RoboSub 2026 Technical Design Report

Carnegie Mellon University (TartanAUV)

Aaron Ding, Aayan Maheshwari, Anthony Wang, Atharv Goel, Dylan Wu, Eleni Georgountzos, Eric Kwon, Nicolas Frazier, Zuri Ye
tartanauv@gmail.com

***Abstract*—In RoboSub 2026, TartanAUV will deploy Osprey, built in 2025 and refined through a season of testing and iterative development. Building upon the lessons learned from Osprey's first season, our team has undertaken a complete redesign of our software stack. We have replaced legacy systems with a robust architecture focused on reliability and maintainability. Alongside continued refinements to our mechanical and electrical systems, in-water testing has validated these developments. Through these improvements, TartanAUV aims to demonstrate a competitive autonomous underwater vehicle at RoboSub 2026 and beyond.**

I. Competition Strategy

A. Team Development

While last year's competition season centered on designing and manufacturing Osprey, this season introduced a new challenge: rebuilding and growing the team itself. As a relatively young RoboSub team, TartanAUV was originally founded in 2018, paused operations during the COVID-19 pandemic, and later rebuilt in 2022. With many of the members who reestablished the team graduating last year, we faced a significant loss of institutional knowledge. To address this, we invested time in improving documentation, onboarding, and internal communication while welcoming a new generation of members.

B. Focused Task Selection

Our competition strategy was highly influenced by this transition to a largely new team. In previous seasons, we have attempted every competition task, including the difficult manipulation tasks. Although we were ambitious, this approach often spread our efforts across too many systems and limited our ability to fully focus on individual capabilities. This year, we intentionally removed the *Resupply (Octagon)* manipulation task from our mission plan. By narrowing our scope, we were able to concentrate testing on areas most likely to provide consistent performance, such as navigation, localization, and our torpedo and dropper subsystems.

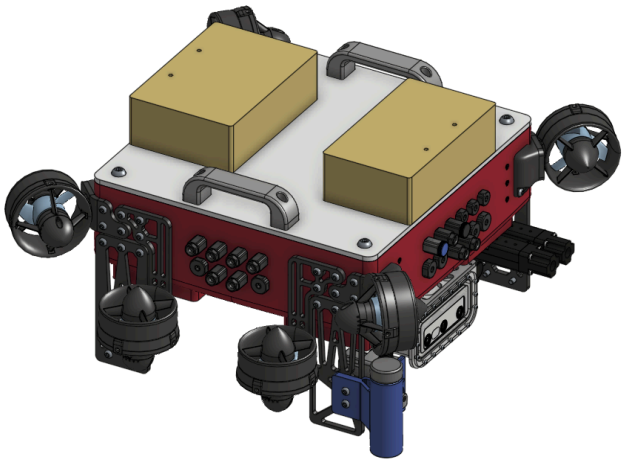


Fig. 1: Osprey Render

C. Task Prioritization

This decision also aligns with our 2026 technical objectives. A complete redesign of our software stack placed emphasis on localization, mapping, and navigation. Prioritizing tasks that directly leverage these improvements allows us to validate our core autonomy systems while maximizing potential points scored. Rather than attempting every available task, our goal is to execute a smaller set of tasks reliably and repeatedly. As a result, we plan to attempt the following tasks in order of priority:

1. **Vehicle Control:** *Coin Flip, Gate*
2. **Localization & Mapping:** *Slalom, Octagon* (Surfacing)
3. **Actuation:** *Bins, Torpedoes*

Although we intend to attempt all of these tasks, this hierarchy reflects both task complexity and our confidence in successful execution. By prioritizing foundational navigation objectives before higher-risk actuation tasks, we aim to maximize scoring opportunities while demonstrating reliable autonomous performance.

II. Design Strategy

A. Mechanical Improvements

Due to the modularity and high thermal overhead of Osprey's hull design, we chose to retain our vehicle's core mechanical architecture. We focused our efforts on developing external modules such as our dropper, torpedo launcher, camera enclosure, and landing struts. This approach allowed the team to dedicate resources toward advancing software, localization, and autonomy while still maintaining key mechanical systems that directly impact vehicle capability. Throughout this season, we emphasized maintainability, ease of integration, and rapid iteration across actuation systems, mounting, and any other external structures. Our goal was to maintain Osprey as a stable and adaptable platform for continued system development and iteration.

