RoboSub 2026 Technical Design Report: *Project Nebula (Dallas, Texas)*

Ben Bowyer, Liam Bray, Nicholas Cheng, Evan Destafano, Mitchell Hulme, Joel Lee, Alan Nguyen, Ethan Rojas

***Abstract**: For RoboSub 2026, Project Nebula deploys its first autonomous underwater vehicle, built on radical affordability, extreme modularity, and strategic simplicity. Transitioning from our successful MATE ROV platform (Project OMEGA), our vehicle achieves a 96% cost reduction versus typical university AUV programs while maintaining competitive capability through intelligent Cat6e tether carrying gigabit component selection and proven architectural reuse. Our competition strategy targets Tasks 1, 2, 3, and 6—Gate (Begin Assessment), Slalom (Avoid Debris), Bins (Recon), and Return Home—executed via a 6-thruster omnidirectional vehicle running ROS2 Humble with classical computer vision. Distributed computing across two Raspberry Pi units, a Blue Robotics pressure hull, and a modular electronics architecture proven over 15 pool deployments form the foundation of our design. This paper details the competition strategy, mechanical, electrical, and software design decisions, and the testing methodology that validates our approach.*

I. COMPETITION STRATEGY

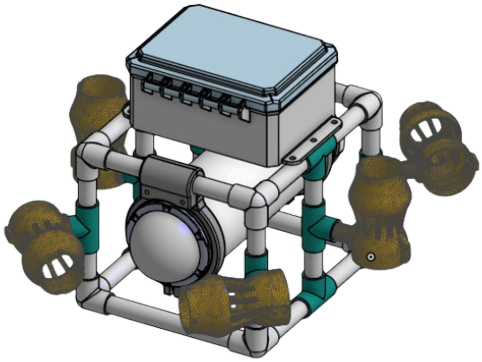


Fig 1: Full render, Project Epsilon

A. Core Philosophy

Project Nebula is an independent nonprofit based in Dallas, Texas, competing in RoboSub for the first time. Unlike university-affiliated teams with institutional funding and dedicated lab space, our team of eight self-funded this campaign through community donations and fundraising. This constraint is not a limitation; it is the organizing principle of our entire design. Every decision traces to a single question: Does this enable task completion at minimum viable cost and complexity?

Our MATE ROV experience with our previous ROV, Project OMEGA, demonstrated that modularity and simplicity yield reliability under competition conditions. We carry this philosophy directly into our AUV design. Systems engineering decisions favor proven approaches over novel ones, battle-tested components over cutting-edge alternatives, and debuggable software architectures over powerful but opaque ones.

B. Task Selection and Rationale

We target Gate (begin assessment, including style maneuvers), Slalom (avoid debris), Bins (reconnaissance), and Return Home. This selection exercises every core subsystem: navigation, vision, control, and manipulation, without requiring acoustics or Doppler Velocity Log (DVL) hardware that would triple vehicle cost while adding integration risk disproportionate to scoring benefit.

The task-to-system mapping is shown in Table I. Gate and style validate our depth hold and 5-DOF omnidirectional control. Bins (Recon) validate our color-based vision pipeline under

real pool conditions. Bins (Recon) validates our marker-drop mechanism and bin identification. vision pipeline under real pool conditions. Return Home closes the run by re-navigating the start gate. This staged validation mirrors sound systems engineering practice: confirm subsystems independently before attempting integrated task execution.

Task	Vision	Control	Manipulation	Acoustics
Gate	Yes	Yes	No	No
Gate Style Bonus	Yes	Yes	No	No
Bins (Recon)	Yes	Yes	Yes	No
Return Home	Yes	Yes	No	No

Fig 2: Task-Requirement matrix.

C. Complexity–Reliability Trade-off

As a first-year AUV team with limited pool access, we face an acute version of the complexity-reliability trade-off every Robosub team encounters. Time spent integrating an additional sensor is time not spent testing and hardening existing capabilities. We resolve this by drawing a firm architectural boundary: no sensor or subsystem is added unless it directly enables a target task.

Concretely, this means no DVL (saves $3,000–$16,000 and eliminates complex integration), no hydrophones (saves $1,200–$3,000 and removes acoustic pipeline entirely), and no GPU compute (saves $1,000+ and forces efficient classical vision). These omissions are strategic decisions rooted in task analysis, not capability gaps. For our target tasks, camera-based localization and IMU dead-reckoning provide sufficient positional accuracy.

This boundary also governs software. A finite state machine is simpler to debug than a behavior tree when pool time is limited. Classical OpenCV runs deterministically on a Raspberry Pi 5 without GPU acceleration. These choices sacrifice theoretical capability for practical reliability, the correct trade for a first-year platform.

