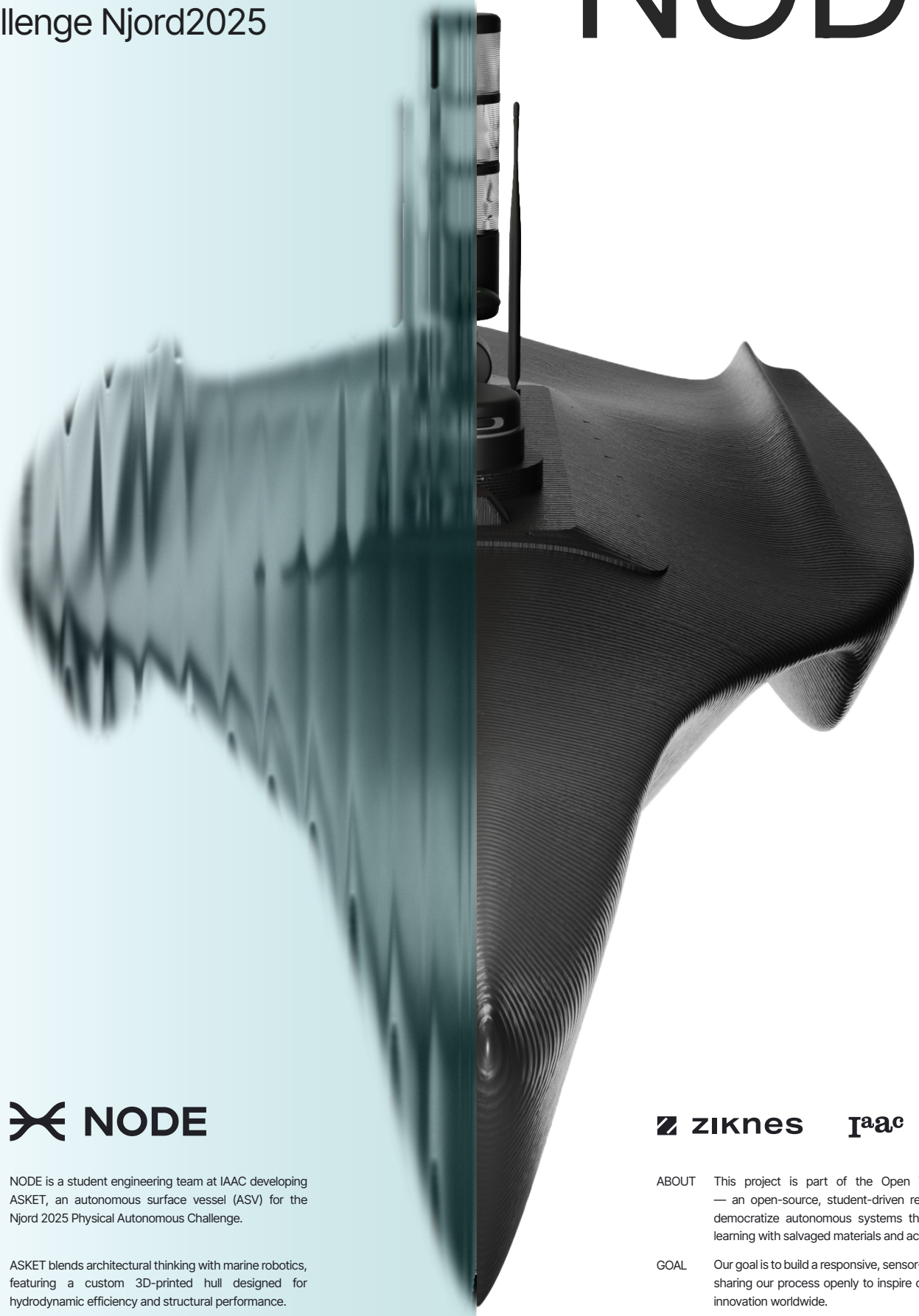
ASKET BY NODE

Technical Report: Physical Autonomous Challenge Njord2025



NODE

TEAM NODE is a student engineering team at IAAC developing ASKET, an autonomous surface vessel (ASV) for the Njord 2025 Physical Autonomous Challenge.

PROJECT ASKET blends architectural thinking with marine robotics, featuring a custom 3D-printed hull designed for hydrodynamic efficiency and structural performance.

ziknes IAAC

ABOUT This project is part of the Open Waters Initiative — an open-source, student-driven research effort to democratize autonomous systems through hands-on learning with salvaged materials and accessible tools.

