2025	=
✓	MROV-75	Decide if we wanna use the pi 5, pi 5 compute module, or jet...	DONE	1 comment	Software	Ruthwick Bandaru	Jan 17, 2025	=
✓	MROV-74	Work on a basic prototype to test controls	DONE	Add comment	Software	Subsystem Team Lead	Jan 31, 2025	=
✓	MROV-72	Iron out some final design decisions for the ROV	DONE	Add comment	Mechanical	arashkhan	Jan 10, 2025	=
✓	MROV-71	Update SID for ROV	BROUGHT UP IN ...	Add comment	Electrical	Logan	Jan 31, 2025	=
✓	MROV-63	Bring float end caps to campus	DONE	Add comment	Mechanical	Daniel Langer	Jan 29, 2025	=
✓	MROV-66	Order float PCB components	DONE	Add comment	Electrical	Daniel Langer	Jan 10, 2025	=
✓	MROV-76	Figure out which taps we need for the float	DONE	Add comment	Mechanical	arashkhan	Jan 6, 2025	=
✓	MROV-65	Order float PCB	DONE	Add comment	Electrical	Daniel Kapryelov	Jan 10, 2025	=
✓	MROV-67	Assemble float PCB	DONE	Add comment	Electrical	Daniel Kapryelov	Feb 7, 2025	=
✓	MROV-79	Tap holes in float endcaps	DONE	Add comment	Mechanical	Daniel Langer	Feb 3, 2025	=
✓	MROV-[illegible]	[FLOAT] Test and fix code	ASSIGNED	2 comments	Software	Ruthwick Bandaru	Mar 16, 2025	^
✓	MROV-80	[ROV] Photosphere Task	ASSIGNED	1 comment	Software	Ruthwick Bandaru		=
✓	MROV-81	[ROV] Length Measurement Task	ASSIGNED	1 comment	Software	Ruthwick Bandaru		=
✓	MROV-82	[ROV] Species task	ASSIGNED	1 comment	Software	Ruthwick Bandaru		=
✓	MROV-83	[ADMIN] Register the team for competition	ASSIGNED	Add comment	Administration	Daniel Langer	May 9, 2025	—

Figure 2: Sample Jira Task Board

All team roles were clearly defined at the start of the semester and documented in our internal team charter, which was shared on Discord.

Scheduling and Workflow Management

To manage task distribution and maintain visibility into progress, we used **Jira**, a robust project management tool that allowed us to:

- **Create and assign tasks** to individuals with due dates and priorities.
- **Tag tasks** by category (Mechanical, Electrical, Software, Admin).
- **Track task completion** with real-time status updates (e.g., TODO, IN PROGRESS, DONE).
- **Visualize the build timeline** using Jira's Gantt chart and calendar views.

This scheduling system allowed the team to break down the entire build season into manageable sprints. For example, the **mechanical team** followed a milestone-driven approach, starting with CAD reviews, followed by part manufacturing, then build sessions. Similarly, the **electrical team** used phased deadlines for schematic design, PCB layout, ordering, and soldering. The **software team** followed an agile structure, with weekly integration check-ins, code reviews, and subteam syncs for CV, control, and GUI.

Teamwork

Each week, a MATE-specific coordination meeting was hosted by the project manager. These meetings reviewed the past week's progress, resolved blockers, and realigned upcoming deliverables with the master timeline. Meeting minutes and action items were documented and shared to keep the full team on track.

Resources, Procedures, and Operational Problem Solving

We developed several internal protocols to ensure consistent performance across the build season:

- **Communication:** All communications, emergency and routine, occurred on our structured Discord server, with separate channels for each subteam and announcements. This ensured clear documentation, fast updates, and strong team coordination.
- **Version Control:** The Software Team Lead maintained the official MROV GitHub repository, with guidelines on branching, committing, pull requests, and code reviews. Subsystem leads met weekly to integrate and test their modules with the main repo.
- **Risk Mitigation:** Early in the project, we identified critical path dependencies (e.g., PCB arrival, waterproofing, integration delays) and built time buffers into the schedule. Redundancy was also built into hardware sourcing to avoid single points of failure.

Resource Allocation: The Treasurer worked closely with all subteams to ensure that items were ordered well in advance. A shared Google Sheet tracked all purchases, arrival times, vendor info, and invoice status. The team kept receipts organized for university reimbursements.

Collaboration Across Subteams: As the robot moved toward integration, cross-functional meetings became more frequent to ensure that subsystems (e.g., software control logic and electrical pin mapping) were aligned. These meetings were key in avoiding integration bottlenecks.

Time Investment and Team Culture
Across the semester, the team invested over 2,000 hours in design, testing, fabrication, soldering, debugging, and documentation.

This included:

- Subteam work sessions 2–3 times per week.
- Nightly build marathons during peak integration periods.
- Testing sessions on campus and at pool trials.

Team members were empowered to take ownership of tasks and encouraged to collaborate openly across subteams. Mistakes were treated as learning opportunities, and internal documentation was created throughout the process to ensure sustainability and knowledge transfer to future cohorts.

We practiced disciplined, real-world project management from day one—balancing structure and flexibility, distributing responsibilities effectively, leveraging tools like Jira and GitHub, and managing time, risk, and resources with professional rigor. This framework allowed us to meet MATE's objectives while building a powerful and capable underwater vehicle as a unified team.

Design Rationale

VEHICLE OVERVIEW

For our first year competing in MATE ROV, our goal was to create a robot that was robust and easy to pilot. We kept these two attributes in mind throughout our entire design process.

To maximize our robot's robustness, we created our mechanisms to have multiple degrees of freedom. First, our chassis has eight thrusters and 6 degrees of freedom of motion. This allows our robot to control is position in all directions and angles, which is crucial for having a stable robot.

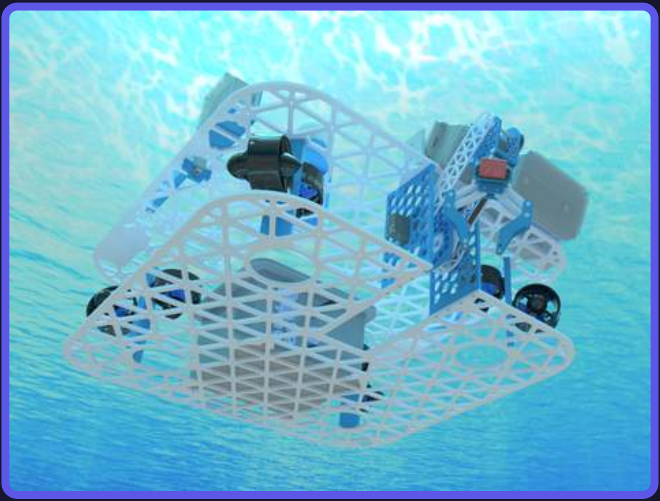


Figure 3: Rendering of the ROV Underwater

For our the chassis' structure, we decided to go for a two plate design out of aluminum to make it a rigid base for the rest of the robot. These plates are designed with a modular hole pattern to allow us to have more long term design options. With this, we can make adjustments to our design without re-machining the robot.

In addition to that, our arm consists of 3 degrees of freedom and a claw. This gives a wide range of motion to the arm which is able to perform most of the manipulation tasks without having to rotate the entire ROV. This gives us a lot of options for how we can approach each task, as we can pick the right arm orientation for the job.

We also optimized our robot to be easy to control and use. The first way we did this was through having a roughly symmetrical design. This made the robot relatively balanced and made for smoother operation of the ROV. In addition to that, we further improved the balancing of the robot by using PID controllers in software to correct for any drift due to gravity, inertia, etc. This means that the pilot can focus on scoring for tasks instead of wasting time correcting micro-imbalances in the ROV.

In addition to the PID controllers, another software feature we used to improve the piloting experience was to have a remappable controller layout on our GUI. This means that pilots can adjust the control settings to their preferences on the fly in case a specific binding isn't comfortable. This also helps with robustness in scenarios where there are controller malfunctions.

Another ease of use feature we added was the slot on the top plate to access our electronics box. This allows us to easily check the electronics to troubleshoot problems on the ROV.

By focusing on robustness and ease of use, we were able to design a versatile robot that is able to perform reliably. This allows us to perform our best on each run, given the appropriate driver practice.

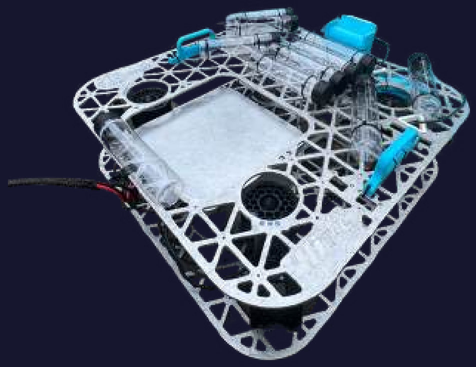


Figure 4: Real Life Picture of ROV

Design Rationale

MECHANICAL DESIGN: CHASSIS

Our main chassis structure consists of two base plates held together by four bottom tapped rectangular beams and the two arm mounting plates. It consists of eight T200 thrusters from Blue Robotics for underwater propulsion.