1) Mounting Layout

Due to the modularity of the Osprey's hull, we carefully develop a layout and integration of both internal and external vehicle components. Internally, our electronics are mounted on machined aluminum plates for improved thermal conductivity. These plates are installed on raised mounting points to prevent water damage and provide routing space for excess wiring underneath the assemblies. Externally, the limited mounting real estate necessitates intentional placement of and packaging of mechanisms, actuators, and enclosures to avoid interference between subsystems. This year we transitioned from a pair of LUCID Vision cameras to the OakD stereo camera mounted directly below the hull. To support this change, we CNC-machined a custom aluminum enclosure for the camera and redesigned several surrounding mechanisms to accommodate the additional space and mounting requirements while preserving accessibility and modularity across the vehicle.

2) Dropper design

Our dropper mechanism has been redesigned to minimize its overall footprint on our vehicle. Inside of the chamber, we stack two aluminum balls as our dropper objects. The servo actuator drives a spoked paddle which releases the balls sequentially, providing more controlled, repeatable behavior in a compact form.

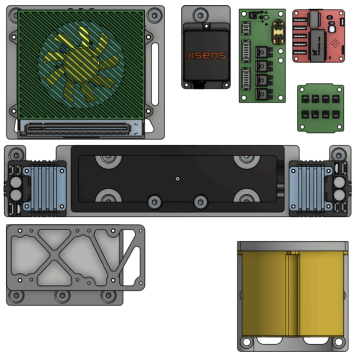


Fig. 2: Osprey electronics Layout

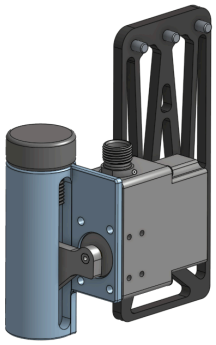


Fig. 3: Dropper module

3) Torpedo launcher

Our torpedo launcher has been reverted to a spring compression-driven design, a system previously used successfully during the 2024 competition season that demonstrated reliable and repeatable performance. Although last year's speargun-like surgical tubing design produced more thrust for the given barrel length, the current spring driven design is more compact, does not obstruct our camera vision, and more optimally conserves mounting real estate. Additionally, we introduced a two-stage trigger mechanism: the first stage prevents the torpedo from falling out of the barrel prematurely during maneuvers, while the second stage controls the release of the compressed spring, ensuring consistent performance.

4) Torpedoes

Our torpedoes are CNC machined out of a low density plastic for smoothness while maintaining buoyancy characteristics that allows for easy retrieval during testing. Spiraling grooves are cut into the external body to induce gyration of the torpedoes as they travel, improving accuracy and stability. Although the complex geometry can be 3D printed out of foaming PLA, the unpredictable nature of the plastic combined with the imperfections of the manufacturing process produces less desirable, low precision torpedoes. Although CNC machining is more time intensive, it proved to be the optimal manufacturing method after testing and analyzing both options.

B. Electrical Improvements

Last year's electrical system exposed limitations in reliability, sensing fidelity, and power management that constrained overall vehicle performance and maintainability. Our redesign addressed these shortcomings through a substantial overhaul of the vehicle's electrical architecture, including the removal of the Real-Time Vehicle Controller (RTVC), upgrades to critical sensing and actuation components, and the development of a custom power regulation and protection board. At the same time, proven hardware was retained where performance continued to meet system requirements.

1) Electrical Architecture

Previously, low-level communication with sensors and actuators was handled by the RTVC, a dedicated microcontroller between the Jetson and the vehicle's hardware. Persistent reliability issues and the unnecessary complexity the board led us to consolidate its responsibilities onto the *Jetson AGX Orin*, which has sufficient I/O and compute headroom to run both high-level autonomy and low-level device communication directly. To enable this, we designed a custom carrier board that breaks out the Jetson's native CAN controller and adds RS422 transceivers, allowing the Jetson to interface with peripherals the RTVC previously managed. The *Movella Xsens Sirius AHRS* and *Waterlinked A50 DVL* were also carried forward after confirming their performance met our requirements.