D. Risk Assessment

Primary technical risks include first-time AUV integration exposing unknown failure modes and constrained pool access limiting iteration cycles. We mitigate these through extensive Gazebo simulation (100+ hours) before any pool deployment, dual electronics box redundancy enabling field swaps in under five minutes, and a modular mechanical design that allows component replacement without disturbing other subsystems. By accepting the known risk of a narrower task set, we reduce unknown integration risk, a needed platform for our new 2026 campaign.

II. DESIGN STRATEGY

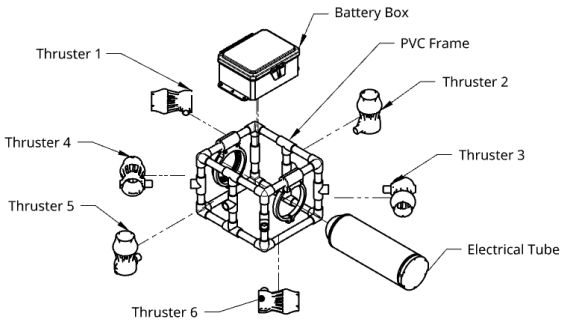


Fig 3: Project Epsilon Exploded View

A. Mechanical Design

The frame employs Schedule 40 PVC pipe in a cubic geometry approximately 50 cm on each side. This configuration directly enables our propulsion strategy: four horizontal thrusters at 45-degree angles in an X-configuration plus two vertical thrusters for depth control. The cubic geometry simplifies thrust vectoring mathematics, the same approach validated in Project OMEGA, and supports straightforward vector decomposition for all five degrees of freedom. PVC was chosen over aluminum for

its off-the-shelf availability, tool-free modifiability, and near-zero material cost (~$40).

1. Hull and Waterproofing

The primary pressure hull is a Blue Robotics 4-inch watertight enclosure (298 mm length, 100 m depth rating). This commodity component was selected after two failed in-house designs: an initial PVC-and-epoxy approach leaked at wire penetrators, and a single O-ring acrylic tube failed after three pool deployments. The Blue Robotics enclosure with aluminum end caps featuring double O-ring grooves, epoxy-coated infill, and dielectric grease on all sealing surfaces has achieved zero leaks across 15+ pools. deployments. Zorbeez hyperabsorbent PVA cloth lining the interior provides a final protection layer against any seepage before electronics are damaged. A slide-in electronics mounting plate allows full interior access without disturbing the O-ring seal, reducing maintenance time and seal disturbance risk.

2. Buoyancy and Ballast

The air-filled electronics enclosure provides approximately 2 kg of positive buoyancy. Four 3D-printed chambers containing adjustable lead shot (0–500 g each) mount at the bottom frame corners to achieve slightly positive buoyancy (~0.5 kg), providing stable hovering behavior when thrusters are idle and predictable surfacing upon power loss. Ballast chamber positioning enables pitch and roll trim adjustment without hardware changes.

B. Electrical Architecture

The electrical design prioritizes isolation, modularity, and proven reliability. One Turnigy Rapid 4S2P LiPo battery pack provides 8000 mAh capacity at 14.8 V nominal (4 cells, 140 C discharge rating) with an XT90 connector. A 20A blade fuse protects the main line. Power is split into two isolated rails: a motor rail supplying the six thrusters via three MDD3A dual motor controllers and a clean logic rail supplying computing and sensing hardware through 12 V-to-5 V DC-DC converters. This isolation eliminates the motor back-EMF crashes encountered in early MATE ROV development.

Computing is distributed across two Raspberry Pi units. An underwater Pi 5 (8 GB) handles autonomy, vision, and state estimation. Another dedicated underwater Pi 5 runs motor control via GPIO PWM at 50 Hz, isolating real-time thruster commands from main processing.

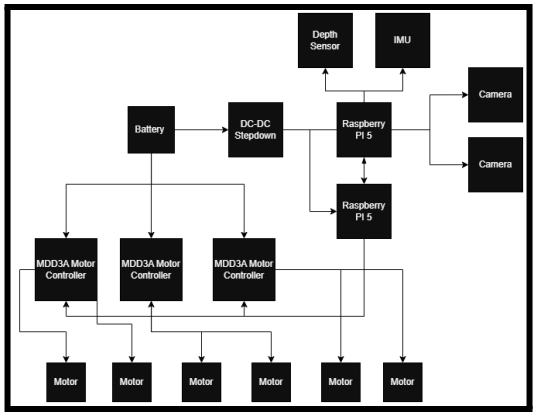


Fig 4: Electrical design architecture

Two complete underwater electronics boxes are maintained at all times. If a box develops a problem during a pool session, the backup unit is swapped in under five minutes, and testing continues. This redundancy strategy, carried forward from Project OMEGA, has prevented total session loss on two occasions.