Thruster Placement:

For our ROV, we decided that it would be critical to make it fully holonomic with 6 degree of freedom motion. To simplify the kinematics, we separated the thrusters into two sets, the x-y planar thrusters and the vertical thrusters which control the propulsion of the ROV in those given directions. For the planar thrusters, we placed them at 45 degree angles such that a specific linear combination of them will allow it to move in any given direction in free space. The thruster orientations were also are mirrored as the T200 thrusters we used are not perfectly bidirectional, so this helped make the motion more uniform along all directions in the x-y plane. We decided to also use four thrusters for the vertical thruster set to make it more balanced and stable. We also decided to point all of the thrusters upwards as the ROV is slightly negatively buoyant, so it needs a little bit more power to ascend vs descend.

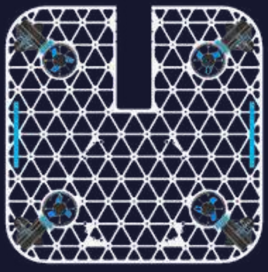


Figure 5: Thruster Layout Diagram

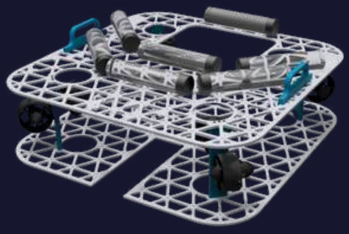


Figure 6: CAD Rendering of Chassis

Balancing and Buoyancy:

The first thing we did to help with balance was design the robot roughly symmetrically so that forces would cancel out. Given that our robot is essentially all metal, we also pocketed our plates to decrease the weight of the ROV, which increased the buoyancy overall. This balanced out the relatively higher density aluminum used to create most of the ROV. Buoyancy wise, our goal was to make the ROV slightly negatively buoyant so it could go into the water but was still roughly stable to the point where software could correct for the difference. Our electronics box was a major aspect of maintaining buoyancy as it is a large box, mainly filled with air. After finishing the design, we used large plastic water bottles and attached them to ROV to finetune the buoyancy and balance to be exactly as we desire.

Pocketing:

We decided to pocket our top and bottom chassis plates with an isogrid pattern. The first benefit of pocketing is that it reduces weight of the robot. This, in turn, increases the robot's buoyancy and makes the robot faster. In addition to weight reduction, pocketing helps with water circulation to make it more hydrodynamic to minimize drag. We decided to go with an isogrid pattern for it's very high strength to weight ratio.

Modular Hole Pattern:

We also decided to include a modular hole pattern placed at the intersection of the isogrid triangles from the pocketing. This allows us to make adjustments after assembling the robot, along with making it more reusable for future years.

Design Rationale

MECHANICAL DESIGN: ARM

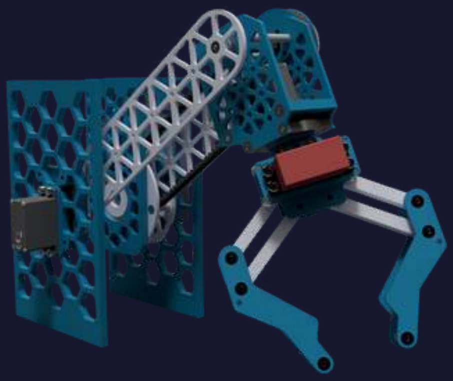


Figure 7: CAD Rendering of Arm

Claw Design:

For our claw, we wanted it to be adaptable to a large range of tasks. We chose this 4 bar claw design because it is able to hold both large and small objects well. This is due to the fact that the 4 bar causes it to close linearly, rather than angularly. While an angular claw is really only optimal for picking up one specific size of object, as the jaws will only be parallel at one point, the linear nature of the 4 bar claw design allows the jaws to always be parallel to each other. The curves in the geometry of the claw are also designed to provide at least four points of contact on the object it's holding

Section View Color Key:
Green: Servo Spline Mounting Hubs
Red: Pulley for Second Stage of Arm
Yellow: Bearings

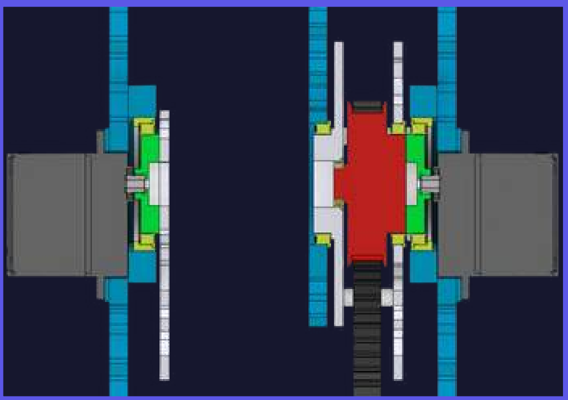


Figure 8: Diagram of Packaging at the Base of the Arm

Arm Structure:

Our arm is composed of 3 degrees of freedom with a claw at the end effector. We decided to make one multipurpose arm that can perform all of the payload tasks we are aiming to do. A multi DoF arm, like ours, allows for us to do a majority of the tasks with one arm mechanism. This reduces the amount we need to resurface, saving valuable time. It is actuated with servos to have precise control over each degree of freedom.

The first stage of the arm is directly driven by the servo to rotate the whole arm. We decided to use a direct drive as the servos that were donated to us already met the torque requirements for our arm. Adding a gearbox or other intermediate power transmission steps would've added unnecessary complexity and points of failure to our design. To reduce the load on the servos, we instead supported them with bearings.

The second stage of the arm is a virtual 4 bar, connected at the end of the first stage. A virtual 4 bar works similarly to a normal 4 bar with parallel bars, except here we replace the bars with a belt. The top and bottom side of the belt represent two opposite sides of the parallelogram, while the two pulleys represent the other two opposing sides. This makes it so that the second stage of the arm always maintains it's orientation, regardless of how the first stage moves, which significantly reduces the programming complexity.

The third stage, the wrist, concentrically rotates the claw. This allows us to easily rotate objects about the center of our claw. This also allows us to reorient the claw to find the optimal grabbing position.

Concentric Packaging:

We designed the actuation of the first two arm stages to be along the same axis. Although it increases the complexity of the packaging, it serves two purposes. First, the base of the virtual 4 bar needs to be concentric to base of the first arm stage for the belt to be able to rotate with it. Secondly, we're able to save weight by not mounting the servo on the end of the arm. This reduces the stress on the first stage servo.

Design Rationale

MECHANICAL DESIGN: ELECTRONICS BOX

Design Motivation:

Our electronics has seen drastic changes throughout the year. At first, we tried using a plastic commercial one, but we ran into an issue where the high current drawing components like the buck converters were overheating. To solve this problem, we decided to use a metal enclosure instead of a plastic one to allow for better heat transfer out of the box and into the water. That being said, there were no commercially available enclosure that met our requirement within our budget so we decided to design and manufacture our own.

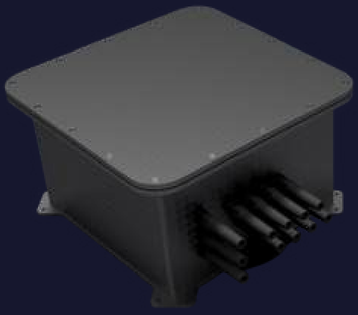


Figure 9: CAD Rendering of EBox Closed

General Structure:

The main body of the electronics box is composed of 4 vertical sections welded to a base plate which contained mounting holes to the chassis. To open and close the box, we created a lid that interfaces with an O-ring seal to keep the box watertight. Holes were strategically drilled on two sides of the box to allow convenient wire routing inside of the enclosure.

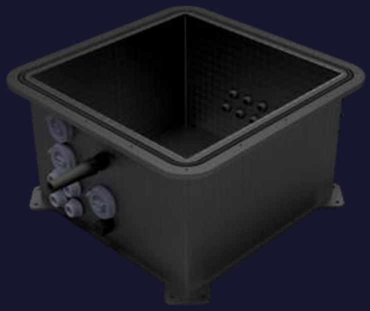


Figure 10: CAD Rendering of EBox Opened

Sealing:

We decided to seal the lid of the electronics box with an O-Ring seal, more specifically a face seal. We chose a face seal as it's generally easier to manufacture over other types of O-Ring seals. Face seals are also relatively simple to open and close, giving us easy access to our electronics.
For sealing the corners, the welds were properly made and tested. Therefore, we could conclude that they were sufficiently waterproof.

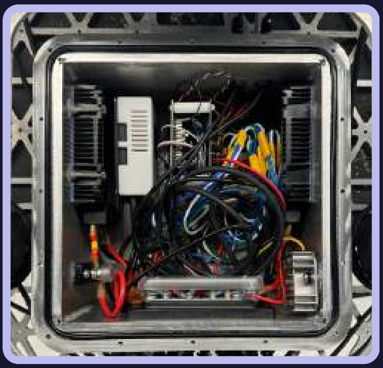


Figure 11: Wire Routing Inside EBox

Component Placement:

The enclosure was designed to be as small as possible, which still being able to all of the components and wire. This meant there was a lot of planning with the component placement. We placed the buck converters on opposing walls to evenly dissipate the heat. The Jetson and PCBs have a more central location as they have to interface with a lot of different components. The ESCs were also mounted on the wall with the cable penetrators for the thrusters to reduce the amount of wire routing needed.