GOAL Our goal is to build a responsive, sensor-rich vessel while sharing our process openly to inspire collaboration and innovation worldwide.

Index

1.0 Introduction..... 2
2.0 Vessel Design..... 3
2.1 Hull Design..... 3
2.2 Propulsion.....3
2.3 Batteries.....3
2.4 Sensors..... 3
2.5 Hardware Implementation..... .4
3.0 Software..... .4
3.1 Software Architecture..... 4
3.2 Software Design..... 4
3.3 Implementation.....4
3.3.1 Robot Communication Description..... 4
3.3.2 Obstacle Detection and Data Analysis..... .4
3.3.3 Navigation and Localization..... 5
3.4 GUI: ASKET's Control Interface.....5
3.5 Docking Strategy..... .6
4.0 Innovative Aspect..... 6
5.0 Testing..... .7
5.1 Vessel Testing.....7
5.2 Hardware Testing..... 7
5.3 Software Testing.....7
5.4 Integrated Testing.....8
6.0 Contributions..... 8
7.0 Appendix.....10

NODE

NODE is a multidisciplinary student team based in Barcelona, Catalonia and formed at the Institute for Advanced Architecture of Catalonia (IAAC). Made up of architects, designers, and engineers, we share a passion for robotic engineering, boat design, sustainability, and digital fabrication. For Njord 2025, NODE is developing ASKET - an autonomous surface vessel (ASV) that combines architectural creativity with advanced marine technology. The vessel features a custom-designed, 3D-printed hull, using additive manufacturing to optimize hydrodynamics and structural performance. The following paper summarizes the main technical aspects of NODE's ASKET ASV.

2.0 Vessel Design

2.1 Hull Design

ASKET features a **trimaran hull** for enhanced stability, low drag, and compact integration *(Figure 1)*. Inspired by multihull principles, this geometry improves roll resistance and reduces pitch in turbulent conditions *(Figure 2)*. The vessel consists of two 3D-printed assemblies *(Figure 3)*:

- A **monolithic main hull** that houses propulsion and two sealed compartments (Pixhawk & compute unit; LiPo battery), suspended from the electronics lid to optimize CG and isolation.
- A **modular electronics lid** that integrates external sensors (dual cameras, LiDAR, GPS, antennas, emergency stop) for quick access.

The hull measures **1.5 × 0.7 × 0.4 m**, with a **central hull** of 1.5 × 0.3 m and **outriggers** of 0.7 × 0.16 m, optimized for hardware fit. Final geometry was simplified from early designs to improve printability, sealing, and assembly, while preserving hydrodynamic performance *(Figure 4)*.

Printed in 12 hours on a **Ziknes Z-Pellet One** using glass microfiber-reinforced PETG, with 9–10 mm thick walls. The bow is solid, with machined features for print accuracy *(Figure 5)*. The lid was printed in PLA on a **Bambu Lab X1 Carbon**, ensuring high precision.

Waterproofing was achieved using mechanical interlocks, adhesives, custom gaskets, and a Smooth-On XTC-3D hydrophobic coating.

Reflections on Stability and Hydrodynamics:

Testing confirmed strong lateral stability, buoyancy under asymmetrical loads, and minimal pitching—thanks to tuned bow curvature and buoyancy centers. The outriggers enhance yaw resistance, while stern-mounted thrusters provide cavitation-free propulsion.

2.2 Propulsion

ASKET uses a **three-thruster underwater propulsion system** *(Figure 3)*:

- **2× rear thrusters** for forward/reverse and differential turning
- **1× lateral bow thruster** for sideways movement and docking

The propulsion system uses **APISQUEEN U2 Mini thrusters** paired with **APISQUEEN 30A bidirectional ESCs**. Operating on **12–16V**, each unit delivers up to **130 W**, controlled via **PWM (1–2 ms, 50 Hz)**. The thrusters are compact (95.8×77 mm, 210 g) and made from **marine-grade materials**.

The **ESCs**, compatible with **2–4S LiPo**, include an integrated **5V 1A BEC**, measure **28 × 15 × 6 mm**, and weigh **36 g**. They feature **95 mm 14 AWG power leads**, **300 mm motor leads**, and accept **PWM signals (1.5–2.0 ms forward, 1.5–1.0 ms reverse)**. Their firmware supports **up to 500 Hz** refresh rates for responsive, fine-grained control.

The system produces a combined **390 W output** and **3.9 kg of thrust**, achieving **1–5 knots** cruising speed and up to **30 minutes** autonomy.