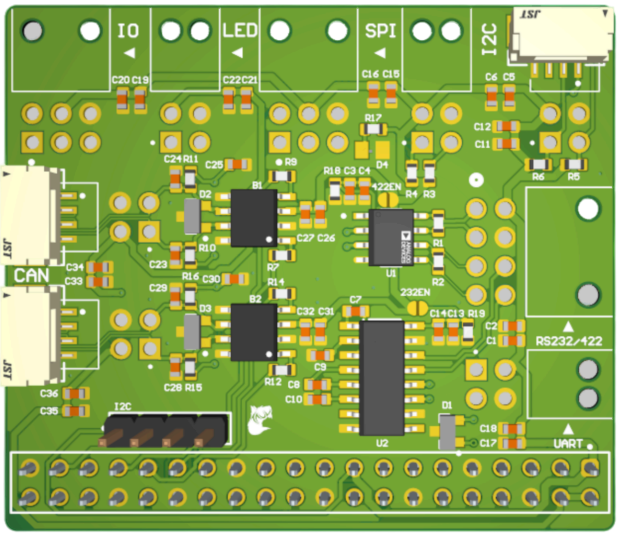


Fig. 4: Jetson Carrier Board

2) Sensor and Actuator Upgrades

Alongside the architectural changes, key component upgrades to the IMU and ESCs improve the quality of data available to the autonomy stack. Removing the RTVC made continuing with our *BlueRobotics* PWM-controlled ESCs impractical, as the Jetson lacks a dedicated real-time oscillator for reliable PWM generation. We switched to *Hargrave NanoDrive* ESCs, which communicate over CAN, provide per-thruster telemetry (RPM, current, temperature), and use a compact 4-in-1 form factor compatible with our T200 thrusters. We similarly replaced our PWM-controlled servos with the *BlueTrail SER-2130*, a CAN-based servo whose telemetry and configurable torque limits prevent the stall-induced failures we experienced with previous servos. To improve state estimation, we added a fiber optic gyroscope (FOG) alongside our Xsens IMU, interfacing with the Jetson via the carrier board's RS422 transceiver. During 15-minute pool tests, the Xsens alone exhibited yaw drift exceeding 10°, worsening under acceleration. The FOG helps eliminate this drift, providing a stable heading reference that the EKF fuses with MEMS data for high accuracy and bandwidth inertial measurements.

3) Power Distribution and Protection

Osprey is powered by a 16000 mAh 4S LiPo battery (14.8 V nominal). With our eight T200 thrusters drawing up to 20 A each, the theoretical peak powertrain demand is 160 A, well within the battery's 320 A continuous discharge rating. To protect the system, we designed a custom power controller that combines hotswap and undervoltage lockout functionality. The hotswap circuit enables battery changes with under 0.5 µs of downtime, laying the groundwork for an externally swappable battery that would allow continuous pool testing without powering down. The board also provides undervoltage lockout to prevent damage to the LiPo cells, and includes a latching 200 A short-circuit protection that disconnects the bus on a low-impedance fault. Together, these protections allow us to test without risking damage to the vehicle or injury to team members working poolside.

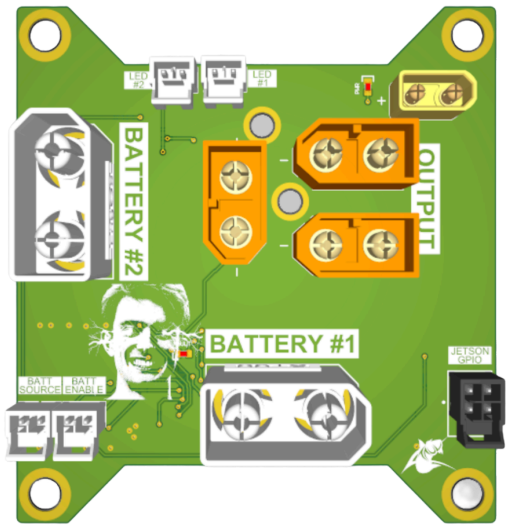


Fig. 5: Power Management Board

4) Tethering and Testing Infrastructure

To support both our testing workflow, we implemented two tether interfaces optimized for different stages of vehicle testing. A 50-meter drymate tether is used during extended pool testing and control algorithm development, providing continuous connectivity for real-time debugging, telemetry monitoring, and rapid software iteration. For competition operations, we use a 2-meter wetmate tether that enables vehicle startup, system verification, and mission initialization while Osprey is already in the water before disconnecting for fully autonomous operation. Because wetmate connectors have a limited mating cycle lifetime, restricting their use to competition runs helps preserve connector longevity and reliability. Together, these tethering solutions improve testing efficiency and reduce turnaround time between runs.

C. Software Stack

Over the past years, our codebase had accumulated significant technical debt, featuring sparse documentation, outdated practices, and what is colloquially known as "spaghetti code." These issues became especially apparent last year, when bringing Osprey to a functional state required substantial debugging and integration effort. Rather than continue maintaining and expanding an increasingly difficult-to-manage architecture, the software team chose to rebuild the codebase from the ground up. The following sections provide a detailed overview of each software subsystem. Note that our codebase was developed with minimal referral to our existing codebase to facilitate the implementation of modern practices and mitigate potential issues from attempting to scale too fast.