Design Rationale

ELECTRICAL DESIGN: SYSTEMS DESIGN

The electrical system of our ROV emphasizes modularity, reliability, and accessibility, with a design that minimizes internal cabling through tightly integrated subsystems. At the center of the control architecture is the NVIDIA Jetson Orin Nano, which serves as the onboard processor. It features a 6-core ARM Cortex-A78AE CPU and supports four USB 3.2 ports, all of which are used to connect a 360-degree camera, two USB cameras, and a ZED 2i stereo vision camera for the photosphere task. The Jetson also manages I²C communication with the PCA9685 PWM controller and two BNO055 IMUs, along with analog input from the SOS leak sensor. We access the Jetson remotely via SSH, allowing us to run control scripts, manage data streams, and monitor sensor output in real time. Its processing speed, extensive I/O, and ability to directly handle sensor and actuator communication make it well-suited for embedded control without requiring additional microcontrollers or topside processing.

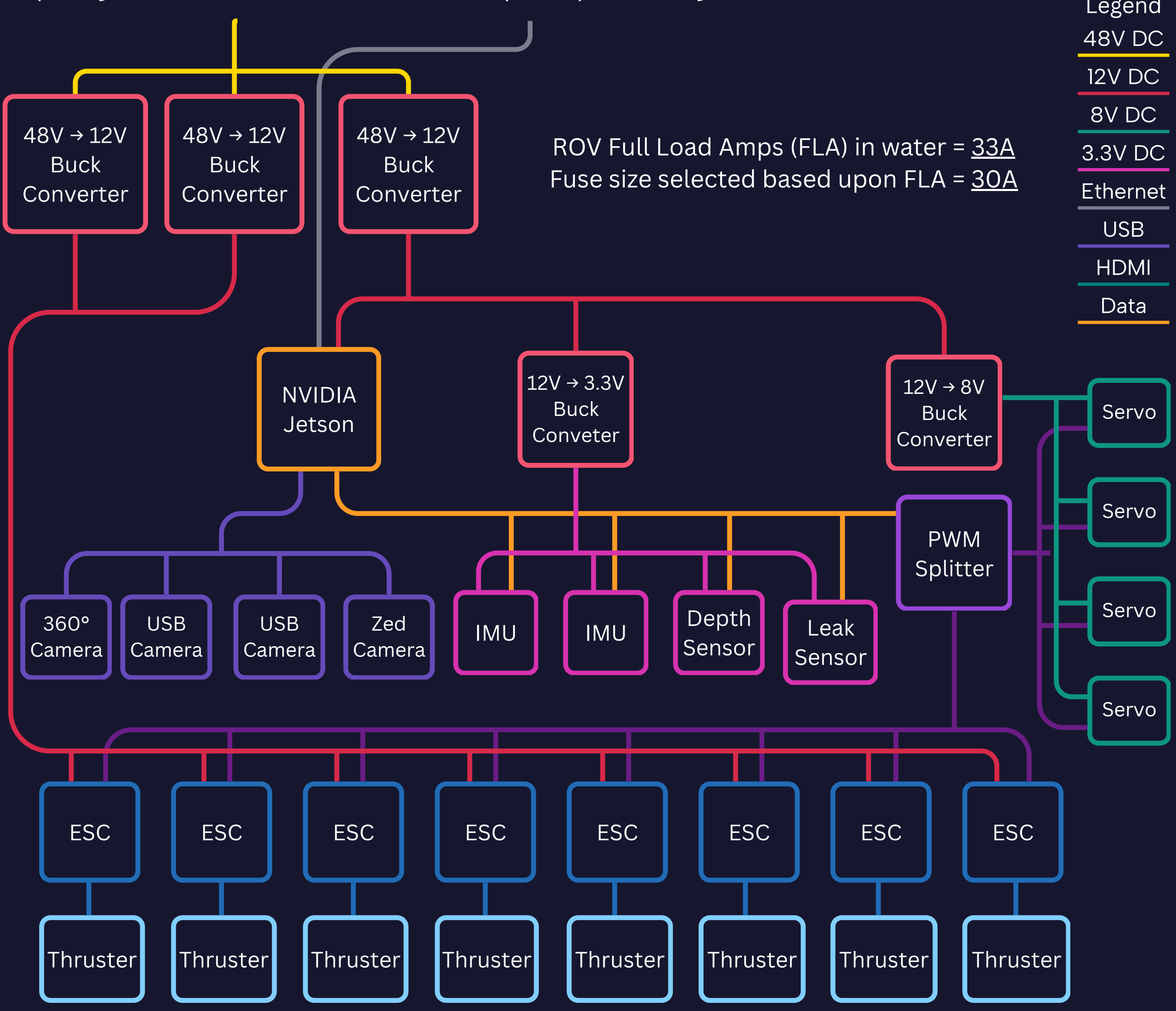


Figure 12: SID for electrical components inside ROV

Design Rationale

ELECTRICAL DESIGN: POWER DESIGN

The vehicle receives a single 48V power rail from the surface through the tether. This input is protected by a 30A fuse, which was selected based on a calculated full-load current of 33A while submerged. Two high-efficiency buck converters, each rated for 100A, are used to step down this 48V input to 12V. This 12V intermediate rail serves as the source for all downstream converters and is routed directly into a custom-designed six-layer power distribution PCB. This board further regulates the 12V input down to 8V and 5V to power different subsystems, and was developed in Altium Designer to meet the high current and thermal demands of our architecture. The 8V rail is regulated using the SIC431CED-T1-GE3 synchronous buck converter and is designed to deliver up to 24A to four high-torque brushless servos. These servos operate the ROV's arm/claw mechanism and draw up to 3A peak current each. The 5V rail is regulated using the TPS564255DRLR buck converter, providing up to 4A to the logic and sensor systems, including the PWM driver, two IMUs, the depth sensor, and the leak sensor.

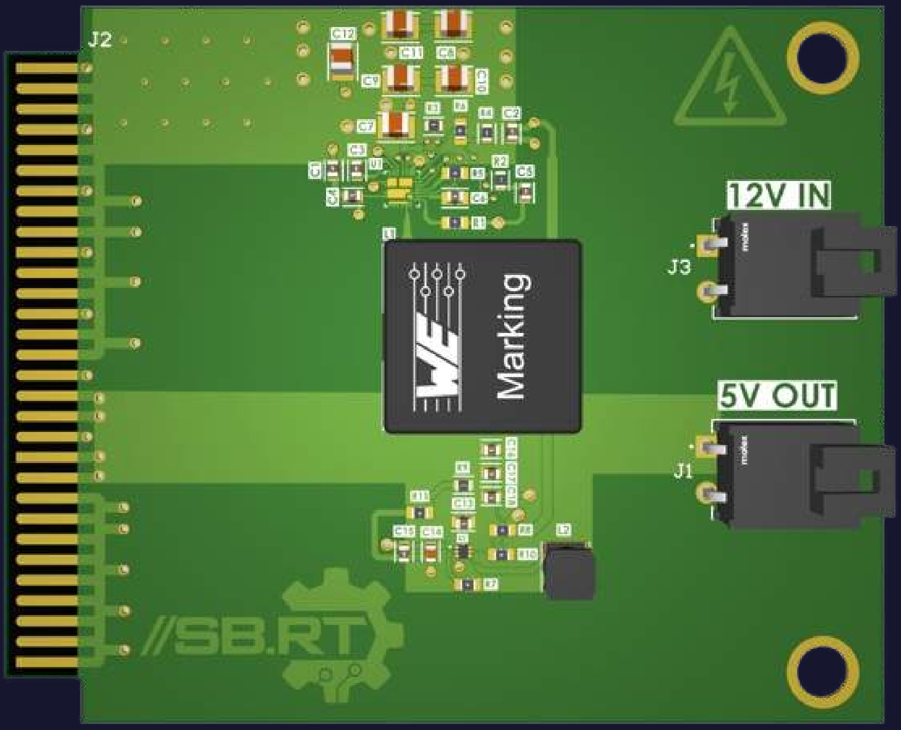


Figure 13: Render of the Power PCB

To handle thermal dissipation, the power board includes heavy 2 oz copper on all layers, with dedicated planes for 12V and 8V power distribution. Stitching vias were incorporated on the high current polygon pours to improve thermal conductivity and reduce localized heating. These decisions were made after evaluating the inefficiencies encountered when relying on off-the-shelf modules, which often lacked adequate consideration to reliably handle high currents under sustained loads.

Design Rationale

ELECTRICAL DESIGN: LOGIC DESIGN

The control logic is consolidated onto a four-layer logic PCB, also designed in Altium. This board manages all digital control signals and sensor inputs, and is structured with a Signal–GND–GND–Signal layer stack to ensure signal integrity and reduce EMI. The PCA9685 PWM driver chip was chosen based on previous prototyping experience using a breakout board. It allows up to 16 PWM channels to be driven simultaneously from a single I²C connection, enabling unified control of all eight T200 thrusters and the four servos. The IMUs, depth sensor, and PWM driver all share the same I²C bus, while the leak sensor is read through an analog pin on the Jetson.