C. Propulsion

Six SPX bilge pump thrusters ($38 each) were selected over Blue Robotics T200 units ($258 each) following a thrust-to-requirement analysis. For a vehicle of approximately 12 kg in air targeting 0.5 m/s forward velocity, the required thrust from the drag equation (CD ≈

0.8, A ≈ 0.25 m²) is F = 0.5 × ρ × v² × CD × A ≈ 25 N. Four horizontal SPX thrusters provide 4 × 7.85 N = 31.4 N, yielding a 1.25× thrust margin—sufficient for our lightweight design. The SPX provides the best thrust-to-power ratio of the three candidates evaluated, as shown in Table II, which was decisive given our battery capacity constraints. Total thruster cost savings of $1,320 represent the single largest cost reduction in the design.

Thruster	Price	Thrust (KGF)	Current (A)	$/kgf
SPX Bilge Pump	$38	0.8	2.5	$47.50
Diamond Dyn. TD1.2	$64	1.2	6.6	$53.33
Blue Robotics T200	$258	3.7	17.0	$69.73

Fig 5: Thruster comparison.

The software stack runs on ROS2 Humble Hawksbill, chosen for its modular node architecture, built-in debugging tooling (RViz, rqt, rosbag), and strong community support [1]. All code is version controlled on GitHub and deployed via Docker containers for reproducible builds across development machines [2]. A CI pipeline runs unit tests and linting on every commit.

1. Perception

The perception pipeline uses OpenCV [3] classical computer vision rather than deep learning. This decision was driven by compute constraints: YOLO11 inference on the Pi 5 achieved only 2–3 FPS at 1080p—below the 10 FPS minimum for stable visual servoing. Switching to HSV color thresholding plus contour detection for bins (Recon), Hough line transform for gate posts, and template matching for Gman restored 30 FPS at 1080p while reducing CPU utilization from 80% to 40%, preserving headroom for control and state estimation. Preprocessing applies red-channel attenuation compensation for underwater color shift and Gaussian blur (5×5 kernel) for noise reduction. Detected objects are published as ROS2 messages containing object ID, estimated 3D position via monocular depth from known object sizes and camera intrinsics, and confidence score.

2. Localization

A 13-state unscented Kalman Filter [4] fuses data from the Adafruit BNO055 IMU (100 Hz, orientation, angular velocity, and linear acceleration), the Blue Robotics Bar30 depth sensor (10 Hz, absolute Z-axis), and visual odometry derived from feature tracking between frames (10 Hz). The UKF prediction step integrates IMU and commands thrust for dead-reckoning; the correction step fuses depth and visual measurements. Outlier rejection discards measurements beyond 3σ from the prediction. Detected landmarks such as gate posts and buoy positions trigger position resets when available. Without landmarks, positional accuracy degrades to ±50 cm under dead-reckoning alone—acceptable for our target task set.

3. Control System

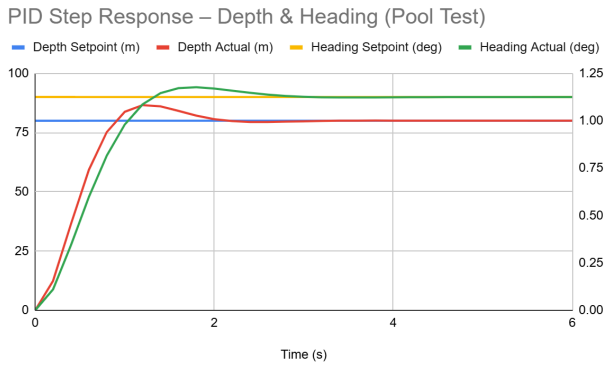


Fig 6: PID Step Response (Pool Testing)

A cascaded controller converts waypoint goals to thruster commands. A trajectory planner generates quintic polynomial paths respecting physical limits (max velocity 0.5 m/s, max acceleration 0.3 m/s²). Five independent PID feedback loops, one per degree of freedom, plus a model-based feedforward term derived

from drag and added-mass estimates account for approximately 70% of required thrust. The Moore-Penrose pseudo-inverse control allocation matrix, $\mathbf{f} = \mathbf{B}\dagger\mathbf{w}$ (full derivation in Appendix D), maps desired body forces and torques to individual thruster commands, optimizing for power efficiency via weighted least squares. The control loop runs at 50 Hz on the dedicated Pi 5. Gains were tuned via Ziegler-Nichols in simulation and refined in-water.

4. Mission Planner

Mission execution uses a finite state machine rather than a behavior tree. For a first-year team with limited pool time, the state machine's linear debuggability outweighs the composability advantages of behavior trees. Each state has an explicit timeout (60 s) and fallback: vision loss switches to 10 s dead reckoning before task abort; depth sensor failure triggers immediate surface; battery below 20% aborts mission. The task sequence is Initialize → Submerge → Gate → Style → Bins (Recon) → Gman → Surface, with each state independently testable.