1) Simulation

Our simulation module features a custom *ROS 2* wrapper around *Stonefish,* an open-source library extending the *Bullet* physics engine for marine purposes. Using *Stonefish*, we hoped to model our sensor suite and thrusters well enough to virtually test our other modules. However, this proved more difficult than expected because of finding the physical parameters of the vehicle itself (e.g. center of mass, center of buoyancy, drag coefficients, etc). Because of its limited support and documentation, too, we plan on switching to *MuJoCo* or *IsaacSim* next year.

2) Drivers

Our Drivers module comprises the *ROS 2* nodes that interface with Osprey's onboard hardware. For the *Waterlinked A50 DVL* and *Movella Xsens Sirius AHRS*, we use established open-source drivers that provide tested, reliable communication with each device. For the *Bar02* depth sensor, fiber-optic gyroscope, ESCs, and servos, we developed custom drivers, since open-sourced, that prioritize fault tolerance through automatic reconnection on communication loss. During ESC driver development, we characterized each thruster's RPM and force response by sweeping command inputs, producing high-fidelity thrust curves that feed directly into our controller. The ESC and servo drivers also publish live telemetry (current draw, temperature, rotational speed) that the controller uses for real-time state estimation and fault detection.

3) Core

Our "core" module contains state estimation, watchdogs, launch files, and custom *ROS 2* messages for both the physical and simulated robot. For state estimation, we use an implementation of an EKF (Extended Kalman Filter) by the *robot localization* package in *ROS 2*. Our watchdogs are implemented via a *ROS 2* node that continuously monitors the state of the robot (e.g. battery voltage, orientation, etc) in case something goes wrong.

4) Autonomy

Our autonomy module includes all the processes related to the actual movement of the robot. For our controller, we use a 6 axis cascade PID controller, giving us control over both position and velocity in 3D space. To avoid uncontrollable movement by thruster saturation, we prioritize axes in the following order: *z (up/down), roll/pitch, yaw, x/y*. Our

autonomy module also contains a trajectory generator, which uses a quintic polynomial spline for point-to-point movement with continuous acceleration. Finally, a mission planner gives commands to the trajectory generator based on vision inputs and estimated state and a predefined behaviour tree.

5) Vision

Our vision module uses RGB + Depth images from an *Oak-D W PoE* camera to produce position estimates of competition task objects and point cloud maps of the environment. To ensure our object detection is robust in various lighting and water conditions, we use machine learning (*yolo26-pose*, small models) to identify keypoints on game elements. We use a mix of synthetic and real images for training, allowing the model to recognize the objects in situations that are less represented in real world datasets (extreme angles, high turbidity, etc). Synthetic data is generated using *NVidia IsaacSim Omniverse*, due to its ability to create scenes with photorealistic lighting. Finally, RGBD camera data is also used with the *RTABMap* library to create a 3D map of the environment. While computational overhead and lack of features in testing environments have posed some challenges, we've seen promising results when using geometry focused approaches for loop closure detection and feature matching.

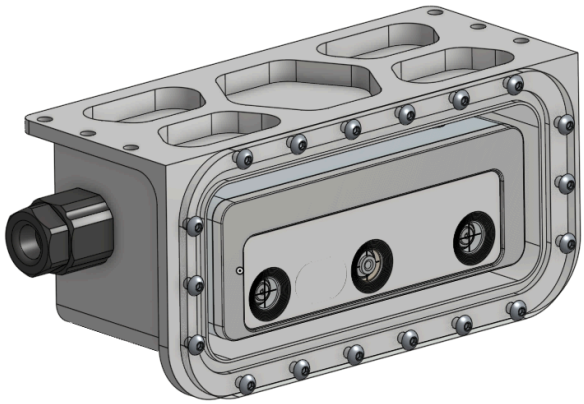


Fig. 6: OakD camera enclosure module

III. Testing Strategy

A. Isolated Electronics Testing

When bringing up new boards or testing components of Osprey, we prioritize validating each individual subsystem independently in order to isolate failures and prevent cascading faults throughout the vehicle. Each component is first inspected for shorts or low-resistance readings across all power rails before power is applied. The component is then connected to a current-limited power supply configured to the expected power draw, ensuring safe operation and preventing unpredictable behavior during initial testing. Once powered, the expected functionality of the component is verified, such as confirming proper switching behavior on power controllers, validating Ethernet link activity, or ensuring successful communication with ESCs. Finally, after the individual component passes standalone validation, it is integrated into the vehicle and all behaviors are tested again within the full system context.