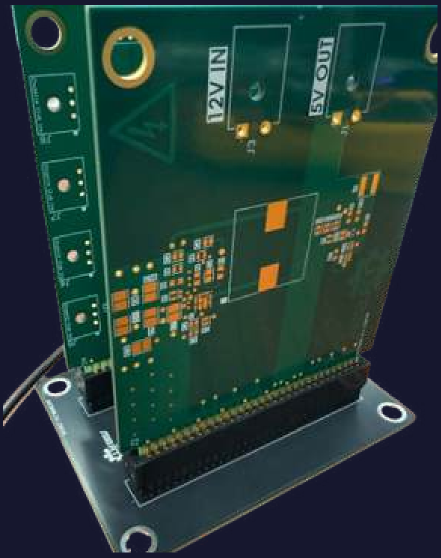


Figure 14: Logic and Power PCBs assembled with the backplane PCB

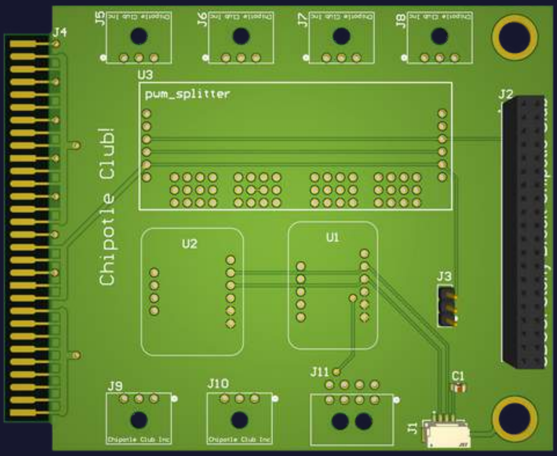


Figure 15: Render of the Logic PCB

Accurate sensor data is essential for control and fault detection. Two BNO055 IMUs provide redundant orientation data, fused using a Kalman filter on the Jetson. The BAR02 depth sensor communicates over I²C, while the SOS leak sensor is read through the Jetson's ADC. All signal and power connections use Molex Micro-Fit connectors, chosen for their compact size, current capacity, and secure fit—well-suited for the tight constraints of an underwater system. The logic and power boards connect to a backplane via gold-finger edge connectors with a 30-degree chamfer, eliminating point-to-point wiring between PCBs. This modular setup simplifies assembly, improves reliability, and allows for quick removal and replacement. Custom 3D-printed standoffs secure the boards while keeping them easily accessible for field servicing.

Switching from breakout boards to custom logic and power PCBs was a key improvement in this year's design. It significantly reduced internal cabling, resulting in a cleaner layout that's easier to troubleshoot and expand. Inside the electronics enclosure, the only cables are the 48V power input to the buck converters, 12V lines to the power board, PWM and power lines to the thrusters and servos, a single Ethernet connection to the Jetson, and cables from the cameras to the Jetson IO.

Design Rationale

SOFTWARE DESIGN: ROV CONTROL SYSTEMS DIAGRAM

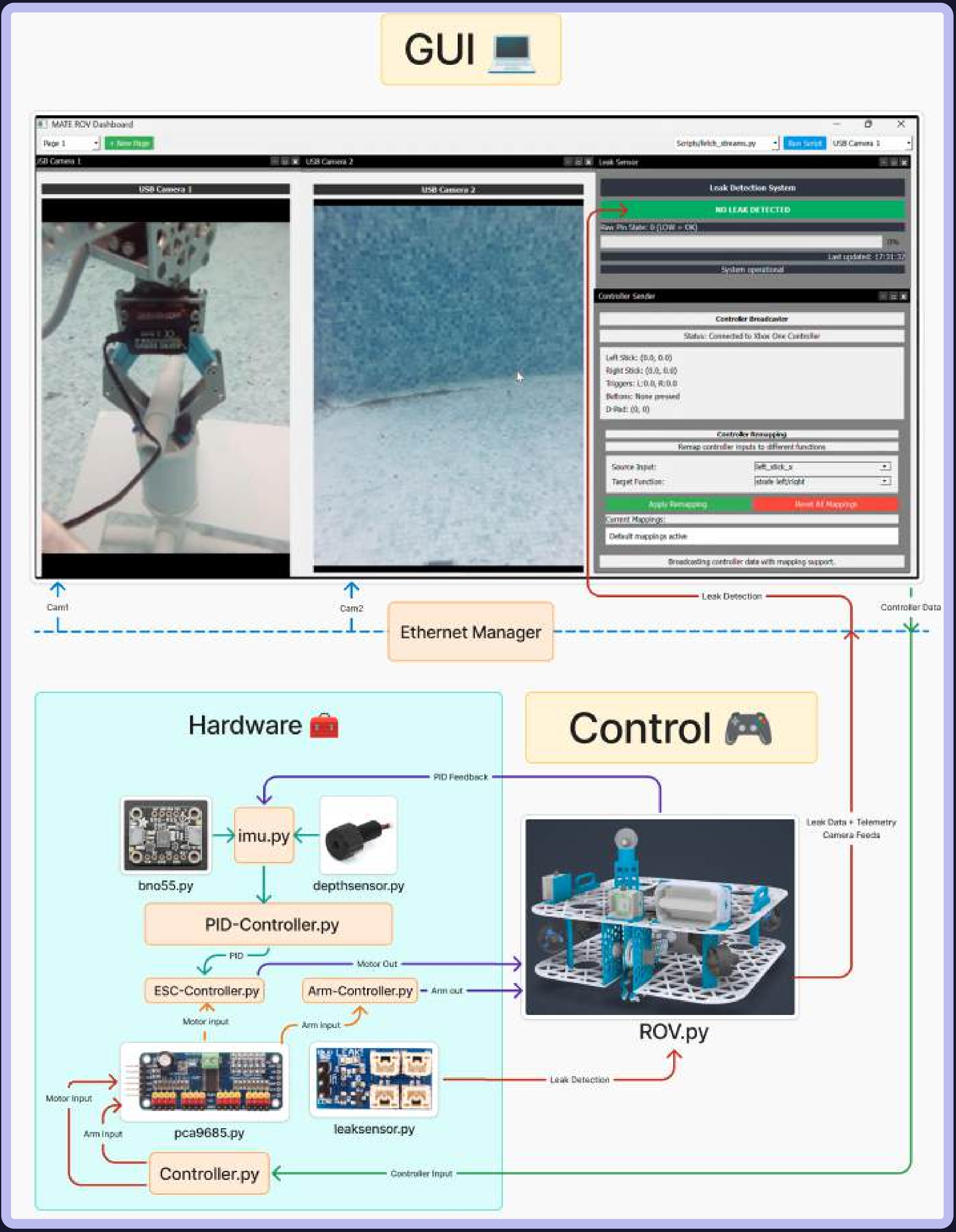


Figure 16: Software Control Systems Diagram

Design Rationale

SOFTWARE DESIGN: CONTROLS DESIGN

The control system of the ROV is designed to provide high-precision, robust, and adaptive control for complex underwater missions. It leverages real-time feedback loops, deterministic command execution, and fault-tolerant design principles to ensure the vehicle can operate in dynamic and often unpredictable aquatic environments.

Architecture:

At the core of the ROV's control architecture is the PID-Controller.py module, which performs multi-axis stabilization using fused sensor data. Orientation data from the BNO055 Inertial Measurement Unit (imu.py) and depth measurements from the MS5837 pressure sensor (depthsensor.py) are continuously sampled and filtered. These inputs feed into a closed-loop PID control system that dynamically adjusts thrust and attitude to maintain our desired positions. The modular nature of the architecture facilitates isolation of critical subsystems, simplifying debugging, maintenance, and future upgrades.

Thruster and Servo Management:

The vehicle uses eight thrusters to achieve full 6-DOF control. Thruster commands, computed from PID and pilot input, are translated to PWM signals via a PCA9685 driver.

Manipulation is handled by Hitec waterproof servos, including continuous rotation models for the claw. These are managed through Arm-Controller.py, with preset positions:

- Stowed: Safe travel position
- Fully Out: Forward extended
- Fully Down: Vertical
- Out Down: Extended downward

These presets support efficient task transitions.

User Input Integration:

Operator control is enabled through a USB gamepad, with input data parsed and normalized using Controller.py. The mapping is fully configurable via JSON, supporting dynamic reassignments and multiple control profiles. Inputs are sanitized for dead zones and merged with PID outputs to ensure stable and intuitive response.

Safety and Reliability:

System safety is reinforced through both hardware and software mechanisms. The Blue Robotics SOS Leak Sensor is integrated to detect internal water intrusion. Upon detection, the system can autonomously stow the arm, cease propulsion, and notify the surface operator. The sensor provides a critical safeguard for early-stage intervention.

On the software side, the PID system incorporates saturation bounds, integrator windup protection, and low-pass filtering for derivative calculations. Each control thread runs in an exception-handling wrapper, allowing localized failures to be caught and logged without affecting global control. Regular telemetry streams—including sensor health, IMU calibration status, and motor duty cycles—provide operators with actionable data for mission monitoring.

Communication:

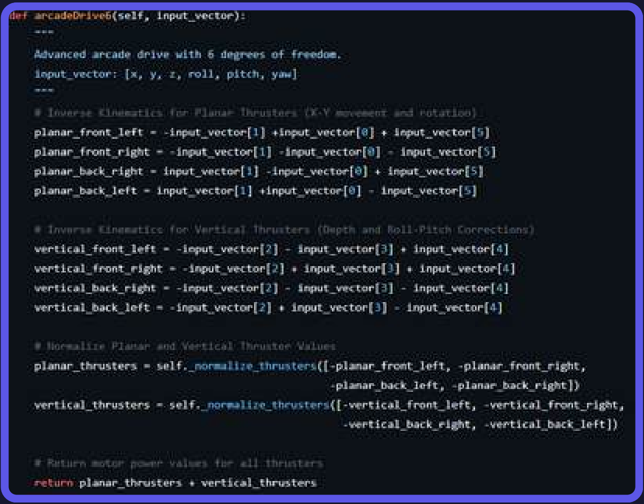
Network communication uses a dual-protocol model. UDP is employed for latency-sensitive control signals and telemetry data, while TCP is reserved for video feeds and camera control. This separation ensures that transient packet loss does not impact core vehicle control, and that video integrity is maintained for situational awareness.