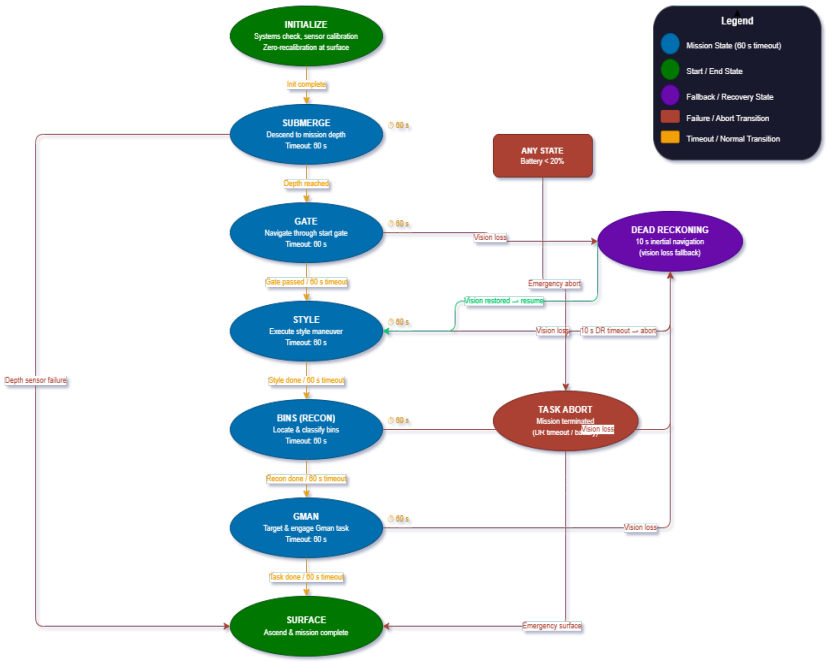


Fig 8: Finite State Machine

III. TESTING STRATEGY

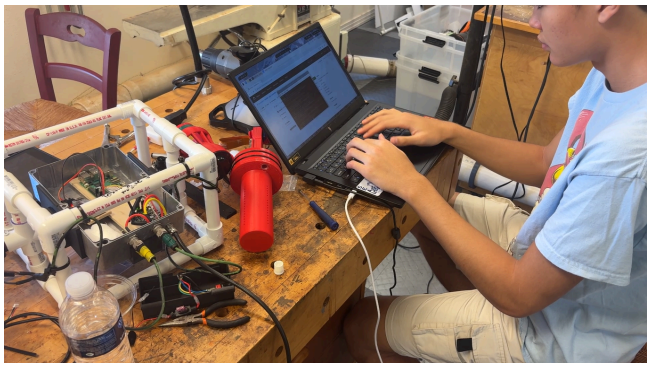


Fig 7: Preseason testing platform

A. Test Platform

To validate core software and electrical systems prior to full vehicle integration, the team constructed a dedicated test submarine. This platform is intentionally minimal, comprising two brushless motors for basic thrust and maneuvering, a Raspberry Pi as the onboard compute unit, and a Raspberry Pi camera module for forward-facing visual perception. By decoupling testing from the primary vehicle, the team was able to iterate rapidly on control logic and perception pipelines without risking damage to competition hardware or creating scheduling conflicts with mechanical and electrical development.

B. Perception and Control Validation

The Raspberry Pi camera was used to validate the computer vision pipeline under real underwater conditions. Detection algorithms and image preprocessing routines were developed and tuned against live camera feeds in pool environments before being ported to the primary vehicle stack. The two-motor configuration, while limited to planar motion, was sufficient to verify closed-loop heading control and basic station-keeping behavior.

C. Software Integration Testing

The test submarine served as the first integration target for new software modules. Each component, including sensor drivers, the

camera interface, and the motor control layer—was validated on the test platform before integration into the full system. This approach reduced the risk of regressions on the primary vehicle and allowed software team members to conduct testing independently of hardware availability on the main AUV.

ACKNOWLEDGEMENTS

Project Nebula's development would not be possible without the support of our mentors, community, and sponsors. We are deeply grateful to Douglas Johnson for providing workshop space, technical guidance, and unwavering support throughout six years of robotics development. Our sincere thanks go to Courtney Hulme, P.E., for engineering mentorship and professional guidance. We extend our gratitude to our two anonymous donors whose combined $2000 contribution made this campaign financially viable and to the Dallas maker community for donated 3D printing materials and fabrication time. We'd like to thank James Hovey for his support throughout our early years. His mentorship and guidance inspired us to continue even in the darkest hours. Finally, we thank our team families for supporting weekend workshops and late-night build sessions throughout this project. Your support has meant the world to all of us, so from the bottom of our hearts, thank you.

REFERENCES

[1] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics, vol. 7, no. 66, 2022.

[2] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," Linux Journal, vol. 2014, no. 239, p. 2, 2014.