B. Preliminary System Tests

Once the sub is pressurized to 10 psi, preliminary leak testing is first conducted on dry land by monitoring the vehicle's internal pressure sensor to ensure that internal pressure remains stable over 5-10 minutes. After passing these checks, testing moves to a small water tank where we can more quantitatively evaluate leaks under external pressure by weighing the vehicle down and partially submerging it. This stage also allows us to perform individual thruster testing and characterization, as well as basic controls verification such as confirming correct thruster spin direction and expected force output before full system deployment.

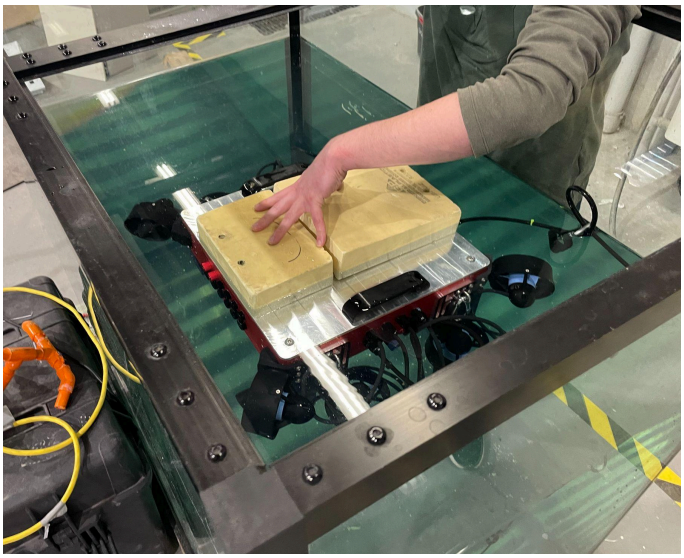


Fig. 7: "Baby" Tank Leak Test

C. Software Validation

To ensure system reliability and mitigate integration risks, we employed a rigorous, incremental testing strategy, verifying each software component independently prior to full-system integration. Hardware drivers underwent preliminary dry-land testing to validate low-level communication protocols and sensor readouts outside the aquatic environment. The state estimation and control packages were subjected to a multi-tiered validation pipeline, transitioning from initial algorithmic verification in a small but controlled water tank before moving to a pool-sized testing facility off-campus. High-level autonomy components, specifically the trajectory generator and mission planner, were first validated using deterministic Python-based unit tests, followed by empirical validation of the integrated autonomy stack during full-scale in-water vehicle trials.

D. Camera Calibration

We developed a collection of scripts to collect calibration data and validate generated camera calibrations. During data collection, we capture images of the calibration target across a wide range of distances, positions, and viewing angles to ensure that lens distortion and camera geometry are thoroughly characterized throughout the cameras' field of view. After calibration parameters are generated using the *Kalibr* software package, we validate the results by rectifying stereo image pairs and verifying that corresponding visual features align along the same image row. This process ensures accurate stereo rectification and improves the reliability of downstream depth estimation and visual perception tasks.

Acknowledgements

Our work as a team would not be possible without the support and guidance of mentors at Carnegie Mellon University, including Michael Kaess, George Kantor, Tim Angert, and Melisa Orta Martinez. We would like to thank the Field Robotics Center for providing access to the Robotics Institute machine shop and workspace. We are also grateful to Robert E. Bittner for allowing us to use the water tank in the newly built Robotics Innovation Center. We would also like to thank Catherine Copetas from the School of Computer Science for her tremendous help with our sponsorship efforts and coordination with university members. Additionally, we are grateful to Alicia Gorman, Caileigh Purcell, and the Carnegie Mellon Aquatics Department for their generosity in granting us use of the Cohon University Center pool for more extensive water testing. Finally, we would like to thank our generous corporate sponsors for their monetary and in-kind contributions. Our 2025–26 sponsors include Carnegie Mellon University, Metcal, NVIDIA, Leidos, Onshape, and Altium. We are proud to continue working with these partners to keep RoboSub thriving at Carnegie Mellon University and to broaden STEM outreach within our community.

References

M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," The International Journal of Robotics Research, vol. 31, no. 2, pp. 216–235, 2012.

T. I. Fossen and O.-E. Fjellstad, "Nonlinear modelling of marine vehicles in 6 degrees of freedom," Mathematical Modelling of Systems, vol. 1, no. 1, pp. 17–27, 1995.

G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," IFAC-PapersOnLine, vol. 53, no. 2, pp. 6563–6569, 2020.

E. Potokar, S. Ashford, M. Kaess, and J. G. Mangelson, "HoloOcean: An underwater robotics simulator," in 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 3040–3046.

J. Song, H. Ma, O. Bagoren, A. V. Sethuraman, Y. Zhang, and K. A. Skinner, "Oceansim: A gpu-accelerated underwater robot perception simulation framework," arXiv preprint arXiv:2503.01074, 2025.