Design Rationale

SOFTWARE DESIGN: CONTROL LOGIC

Inverse Kinematics:

Our inverse kinematics system translates intuitive directional commands into precise thruster instructions through a streamlined mathematical approach. The system accepts a six-component vector (x, y, z, roll, pitch, yaw) and automatically calculates power levels for each thruster on a -1 to 1 scale. This creates an abstraction layer that handles the complex coordination between planar thrusters (for x-y movement and yaw) and vertical thrusters (for depth control and pitch/roll stability), enabling us to express calculations in terms of the ROV's desired motion rather than having to manually determine individual thruster outputs.



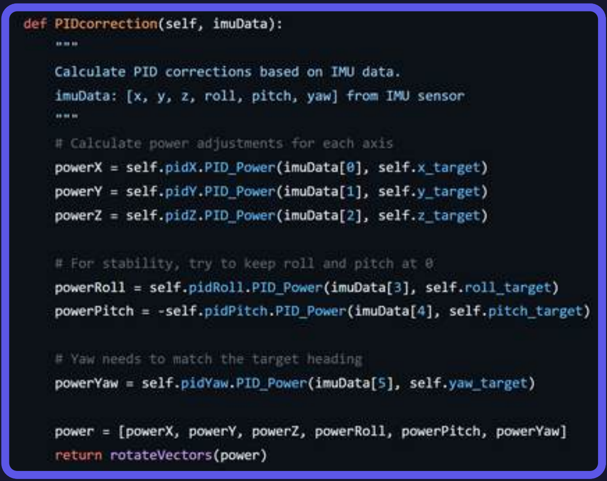
```python
def arcadeDrive6(self, input_vector):
    """
    Advanced arcade drive with 6 degrees of freedom.
    input_vector: [x, y, z, roll, pitch, yaw]
    """
    # Inverse Kinematics for Planar Thrusters (X-Y movement and rotation)
    planar_front_left = -input_vector[1] +input_vector[0] + input_vector[5]
    planar_front_right = -input_vector[1] -input_vector[0] - input_vector[5]
    planar_back_right = input_vector[1] -input_vector[0] + input_vector[5]
    planar_back_left = input_vector[1] +input_vector[0] - input_vector[5]

    # Inverse Kinematics for Vertical Thrusters (Depth and Roll-Pitch Corrections)
    vertical_front_left = -input_vector[2] - input_vector[3] + input_vector[4]
    vertical_front_right = -input_vector[2] + input_vector[3] + input_vector[4]
    vertical_back_right = -input_vector[2] - input_vector[3] - input_vector[4]
    vertical_back_left = -input_vector[2] + input_vector[3] - input_vector[4]

    # Normalize Planar and Vertical Thruster Values
    planar_thrusters = self._normalize_thrusters([-planar_front_left, -planar_front_right,
                                                  -planar_back_left, -planar_back_right])
    vertical_thrusters = self._normalize_thrusters([-vertical_front_left, -vertical_front_right,
                                                    -vertical_back_right, -vertical_back_left])

    # Return motor power values for all thrusters
    return planar_thrusters + vertical_thrusters
```

Figure 18: Inverse kinematics Implementation



```python
def PIDcorrection(self, imuData):
    """
    Calculate PID corrections based on IMU data.
    imuData: [x, y, z, roll, pitch, yaw] from IMU sensor
    """
    # Calculate power adjustments for each axis
    powerX = self.pidX.PID_Power(imuData[0], self.x_target)
    powerY = self.pidY.PID_Power(imuData[1], self.y_target)
    powerZ = self.pidZ.PID_Power(imuData[2], self.z_target)

    # For stability, try to keep roll and pitch at 0
    powerRoll = self.pidRoll.PID_Power(imuData[3], self.roll_target)
    powerPitch = -self.pidPitch.PID_Power(imuData[4], self.pitch_target)

    # Yaw needs to match the target heading
    powerYaw = self.pidYaw.PID_Power(imuData[5], self.yaw_target)

    power = [powerX, powerY, powerZ, powerRoll, powerPitch, powerYaw]
    return rotateVectors(power)
```

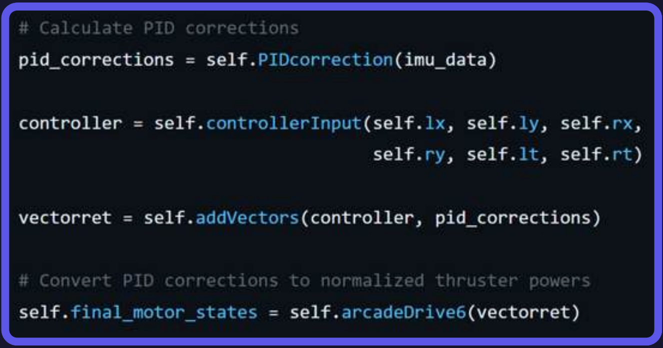
Figure 17: PID stabilization implementation

Integration into Control Loop:

Our control system combines automatic stabilization with pilot commands through a three-step process. First, we calculate PID corrections based on IMU data to maintain vehicle stability. Second, we process controller inputs (joysticks and triggers) into a vector representing desired movement in 3D space. Last, we combine these two vectors and feed them into our inverse kinematics function, which calculates the precise thruster power needed for each motor. This integrated approach ensures the ROV responds accurately to pilot commands while automatically maintaining orientation and depth, creating a responsive yet stable underwater vehicle.

PID Stabilization:

Our PID stabilization system employs proportional-integral-derivative controllers to automatically maintain the ROV's orientation and depth. The roll controller keeps the vehicle level at 0 degrees, while pitch and z-direction controllers feature adaptive targeting that updates when a position is held intentionally. For vertical movement, we incorporate a 3D rotation matrix to account for how the ROV's orientation affects movement relative to global coordinates. This system effectively eliminates drift and reduces the need for constant manual corrections, making the vehicle significantly easier to operate during mission tasks.



```python
# Calculate PID corrections
pid_corrections = self.PIDcorrection(imu_data)

controller = self.controllerInput(self.lx, self.ly, self.rx,
                                  self.ry, self.lt, self.rt)

vectorret = self.addVectors(controller, pid_corrections)

# Convert PID corrections to normalized thruster powers
self.final_motor_states = self.arcadeDrive6(vectorret)
```

Figure 19: Implementation of Control Logic in Main Loop

Design Rationale

SOFTWARE DESIGN: GUI AND COMPUTER VISION

Computer Vision:

This report outlines the computer vision capabilities onboard the ROV, including real-time camera streaming via GStreamer, 3D length measurement using ZED2 stereo pointclouds, and 360-degree visualization using a dual-fisheye Insta360 Air camera. Each section includes practical code snippets and workflow explanations.

Camera Streaming with GStreamer:

The ROV streams video from multiple onboard cameras to the surface station using GStreamer pipelines, enabling efficient, low-latency video transmission over UDP. GStreamer Pipeline Example (ZED2 Stereo Camera):

```
gst-launch-1.0 zedsrc camera-resolution=3
camera-fps=30 stream-type=0 ! \
  videoconvert ! x264enc byte-stream=true
tune=zerolatency speed-preset=superfast
bitrate=10000 ! \
  h264parse ! rtph264pay config-interval=-1
pt=96 ! \
  udpsink host=<CLIENT_IP> port=5000
sync=false async=false
```

This pipeline captures video from the ZED2 camera at 720p/30fps, encodes it with H.264, and sends it via UDP to the client for real-time preview.

Figure 20: Additional USB cameras (e.g., for manipulator view) use similar pipelines with v4l2src.

Length Measurement via ZED2 Camera

The ZED2 stereo camera generates a real-time pointcloud from its dual lenses. This pointcloud allows the measurement of objects in 3D space using Euclidean distance calculations and neural network optimization.

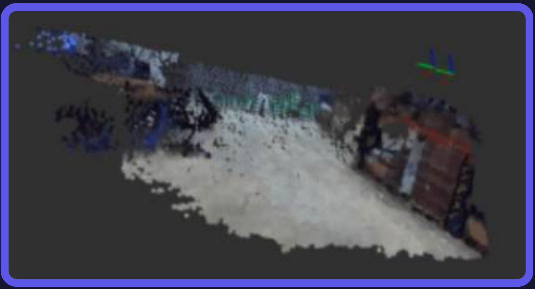


Figure 21: ZED2 camera Pointcloud

360-Degree Capture with Insta360 Air:

The Insta360 Air provides panoramic situational awareness using two fisheye lenses. Images are captured independently then stitched and converted for panoramic viewing.

Figure 22: Insta 360 Air

Viewing in FSP Viewer:

The output stitched.jpg is opened in a Fisheye-to-Equirectangular (FSP) viewer, allowing users toexplore the 360° environment. Proper calibration ensures accurate alignment of the two fisheye images.