Thrusters are mounted using **custom PLA brackets**, securely bolted to the hull for **minimal added weight** and **stable operation**.

Control is handled via **PWM from the Pixhawk 4**, with **power delivered through the PM07 power module** from the **14.8V main battery**.

2.3 Batteries

ASKET is powered by a single **GensAce 4S2P 8000mAh LiPo battery** (14.8V, 118.4Wh). It weighs **737 g**, measures **157×55×42 mm**, and features **JST-XHR**(balancing) and **EC5** (discharge) connectors. Supporting **25–30C discharge rates**, it delivers up to **200–240 A** peak output, with a safe cutoff at **12.8 V**.

Power Distribution:

- **Direct 14.8V** → thrusters & signal tower.
- **5 V regulated** (DUMVOIN 10A DC-DC) → electronics (Pixhawk, Raspberry Pi 5, GPS, LiDAR, etc).

Simplified single-battery system offers ~30–35 min runtime while maximizing internal space and reducing weight/complexity.

The battery is **centrally mounted** in a **sealed, fire-retardant compartment** with **top access** and a pending **emergency cutoff switch** for safety.

2.4 Sensors

ASKET integrates a **multi-modal sensor suite** for aquatic navigation:

LiDAR: **Slamtec RPLIDAR A2M8**, 360°, 10–15 Hz, 8,000 pts/sec, 8–12 m range. Balances **resolution**, **cost**, and **surface-level obstacle detection**.

GPS: **Holybro M9N**, 10 Hz, ~2.5 m accuracy (up to 1 m w/ SBAS), connected via **UART** to Pixhawk.

Cameras: **2× Logitech C270** (720p @ 30 fps), positioned **forward and aft** for situational awareness and visual detection.

Sonar: **IP68 ultrasonic**, range **5–600 cm**, ±(0.5–1 + S×%) cm accuracy, **8–16° beam**, UART interface.

IMU: Built into **Pixhawk 4**, includes **3-axis gyro**, **accelerometer**, **magnetometer** for **pose estimation**.

The system combines **LiDAR**, **GPS**, **vision**, **IMU**, and **sonar** for robust navigation, even in **GPS-denied conditions**.

2.5 Hardware Implementation

ASKET runs on a **dual-voltage architecture**. This ensures efficiency, compatibility, and safety.

- **Direct 14.8V** → thrusters & signal tower (max current, no conversion).
- **5 V regulated** (DUMVOIN 10A DC-DC) → electronics (Pixhawk, Raspberry Pi 5, GPS, LiDAR, sonar, cameras, RC receiver). *(Figure 6)*

The **PM07 Power Module** acts as the **primary power system**, distributing **14.8V directly** to the **thrusters** and **signal light**, ensuring high-current delivery and minimizing conversion losses.

The **secondary power system** is handled by a **DUMVOIN DC-DC converter**, which steps down the voltage to provide a regulated **5V 10A** supply for low-power electronics, including: **Pixhawk 4** (via PM07), **Raspberry Pi 5, Holybro M9N GPS, RPLIDAR A2M8, 2× Logitech C270 webcams, RC receiver, IP68 sonar.**

The total current draw (~**2.25A**) remains well within the converter's capacity.

Control is architected in two layers:

- **Raspberry Pi 5** → vision, LiDAR/sonar, high-level logic
- **Pixhawk 4** → PWM actuation, IMU/GPS, safety management *(Figure 7)*.

3.0 Software

3.1 Software Architecture

The ASKET ASV software stack is built as a **lightweight, modular Python architecture**, running entirely on a **Raspberry Pi 5**. It uses custom-designed TCP/IP communication layers to interface with onboard processors and external clients, and communicates with the Pixhawk 4 flight controller using **pymavlink** over MAVLink.

This flexible architecture allows **easy debugging, development, and testing** across subsystems. The software is designed to run autonomously in real time while maintaining compatibility with manual override and live telemetry through the ASKET GUI.

3.2 Software Design

The software stack follows a **modular design** with clear separation between:

- **Perception** modules (vision pipelines, YOLO object detection, LiDAR and sonar processing)
- **State estimation** (GPS, IMU, Kalman filtering for position, velocity, orientation)
- **Navigation** (A* and Dijkstra-based path planning, reactive behaviors for obstacle avoidance, docking logic)
- **Control** (motor commands sent via MAVLink to Pixhawk)
- **Communication** (TCP/IP protocols for telemetry, commands, video, LiDAR streaming)
- **Human-machine interface** (React-based GUI)

Each module is developed in **Python**, and operates independently but communicates over the shared TCP/IP layers. Control data, sensor streams, and mission state are passed between modules using structured JSON or binary protocols.