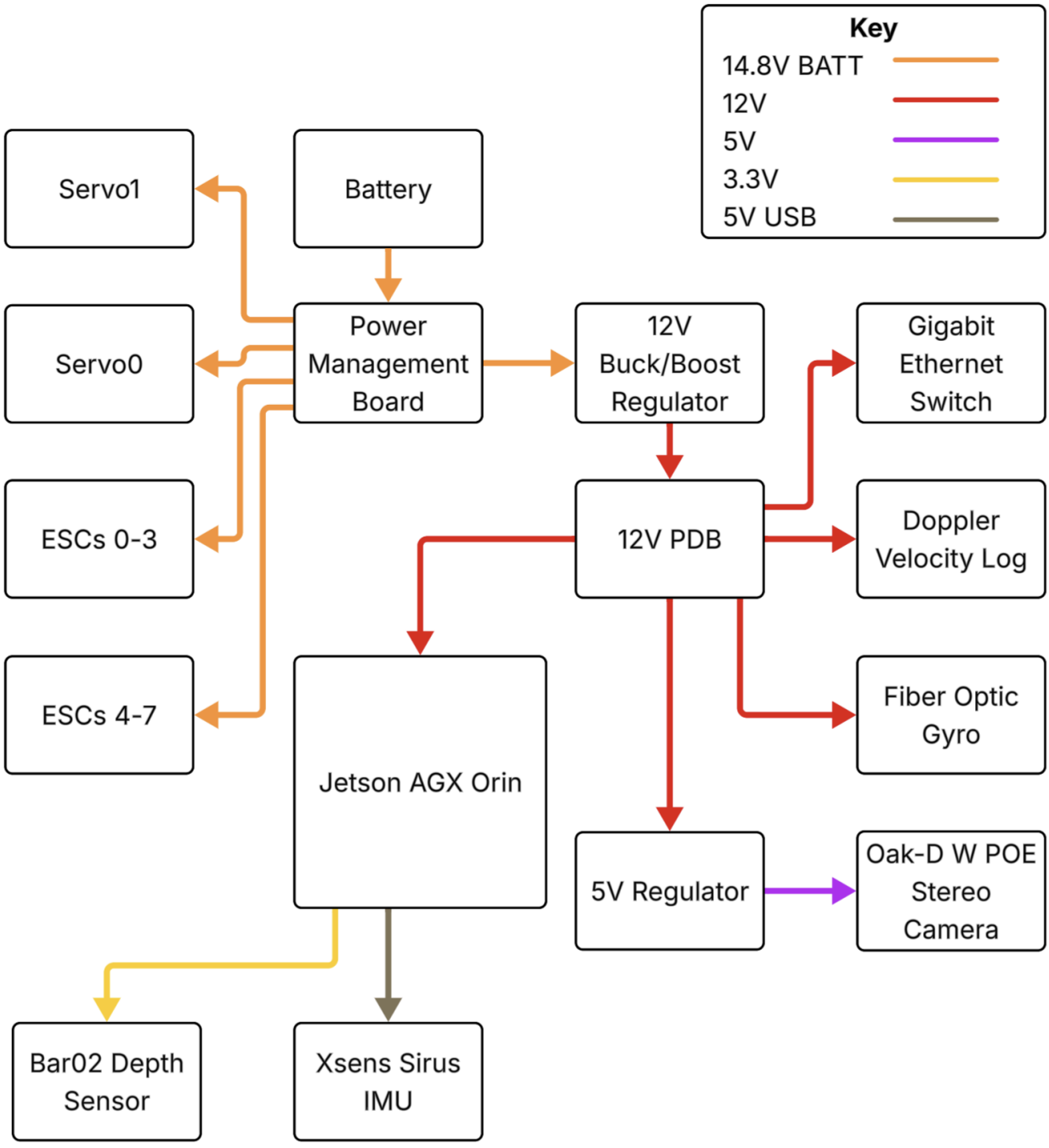
M. Grimaldi et al., "Stonefish: Supporting machine learning research in marine robotics," arXiv preprint arXiv:2502.11887, 2025.