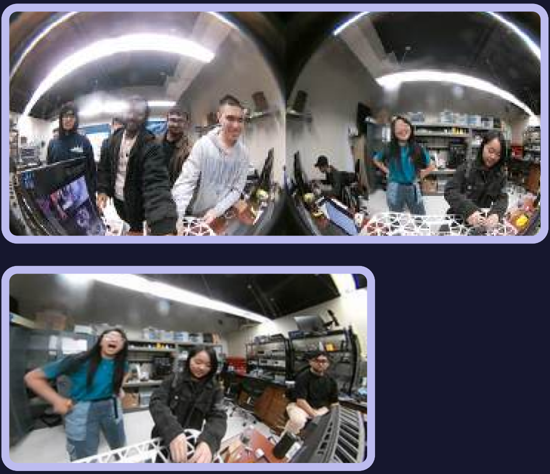


Figure 23: 360 Camera Output

Manufacturing

General Strategy:

As a rule, we decided to manufacture as much of our parts in house as possible to reduce on costs. By machining it ourselves we only have to pay for the materials. In addition to that, the large size of the top and bottom chassis plates make it hard to find machine shops with both the precision and bed size to fit our needs. Also, machining these parts ourselves reduced lead times as we could avoid the intermediary steps such as requesting quotes, shipping times, etc. We also generally avoided commercial of the self parts to both save on costs and provide more design flexibility.

The machines available to us were a CNC router, manual mill, and FDM printer in addition to other general shop machines like a belt sander, band saw, etc.

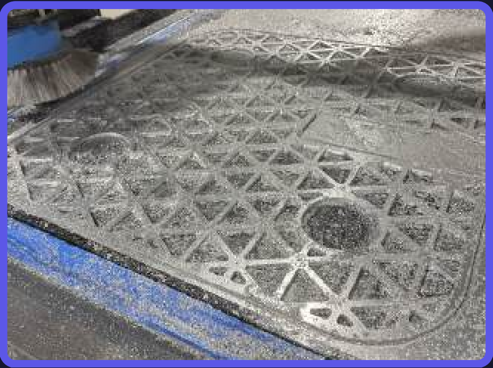


Figure 24: Chassis Plate after CNC File Finished

Due to the complex nature of these parts, it would have been extremely expensive to get them outsourced. For example, a local machine near us charges $50 per hour of work. If one of those plates took 10 hours, then it would cost us $500 per plate immediately. This is also ignoring material or shipping costs, which would not be insignificant as it is costly to ship such large items.

Chassis:

Most of the components on our chassis are custom manufactured, barring the thrusters and screws. All of the other parts were made with a CNC router and post-processed either on a manual mill or by hand.

The top and bottom plates were the main machining challenge for the ROV due to their size and hundreds of pockets. They each took about 10 hours of machining time. We counterbored all of the screw holes so they sit flush in the plate. This avoids having unnecessary screws sticking out of the surface of the robot and helps with tighter packaging. That being said, this meant that we had to machine features on both sides of the plate, further increasing the challenge of the machining them.

The 4 vertical support beams were also first machined on the CNC router. We also had to flip these over while machining for counterbores. Then we proceeded to bottom tap them on a manual milling machine.

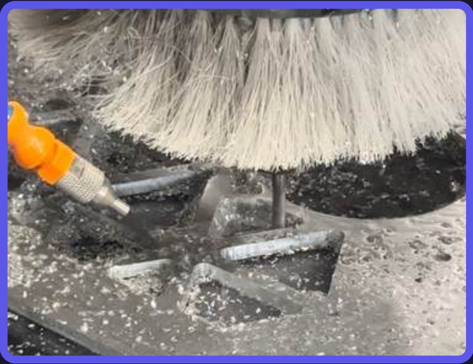


Figure 25: CNC Router Cutting Pocket Plate

After machining, we post-processed the parts to remove any sharp edges. To do this we started by sanding them using an orbital sander with at 400 grit and then moving up to 1400 grit sandpaper to create a smooth finish. After that, we broke the edges on all of the sharp corners by using a pneumatic deburring tool, which created smooth and consistent chamfers.

Manufacturing

Arm:

The arm is made out of a combination of CNC machined, 3D printed, and commercial off the shelf parts. We decided to buy some of the more intricate parts. For example, we used a servo spline mounting hub from goBilda instead of making our own servo attachment. It is challenging to make a servo spline in-house without the right broaching tools and setup so we decided it would be better to buy them instead. Also for components like bearings, those are far too intricate for us to make without specialized machinery so we also opted to buy those as well.

For the arm plates, we decided to use some eighth inch aluminum we received from a sponsor in order to keep it lightweight as opposed to the quarter inch used with the chassis plates. This way we had no upfront costs for machining the arm plates.

We used quarter inch aluminum for the 2 plates that the servos are mounted to and connect the arm to the chassis. Like the vertical support beams in the chassis, these were bottom tapped.

We decided to make the pulleys for the belts in house with our Bambu Lab P1S 3D printers. This gives a lot more flexibility with mounting the pulleys as opposed to buying commercially available metal pulleys.

Figure 27: 1/8th inch Aluminum Donated by Sponsor

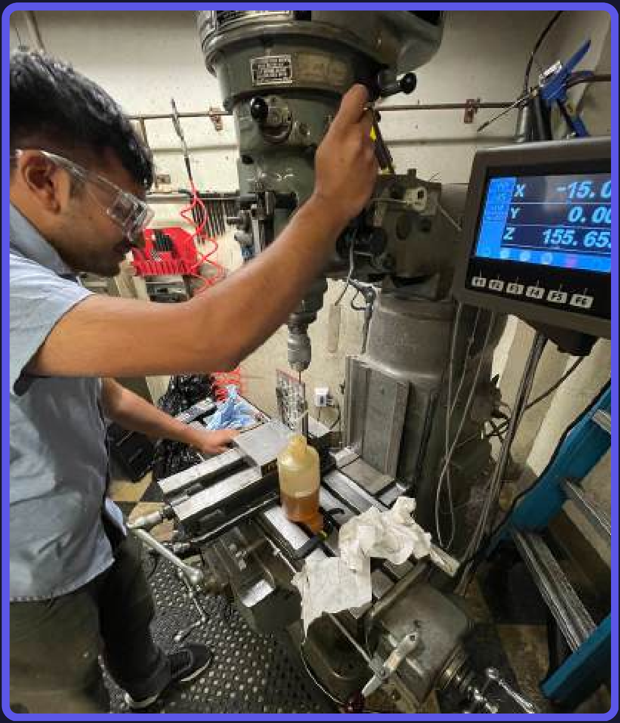


Figure 28: Bottom Tapping Holes on Manual Mill

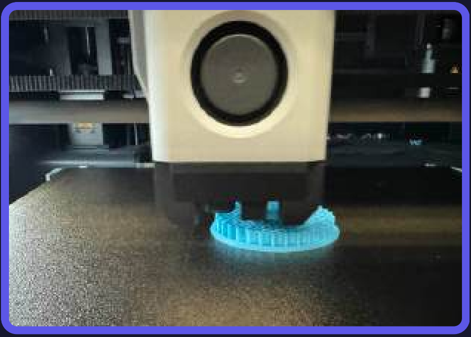


Figure 26: Bambu Lab P1S Printing Arm Pulley

Manufacturing

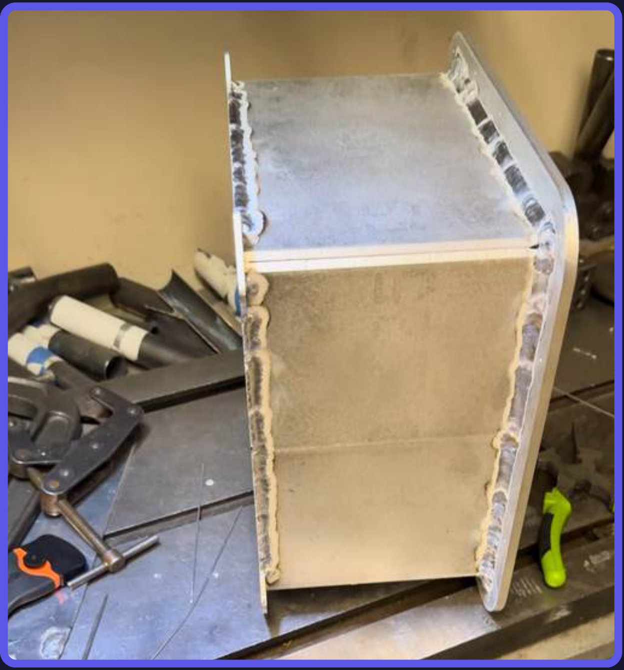


Figure 29: Partially Welded Ebox

Electronics Box:

Our original plan for the electronics box was to sheet metal bend eighth inch thick aluminum into a square and then weld it onto a base plate and a lid interface plate which has a seat for the O-Ring. This did not work out as planned as the grade of aluminum that our sponsor provided us was 6061-T4, which is not bendable. Therefore, we had to improvise and cut the bent sheet into 5 pieces and get them individually welded to the base.

Since our team does not have a welding setup, we decided to ask our neighboring design team, the SAE motorsports team, to help us with the welding. They were able to weld it in a way such that it was completely sealed and watertight. By working with the motorsports team, we were able to reduce our costs as opposed to having it done professionally.