[3] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[4] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," Proceedings of the IEEE, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[5] T. I. Fossen, Handbook of Marine Craft Hydrodynamics and Motion Control, 2nd ed. Hoboken, NJ: Wiley, 2021.

[6] Blue Robotics, "Bar30 High-Resolution 300m Depth/Pressure Sensor," 2024. [Online]. Available: https://bluerobotics.com

[7] Adafruit, "BNO055 Absolute Orientation Sensor," 2024. [Online]. Available: https://www.adafruit.com/product/2472

Appendix A: Parts List and Cost Accounting

Component	**Year purchased**	**New/Used**	**Price**	**Model/Type**	**Link**
PVC Frame	2022 - 2026	Used	$34.47	Structure	homedepot.com
Blue Robotics Tube	2026	New	$467.88	Structure	bluerobotics.com
Battery Box	2026	New	$79.00	Structure	bluerobotics.com
Penetrators	2026	New	$8.00	Structure	bluerobotics.com
SPX Motors	2019	Used	$120.90	Propulsion	progressiverc.com
MDD3A Motor Controllers	2020	Used	$87.00	Electronic	cytron.io
Voltage Regulator	2026	New	$12.99	Electronic	amazon.com
Raspberry Pi 5	2025	Used	$74.99	Electronic	raspberrypi.com
ArduCam Cameras	2026	New	$49.98	Sensor	arducam.com
IMU	2026	New	$34.95	Sensor	adafruit.com/product/2472
Blue Robotics Depth Sensor	2026	New	$68.00	Sensor	bluerobotics.com
Batteries	2026	New	$79.90	Power	hobbyking.com
TOTAL			$1,118.06		

Appendix B: Architecture

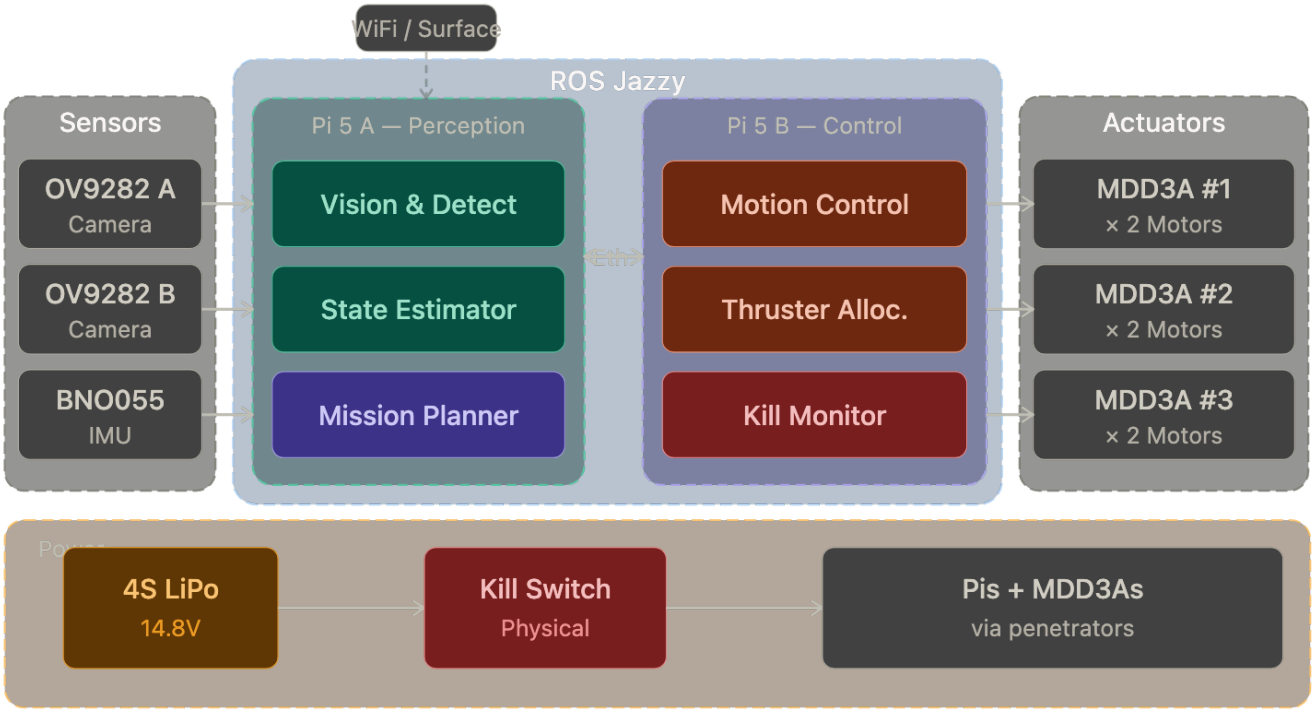


Fig. A.1: AUV system architecture overview. Motors: 4× horizontal (X-frame) + 2× vertical (roll / heave).