Appendix A: Osprey Components

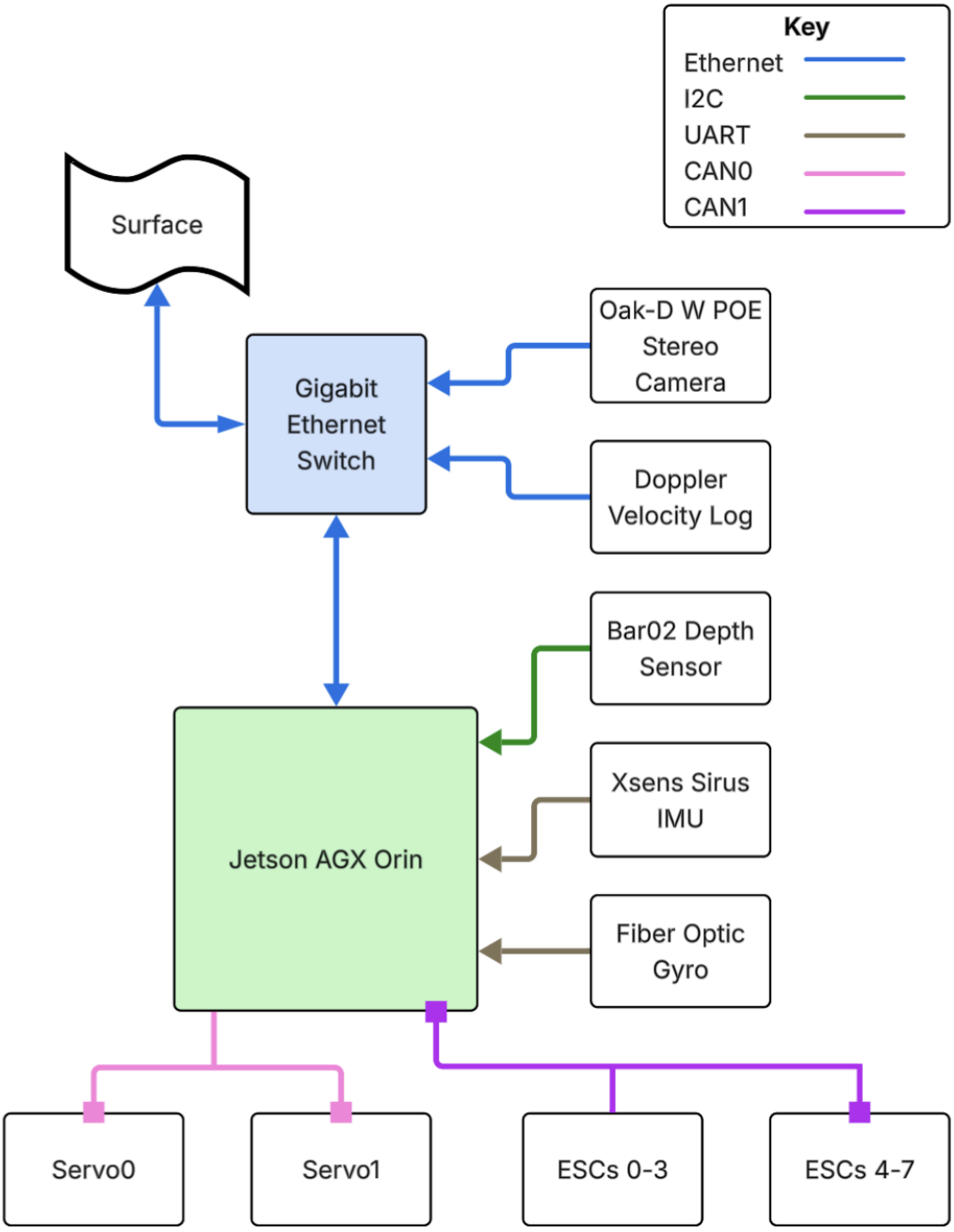
Component	Vendor	Model/Type	Specs	Custom/ Purchased	Cost	Year of Purchase
Hull	Custom	N/A	5 Axis Billet	Custom	$1,500.00	2024
Propulsion	Blue Robotics	T200 Thruster	Specs	Purchased	$270.00 ea.	2026
Waterproof Connectors	Blue Robotics	WetLink Connectors	Assorted	Purchased	$13.00-$22.00 ea.	2025
Actuation	Blue Trail	SER-2130	Specs	Purchased	$525.00 ea.	2026
Inertial Measurement Unit	Movella Canal Geomatics	Xsens Sirius AHRS KVH 1775	Specs Specs	Sponsorship Gifted	NA	2025 2023
Doppler Velocity Log	Waterlinked	DVL A50	Specs	Purchased w/ Sponsor Discount	$5,000.00	2025
Cameras	Luxonis	OakD POE W	Specs	Purchased	$500.00	2024
5V Regulator	Gobilda	6A BEC/Voltage Regulator	Specs	Purchased	$27.99 (Discontinued)	2025
Compute	NVIDIA	Jetson AGX Orin	Specs	Purchased	$2,000.00	2025
External Comm Interface	Blue Trail MacArtney	8-Pin Dry-Mate Connector Subconn Data Series 8-Pin Circular	NA Specs	Purchased Purchased	$80.00 $80.00	2025 2025
Internal Comm Network	BotBlocks	Gigablox	NA	Purchased	$125.00	2025
Motor Control	Hargrave	nanoDrive 4LPi	Specs	Purchased	$444.13	2025