PCB Manufacturing:

Outsourcing the manufacturing of our PCBs to a professional fabrication service like JLCPCB was not only a practical decision but also a necessary one. Our university's facilities are limited to basic double-sided prototyping using equipment such as the Voltera V-One, which cannot accommodate the complexity and precision required for our designs. The electrical team developed a 6-layer power board with 2 oz copper on the inner layers to handle high current loads (up to 24A), a 4-layer logic board, and a 2-layer backplane. These boards feature fine 8 mil traces, gold fingers with 30° chamfers, and dense via stitching. Manufacturing all of this in-house is far beyond the capabilities of our lab infrastructure.

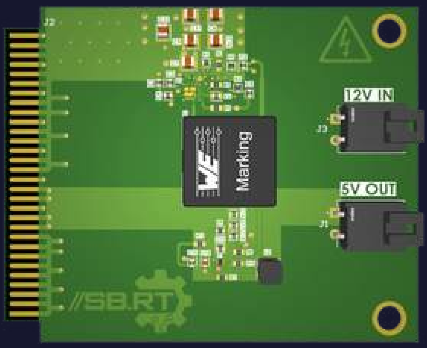


Figure 30: Render of Power PCB

By choosing JLCPCB, we leveraged their industrial-grade processes, which we had previously tested and trusted. All of the boards were fabricated in under a week and arrived within just two days of shipping, which dramatically reduced our turnaround time compared to any local or manual method. Their consistency, high-quality finish, and support for complex stackups and features allowed us to focus on electrical design and testing rather than worrying about fabrication logistics or quality assurance. Outsourcing to JLCPCB allowed us to meet competition deadlines, ensure electrical reliability, and stay within budget, which wouldn't have been possible through in-house production.

Figure 31: Ordering through JLCPCB

Critical Analysis

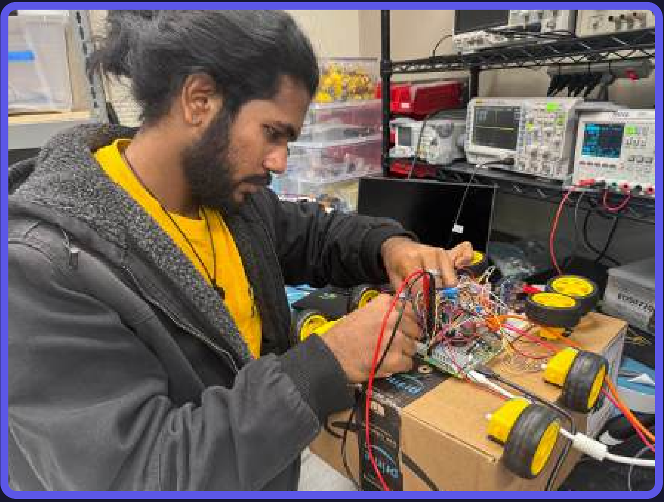


Figure 32: Cardboard Box Test Robot

Testing in Miniature Pool:

Our team doesn't have access to a full size pool at all times. To work around this, we contacted our team's advisor and he gave us access to the miniature pool in his design lab. We were able to use this to smaller scale software testing. We were able to get our first tests to check if our controller logic worked in the water with this pool. We were also able to use this pool to tune our PID controllers for stabilizing the ROV while driving. Using this pool saved us valuable time when we did get access to the larger pool to instead work on driver practice for the mission tasks and other tests that require more pool space.

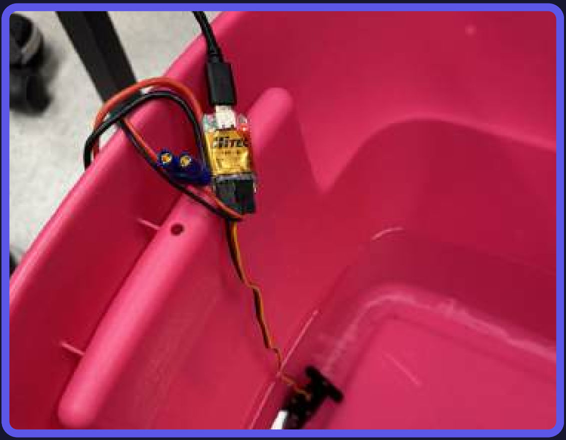


Figure 33: Servo Waterproof Testing in Bucket

Cardboard Box Controls Testing:

At the beginning of the year, we tested our control logic using a setup where we taped cheap hobby motors to a cardboard box. With this setup, we could test if the motors spun in the correct directions in response to the corresponding controller inputs. This allowed us to do software testing earlier in the year before the ROV was built.

This also lead to us deciding to use PCBs for our ROV as this simplified version of the ROV was already extremely disorganized, electrically speaking. A full version of the ROV would be even more unmanageable to wire without PCBs

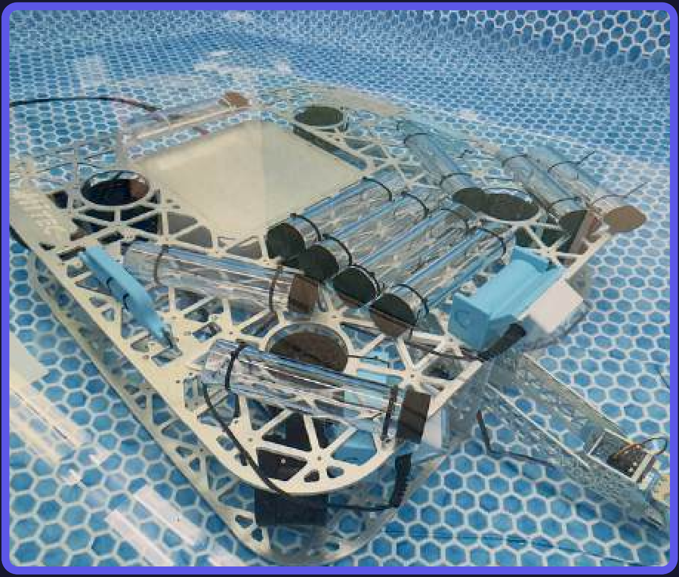


Figure 34: ROV in Miniature Pool

Large Bucket for Testing:

We were able to use this bucket in our lab to leak test components and check if everything was waterproof. We were able to confirm the servos we had gotten from our sponsor were properly waterproof, as opposed to other servos marketed on amazon to be waterproof. We were also able to fill this bucket up to leak test our electronics box and identify the locations of the leaks to seal them properly.

Critical Analysis

Figure 35: Pygame Thruster Simulation

Prototype Electrical Box:

We laser cut 5 sides of the electronics box and used a resin bond to connect them. We then used it spatially organize the components in the ebox before we making the final one. This allowed us to make small adjustments to the hole placement. We were also able to adjust the size of the box to be a bit larger as we need more space to more cleanly route the wires.

ROV Chassis Prototype:

In the middle of our competition season, we machined a prototype version of our chassis before finishing the entire design. Although it cost us more money to buy the extra aluminum, we decided it was worth it to check for any design flaws in advance and give the software team a physical robot to work with. We, at first, tried laser cutting the plates out of acrylic but realized that they were too flimsy to function properly so we decided to commit to using aluminum for the prototype.

This prototype, motivated several changes in our final ROV design. First, we originally, we planned to have the electronics box mounted on top of the ROV. After testing, we realized that this caused major balance problems and it would be best to repackage it to be inside the chassis.

We also realized that the electronics enclosure would be too large to fit inside the ROV as is, so we decided to expand the chassis by 2 inches on the final design.

Thruster Simulation:

At the beginning of the year, we decided to make a simulation of the thrusters using pygame. This let us do control logic testing without a physical robot and plan out how we would control the ROV. We were able to write the initial code to interface with the XBox controller we're using for competition with this simulation.

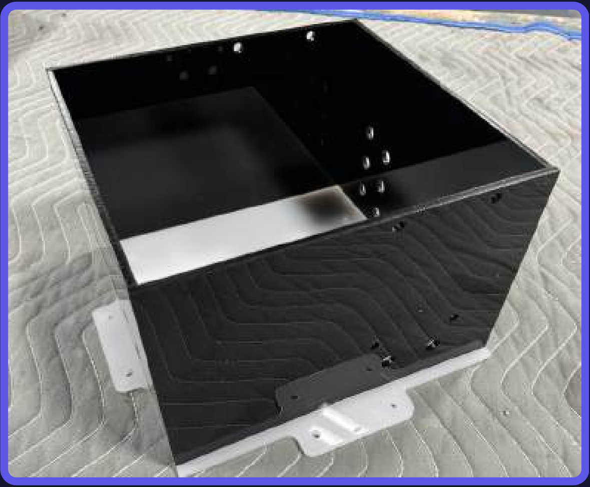


Figure 36: Prototype Electrical Box

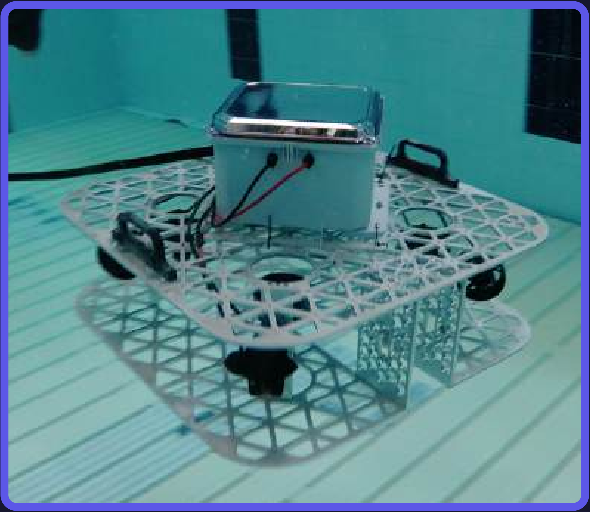


Figure 37: Chassis Prototype

Safety

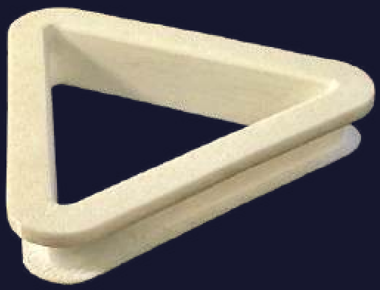


Figure 38: Printed Grommet

Custom Grommets:

To protect wiring in our metal chassis, we developed custom 3D-printed TPU grommets. The fully metal construction of our chassis presents a significant safety hazard for wire routing, as sharp edges can cut through wire sheathing even after deburring. Our custom grommets are specifically designed to fit securely into the chassis plate pockets, featuring a chamfered lip that ensures easy installation while preventing accidental dislodgement. This solution effectively protects all wiring passing through the chassis, eliminating the risk of electrical shorts and potential system failures that could compromise ROV operation.

Handles:

To address the handling challenges of our heavier metal-chassis ROV, we designed custom 3D-printed ergonomic handles. These handles attach securely to the top of the chassis, providing safe and comfortable lifting points. The design incorporates carefully measured fillets that create smooth edges to prevent hand discomfort during transport. We also optimized the handle dimensions by measuring actual human hands. This practical safety feature significantly reduces the risk of dropping the ROV during deployment and retrieval operations, protecting both the vehicle and team members.

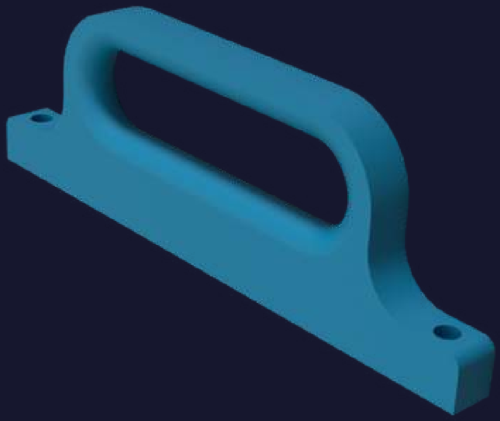


Figure 39: CAD Render of a Handle

Leak Sensor:

To mitigate risks associated with our custom electronics enclosure, we implemented a dedicated leak detection system. Recognizing that our student-built housing lacks the precision tolerances of professional manufacturing, we purchased a leak sensor as a critical safety backup. This sensor module monitors for water intrusion and immediately alerts operators through the GUI if moisture is detected. Upon receiving a leak notification, pilots can immediately retrieve the ROV from the water, preventing catastrophic damage to sensitive electronics and minimizing downtime for inspection and repairs.

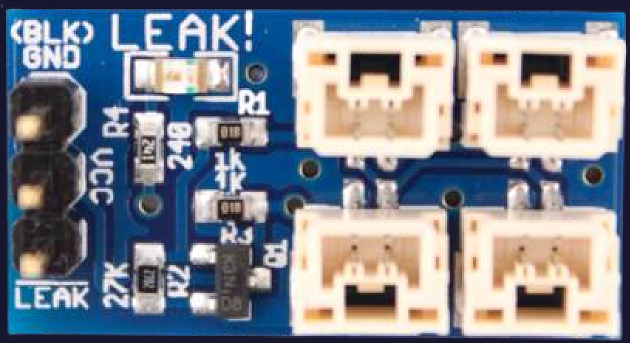


Figure 40: Blue Robotics SOS Leak Sensor

Safety

Check List for ROV Operation:

Our checklist serves as a critical safety protocol for ROV operations, ensuring consistent and thorough preparation before each deployment. This systematic approach prevents overlooking essential safety checks that could lead to equipment damage, operational failures, or safety hazards. By methodically verifying initial setup conditions, calibration parameters, and system integrity, the checklist minimizes human error and creates accountability in the preparation process. The structured format also provides valuable documentation for troubleshooting when issues arise, allowing the team to trace problems to their source. For student teams with varying experience levels, this standardized procedure ensures that even less experienced operators maintain the same rigorous safety standards, ultimately protecting both expensive equipment and team members while maximizing successful mission outcomes.

Checklist for SBRT

INITIAL SETUP

□ Power is off for ROV
□ Tether is checked for damage and is anchored securely
□ Screws are tightened for E box
□ Cables are properly tightened into E box
□ Router is properly plugged in and setup
□ Laptop is properly plugged in
□ Laptop has controller connected and on
□ Laptop is running front end code

CALIBRATION

□ Prepare power supply
□ Properly seal ROV
□ Plug ROV into the power supply
□ SSH surface laptop into jetson
□ Connect surface laptop to jetson server
□ Check if arm working properly
□ Place ROV in water before testing thrusters
□ Check if thruster array is properly calibrated
□ Check if PID is working properly
□ Check cameras are running
□ Check if there is a leak from front end code

LEAK DETECTION

□ Pull ROV out of pool
□ Turn off ROV
□ Dry ROV
□ Open EBox
□ Inspect EBox for leaks
□ Inspect sponges near leak sensor
□ Attempt to find, document, and fix source of leak

Accounting – Budget

		Reporting Period	
School Name	Stony Brook University	From:	8/26/2024
		To:	5/21/2025

Income

Source	Amount
USG Line Budget	$7,000.00

Expenses

Category	Type	Description/Examples	Projected Cost	Budgeted Value
Mechanical	Purchased	Aluminum, hardware, 3D printer filament	$500.00	$500.00
Electrical	Purchased	Monitor, controller. power strip	$300.00	$300.00
Tether	Purchased	10/2, Cat 6, nylon cord, sheathing, strain relief	$300.00	$300.00
Thrusters & ESC's	Purchased	8 Blue Robotics Thrusters & Basic ESC's	$2,000.00	$2,000.00
Electrical	Purchased	Jetson, sensors, cameras	$800.00	$800.00
Electrical	Re-used	ZED	$500.00	$0.00
Float	Purchased	Custom end caps, microcontroller, sensors, tube, PCB	$1,000.00	$1,000.00
Pool Access	Purchased	Lifeguard fees	$1,000.00	$1,000.00
Props	Purchased	PVC pipe, fitting, etc. for props	$600.00	$600.00
Fees	Purchased	Competition Registration	$650.00	$650.00
Travel	Purchased	Gas, tolls, parking	$800.00	$800.00
Lodging	Purchased	Hotel/Airbnb for 5 nights	$4,000.00	$4,000.00
			Total Income:	$7,000.00
			Total Expenses:	$12,450.00
			Total Expenses - (Re-use & Donations):	$11,950.00
			Total Fundraising Needed:	$4,950.00

Accounting – Project Costing

		Reporting Period	
School Name	Stony Brook University	From:	8/26/2024
		To:	5/21/2025

Funds

Type	Category	Expense	Description	Amount	Running Balance
	Starting Budget		USG Line Budget	$7,000.00	$7,000.00
Purchased	Props	PVC, fittings, etc.	All materials to make props	$(576.18)	$6,432.82
Purchased	Mechanical	Float parts	Custom End Caps, o'rings, tube, PCB components	$(600,05)	$5,823.77
Purchased	Electrical	Float parts	ESP32, pump, sensors, custom PCBs, PCB components	$(377.01)	$5,446.76
Purchased	Electrical	Tether	Power cable, ethernet, paracord, sheathing	$(322.04)	$5,124.72
Purchased	Tools	Machines and Tools	3D Printers, waterproof camera, clamp multimeter, crimpers, consumables	$(2,046.50)	$3,078.22
Purchased	Mechanical	Chassis parts	Aluminum, cord grips, o'rings, misc. hardware	$(1,023.18)	$2,055.04
Purchased	Mechanical	Thrusters & Buoyancy	Thrusters, ESC's, weights, water bottles	$(1,655.67)	$399.37
Purchased	Electrical	Control Station Parts	Xbox controller, seconds monitor, extension cord, router	$(120.86)	$278.51
Cash Donated	Income		USG Asset Grant	$1,428.00	$1,706.51
Parts Donated	Mechanical	Servos	Servos and programmer	$(742.95)	$963.56
	Income	Fundraiser		$42.00	$1,005.56
Purchased	Electrical	Power components	Power PCBs, components, bucks converters, busbars, wires	$(822.03)	$185.53
Purchased	Electrical	Electrical components	Logic PCB's, components, Jetson, cameras, sensors	$(1,593.69)	$(1,410.16)
Purchased	Administrative	Lifeguard Fees		$(792.00)	$(2,202.16)
Purchased	Administrative	Competition Registration		$(650.00)	$(2.852.16)
Purchased	Lodging	Hotels & Airbnb		$(3,558.62)	$(6,410.78)
				Total Raised	$1,428.00
				Total Spent	$ (14,838.78)
				Final Balance	$ (13,410.78)