Appendix C: Score Estimation and Confidence

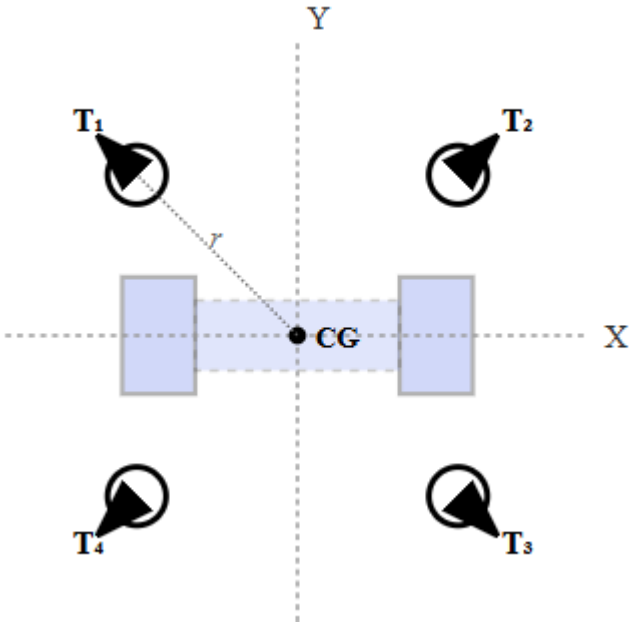
Task	Point Value (pts)	Est. P(Success) %	Expected Value (pts)	Key Subsystems Validated
Task 1 — Gate Passage	300	85%	255	Depth hold, 6-DOF omnidirectional control, IMU dead-reckoning
Task 2 — Slalom (Avoid Debris)	200	80%	160	Depth hold, course correction, visual pipe/obstacle detection, heading control
Task 3 — Bins (Recon)	400	60%	240	Bin/image detection, depth hold, station-keeping, marker-drop mechanism
Task 6 — Return Home	500	50%	250	Depth hold, sustained heading control, gate re-navigation
TOTAL (Targeted Tasks)	**1,400**	**~65% avg**	**905**	**Navigation, Vision, Depth Control, Marker-Drop**

Note: Expected Value = Point Value × Est. P(Success). Probabilities reflect first-year team estimates based on subsystem readiness and pool testing results. Tasks not attempted (acoustics, cameras-only tasks, pinger tracking) were excluded to minimize hardware cost and integration risk. Total available competition points exceed 1,400; our targeted subset represents ~65% average confidence across all attempted tasks.

Appendix D: Pseudo-inverse thrust allocator: Thruster Configuration and movement analysis

1. Thruster Configuration:

EPSILON



X-Configuration thruster layout. T_1 through T_4 represent individual thrusters positioned at 45°, 135°, 225°, and 315° from the positive X-axis, respectively. The parameter r denotes the distance from the center of gravity to each thruster.

2. Forward Kinematics

The allocation matrix represents the relationship between individual thruster forces and the resulting body forces. Each thruster can be represented by a radius and a thrust vector, respectively:

$$\mathbf{r}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad \hat{\mathbf{t}}_i = \begin{bmatrix} \cos\left(\frac{\pi}{4}\right) \\ \sin\left(\frac{\pi}{4}\right) \end{bmatrix} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The force generated by each thruster can be calculated by directly multiplying the thrust vector by a force scalar. The torque is the resultant of the cross product of the radius and thrust vectors.

$$\mathbf{F}_i = f_i \hat{\mathbf{t}}_i = f_i \begin{bmatrix} \cos\left(\frac{\pi}{4}\right) \\ \sin\left(\frac{\pi}{4}\right) \end{bmatrix}$$

$$
\begin{aligned}
\tau_i &= \mathbf{r}_i \times \mathbf{F}_i \\
&= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_i & y_i & 0 \\ F_{x,i} & F_{y,i} & 0 \end{vmatrix} \\
&= x_i F_{y,i} - y_i F_{x,i} \\
&= f_i \left(x_i \sin\left(\frac{\pi}{4}\right) - y_i \cos\left(\frac{\pi}{4}\right) \right)
\end{aligned}
$$

Thus, the kinematics of can be represented by:

$$
\mathbf{w}_i = \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ \tau_i \end{bmatrix} = f_i \begin{bmatrix} \cos\left(\frac{\pi}{4}\right) \\ \sin\left(\frac{\pi}{4}\right) \\ x_i \sin\left(\frac{\pi}{4}\right) - y_i \cos\left(\frac{\pi}{4}\right) \end{bmatrix}
$$

$$
\begin{bmatrix} F_x \\ F_y \\ \tau \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) & \cdots \\ \sin\left(\frac{\pi}{4}\right) & \sin\left(\frac{\pi}{4}\right) & \cdots \\ x_1 \sin\left(\frac{\pi}{4}\right) - y_1 \cos\left(\frac{\pi}{4}\right) & x_2 \sin\left(\frac{\pi}{4}\right) - y_2 \cos\left(\frac{\pi}{4}\right) & \cdots \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \end{bmatrix}
$$

3. Inverse Kinematics (Control Allocation)

We model the relationship between thruster forces and the resulting body force and torque as a linear system.