Autonomy	Cascade PID Controller	2026
Localization & Mapping	Extended Kalman Filter	2026
Open Source Software	ROS2, Stonefish	NA
Programming Languages	C/C++, Python	NA

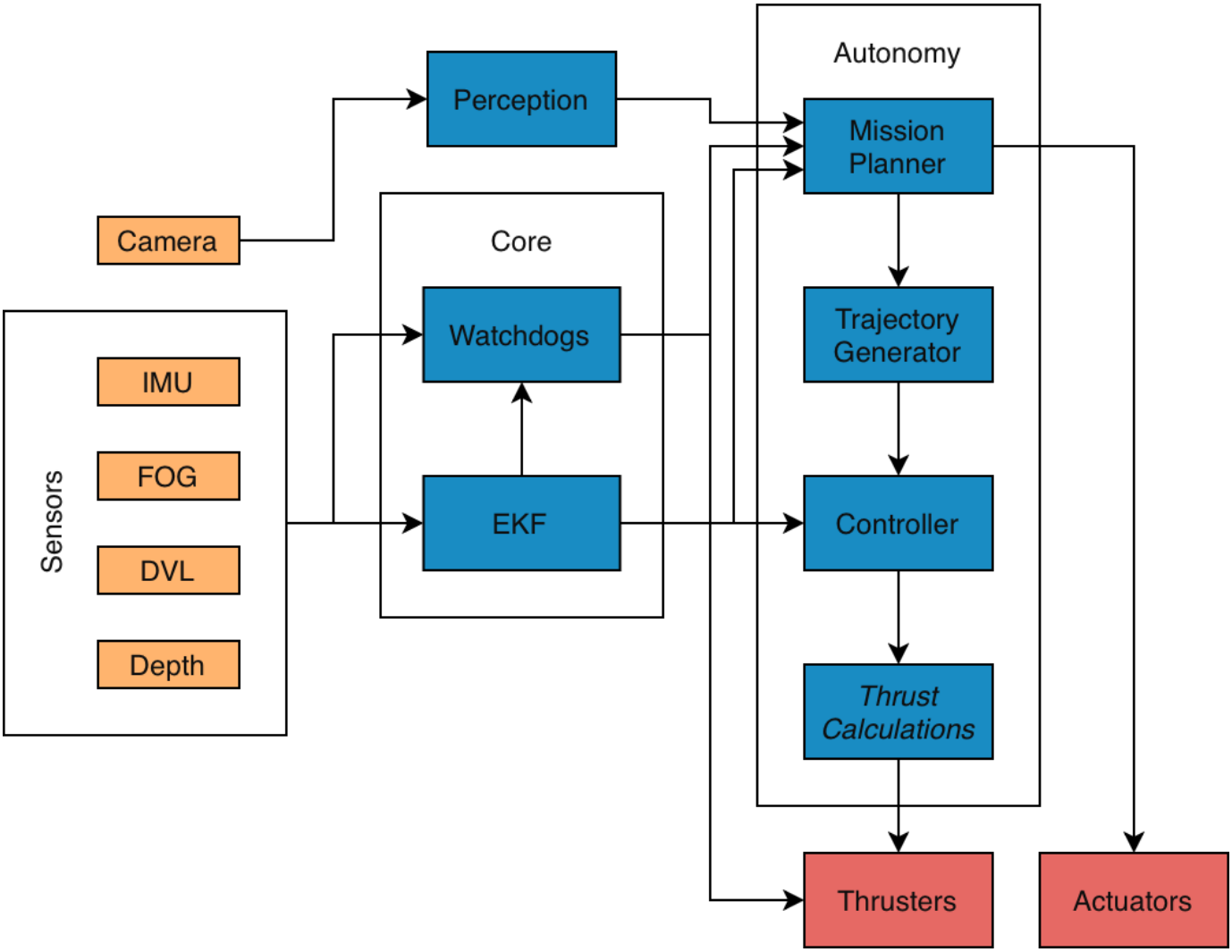
Appendix B: Electronics Architecture Block Diagrams



Appendix B: Electronics Architecture Block Diagrams (cont.)



Appendix C: Software Architecture Block Diagram



Appendix D: Exploded View of Dropper Mechanism