This architecture provides **robust performance, flexibility, and ease of integration** for ASKET's mission needs in the Njord Challenge.

3.3 Implementation

3.3.1 Robot Communication Description

ASKET uses a **layered TCP/IP protocol** for full-duplex communication between controller, processors, and clients (Ethernet or Wi-Fi).

TCP ensures reliable transport; a lightweight protocol defines structured message exchange in a **client-server model**, with ASKET acting as TCP server. Dedicated sockets handle:

- **Control & Telemetry**
- **Video Stream**
- **LiDAR Stream** *(Figure 8)*

To manage TCP's stream-oriented flow, messages use explicit framing: start tag, header length, payload length, and data.

Control/telemetry messages include a sync header, message type, payload size, data (typically JSON), and checksum, with binary encoding used for real-time control. Video messages (tagged 'VIDF') include metadata (timestamp, frame ID, resolution, frame rate, format) and compressed image data. LiDAR messages (tagged 'LIDR') transmit point cloud data with relevant scan details *(Figure 9)*.

The system enables **motion control, task commands, parameter updates**, and telemetry (system status, sensor streams, error logs). Safety is reinforced through **heartbeat signals, timeouts, auto-reconnect**, and watchdog timers to stop the ASV if communications fail.

3.3.2 Obstacle Detection and Data Analysis

To ensure safe autonomous navigation, ASKET employs a dual-pipeline strategy for obstacle detection and avoidance. The system integrates **traditional computer vision techniques** and **deep learning-based object segmentation**, enabling both interpretability and robust performance under varying environmental conditions. The two approaches — **color/feature-based detection** and **YOLO-based object**

detection—are evaluated in parallel to determine the most effective solution for specific competition scenarios.

For red/green buoys, the system isolates colored regions and estimates proximity via pixel area.

For **cardinal buoys** (complex black/yellow patterns), detection combines **SIFT feature extraction** and YOLOv8 object detection.

YOLOv8 achieves up to **250 m detection range** at 1000px resolution, processing at **30 ms per frame**. The system maintains stable performance even under challenging conditions such as **sun glare and surface reflections**. Onboard processing is efficient, with each frame analyzed in approximately **30 milliseconds**.

Models were trained on a custom dataset using an NVIDIA GTX 1650 GPU (27 hours). Optimal performance was achieved with a **batch size of 8** and an **input resolution of 1000 pixels**. *(Figure 10)*

The outputs from both pipelines feed into a rule-based decision system. Based on the detected buoys and their positions within the camera frame, the ASV executes specific navigation behaviors. *(Figure 11)*

As part of the detection pipeline, **annotated images** are generated to visualize key outputs at various stages, including **color segmentation results** for red and green buoys, **feature detection overlays** for cardinal markers, and **bounding boxes with confidence scores** from YOLO-based object detection. Additionally, the system visualizes **proximity-based filtering** and the resulting **navigation decision outputs**. These visualizations are instrumental for **debugging**, **performance validation**, and real-time interpretation during **field testing**. *(Figure 12)*

3.3.3 Navigation and Localization

ASKET's navigation system fuses **path planning, reactive control**, and robust state estimation for dynamic environments.

Manual Path Planning and Algorithmic Control:

ASKET's initial **path planning** system is a **custom Python software stack** that computes the **shortest route** through provided **GPS waypoints**. It uses **Dijkstra** and **A*** algorithms with **marine navigation heuristics** to generate **heading** and **speed profiles**. **Motor control** is handled via **PWM**, continuously updated with live **GPS**, **IMU**, and **LiDAR** data.

During missions, the ASV processes a **CSV list of waypoints** and dynamically adjusts **propeller speeds** to minimize **cross-track error**:

Heading Error = Target Bearing – Current Heading

To handle obstacle-rich scenarios—such as buoys, gates, or walls—the system switches to reactive behaviors, using camera and LiDAR inputs to adjust speed and heading or stop entirely to prevent collisions..

Integration with QGroundControl (QGC):

ASKET integrates **QGroundControl (QGC)** with the Pixhawk 4 for mission planning and execution. Waypoints can be uploaded via QGC or MAVLink API, and the ASV autonomously follows the path. QGC provides live telemetry—trajectory, mode, battery, camera, and LiDAR—and supports **manual override (RC mode)** for safety.