Because the system usually has more thrusters than controlled degrees of freedom, this matrix is not square and cannot be inverted directly. Instead, we use the Moore–Penrose pseudoinverse, which provides the least-squares solution that minimizes the total thrust magnitude while best achieving the desired force and torque.

The pseudoinverse gives the thruster forces that most closely produce the requested body wrench in a minimum-norm sense.

$$\mathbf{w} = \begin{bmatrix} F_x \\ F_y \\ \tau \end{bmatrix} = \mathbf{B}\mathbf{f}, \qquad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}$$

$$\begin{aligned} \mathbf{f} &= \mathbf{B}^{\dagger}\mathbf{w} \\ &= (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{w} \end{aligned}$$

$$\mathbf{B} = \begin{bmatrix} \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) & \cdots \\ \sin\left(\frac{\pi}{4}\right) & \sin\left(\frac{\pi}{4}\right) & \cdots \\ x_1 \sin\left(\frac{\pi}{4}\right) - y_1 \cos\left(\frac{\pi}{4}\right) & x_2 \sin\left(\frac{\pi}{4}\right) - y_2 \cos\left(\frac{\pi}{4}\right) & \cdots \end{bmatrix}$$

Combined with constraints, it is possible to derive the necessary outputs from each thruster to achieve each of the desired velocities.

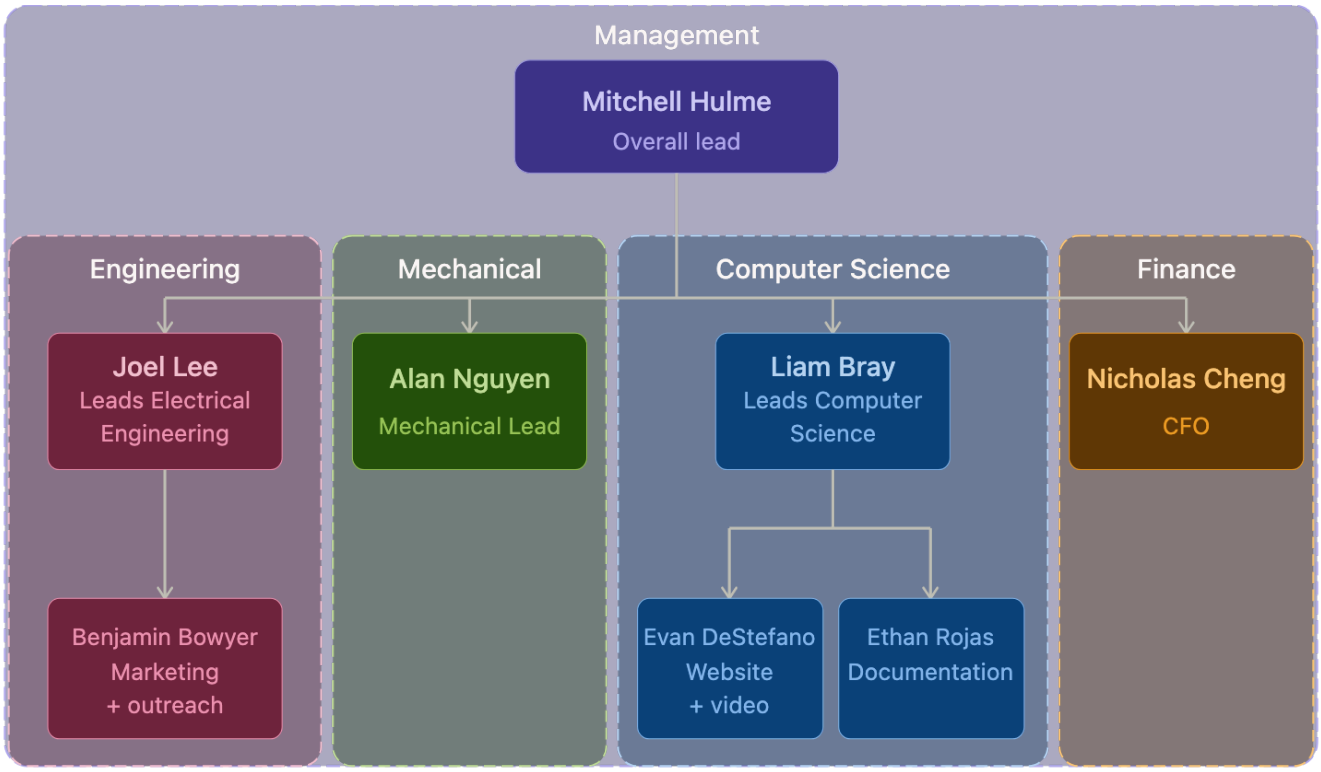
4. Motion Capabilities

The symmetrical design of the X-configuration offers several operational benefits. Pure translation in any direction is achievable by activating specific thruster pairs. Pure rotation is produced through differential thrust; T_2 and T_4 generate clockwise torque, while T_1 and T_3 produce counterclockwise torque. The system allows for simultaneous translation and rotation, enabling complex maneuvers like arc trajectories and point turns. Its balanced geometry ensures equal maximum acceleration in all horizontal directions, simplifying path planning and enhancing maneuverability in confined spaces such as competition courses.

5. Considerations

In real-world applications, control allocation must handle thruster limits. When thrust surpasses the maximum, the control system scales commands proportionally to preserve the desired direction while reducing magnitude.

Appendix E: Company Breakdown



Appendix F: Meta Llama Integration; Alternative Researched Solution

1. System Overview

Project Nebula incorporates an AI assisted robotics framework using a Raspberry Pi 5, live camera input, and a Meta Llama compatible language model. The subsystem explores how AI models can assist with autonomous navigation on embedded hardware.

Using live camera footage with OpenCV, frames are locally processed and converted into structured prompts. Finally they are interpreted by a locally hosted Llama model and translated into adjustments through the motor control system.This choice emphasizes modular local execution without requiring custom AI training.

2. Hardware and Software Integration

The subsystem is centered around the Raspberry Pi 5, which performs vision processing, AI communication, and command generation. A Raspberry Pi camera module provides a reliable and simple method to solving vision due to its GPIO interfaces.

The software consists of Python, OpenCV, Ollama, and Meta Llama GGUF running locally which provides a format for integrating llama to the pi system. OpenCV performs obstacle and pathway detection while Ollama hosts the Llama model for contextual reasoning and navigation interpretation.

3. Meta Llama Integration

Ollama hosts Meta Llama directly on the Raspberry Pi 5, requiring no fine-tuning or custom training. OpenCV detection results are condensed into structured natural language prompts describing obstacle positions and available pathways, which Llama interprets to return directional movement recommendations translated into motor command adjustments. A response timeout ensures automatic fallback to classical OpenCV-derived commands if inference stalls, maintaining vehicle stability. Due to inference latency inherent to running Llama on embedded hardware, the model functions as a supervisory advisory layer rather than a primary real-time controller.

4. *Advantages and Performance*

Running the Llama model locally provides several advantages, including offline operation, reduced latency, improved privacy, and elimination of recurring API costs. Lightweight quantized models are recommended to maintain acceptable inference speed and reduce memory usage on embedded hardware. Optimization may include reducing image resolution, and frame-skipping strategies.

5. Limitations

The subsystem remains limited by the processing power and thermal constraints of embedded hardware. Larger language models introduce additional inference latency and increased heat generation during continuous operation.

Future improvements may include GPU or Edge TPU acceleration, multi-camera depth perception, LiDAR integration, and hybrid AI architectures combining classical robotics algorithms with language-model-based reasoning.

6. Conclusion

The integration of Meta Llama with the Raspberry Pi 5 demonstrates how language models can support low cost autonomous robotics systems without requiring expensive infrastructure or custom AI training. By combining OpenCV computer vision with Ollama and Llama the system is capable of advanced reasoning and contextualization while remaining executable on mobile hardware.

Appendix G: Outreach Activities

1. School Demonstrations and STEM Engagement

Project Nebula actively engages local middle and high school students to inspire interest in robotics, engineering, and autonomous systems. Our team visits classrooms and STEM programs to demonstrate our previous ROV, Project OMEGA, and discuss our journey into AUV development. These sessions cover hands-on robotics concepts, the engineering design process, and what it means to compete in international robotics competitions as a self-funded, community-based team.

During these visits, students are invited to interact directly with the vehicle and ask questions about how it works, ranging from thruster control and waterproofing to computer vision and autonomous navigation. Our goal is not only to spark curiosity but to show students that competitive robotics is achievable without a university lab or institutional budget.

Fig. G1: Project Nebula team demonstrating Project OMEGA to middle school students in a STEM classroom.

2. Peer Mentorship and Community Sessions

Beyond formal school visits, Project Nebula hosts mentorship sessions with student groups and young engineers in the Dallas area. These small-group sessions allow us to engage more deeply with students who are actively working on robotics or engineering projects of their own, sharing lessons learned from our design iterations, hardware failures, and competition experience across both the MATE ROV and RoboSub programs.

These sessions are designed to help students bring their own projects and questions, with our team members sharing what they know from years of engineering work. We believe this peer-to-peer model

is extremely effective for students who may not have access to formal robotics programs or university mentorship networks.

Fig. G2: Project Nebula team member conducting a peer mentorship session with local students exploring robotics and engineering.