Telemetry is logged for post-mission analysis. The system combines **custom pathfinding with QGC execution**, ensuring flexibility and robustness. Vision-based navigation can be re-enabled after manual intervention for complex scenarios. *(Figure 13)*

Object localization fuses YOLO detection with LiDAR for spatial accuracy.

Through testing, four key navigation actions were identified: **going to waypoints, detecting and passing buoys, avoiding walls and obstacles, and docking**. These actions were used to address the project's four main challenges. **Maneuvering** and **path finding** relied on waypoint following and buoy detection. **Collision avoidance** combined waypoint navigation, buoy detection, and obstacle avoidance. **Docking** required the full set of actions—waypoint following, buoy detection, obstacle avoidance, and precise docking maneuvers—to complete this advanced task.

ASKET's **localization** fuses **GPS** (latitude, longitude, altitude) with **IMU orientation** (roll, pitch, yaw) for robust **position** and **pose estimation**, even under **degraded GPS conditions**. By combining these data sources, the localization system remains accurate and robust, even under conditions of GPS degradation caused by reflections on the water surface or temporary satellite signal loss. GPS and IMU data are fused via **Kalman Filters**:

Basic KF: roll, pitch, yaw (single-dimensional IMU streams):

- Predicts state evolution
- Updates with new sensor data
- Balances prediction and measurements

Data Fusion KF: GPS + IMU → position (X, Y), velocity (Vx, Vy), heading (deg).

The system outputs stable state estimates for navigation. Testing showed **significant reduction in GPS drift**, with improved yaw/heading stability *(Figure 14)*.

3.4 GUI: ASKET's Control Interface

The ASKET GUI is a **web-based control and monitoring interface**, built with **React** and **Tailwind CSS**. It enables operators to supervise the ASV in real time, plan missions, monitor system state, and issue manual overrides.

Displayed information includes:

- **Live camera feed** with overlays
- **Latitude, longitude, heading, and course over ground** with a real-time trail
- **Speed over ground (SOG)**
- **Plot of the actual path vs. the ideal GNSS-based route** for performance comparison
- **Distance to the next waypoint**
- **Battery status** (both % and Wh remaining)
- **Status indicator** showing the current ASV mode: *Autonomous*, *Standby*, or *Out of Control*
- **Live telemetry** from onboard sensors

Architecture and Layout:

The GUI is built with **React**, using **React Query** for asynchronous data handling, **Leaflet** for interactive maps, and **OpenCV.js** for processing camera streams. State is managed with **React hooks**, updated via telemetry over TCP/IP. The interface follows a **card-based visual layout** that segments key functions—map, video, telemetry, and controls—into clear zones to guide operator attention. The **left panel** displays the live map with route planning and trail overlay, while the **right panel** shows video feed, system status, controls, and telemetry. The **responsive design** adapts automatically to smaller screens, collapsing into a single column for usability on field laptops or tablets.

The GUI uses **color-coded indicators** (green for normal, yellow for warnings, red for faults), and supports **direct interaction** with map-based waypoint editing and dedicated controls for emergency stop and manual override. It delivers **real-time feedback**, with live camera streams, sensor data, and toast notifications. Advanced functions—such as camera selection and logs—are revealed through **progressive disclosure**, while core safety controls remain always visible.

To ensure responsiveness in mission-critical operations, the GUI uses **canvas-based video rendering**, **localized telemetry updates**, and **modular components** for real-time interaction. It serves as the primary **HMI** for ASKET, providing access to navigation, mission state, sensors, and emergency controls, supporting both **manual and autonomous modes**, and designed for future extensibility *(Figure 15)*.

3.5 Docking Strategy

ASKET's docking process combines **high-precision positioning, real-time perception, and adaptive control** to achieve fully autonomous docking in both structured and cluttered environments. It fuses **GNSS-IMU data, computer vision, proximity sensing, and dynamic path planning** to guide the vessel through each docking phase—from initial approach to final alignment and exit—while adapting to variable berth geometries and obstacles.

GNSS + IMU: provides position and heading (latitude, longitude, yaw) for navigation and tracking.

Camera system: front-mounted camera detects **AR-tags** and obstacles (Otter vessel, structures).

Pixhawk flight controller: executes low-level control of propulsion and steering via MAVLink.

Proximity sensors (Ultrasonic / LiDAR): deliver short-range distance measurements to enhance final docking safety.

Implemented in **Python**, the docking system uses **pymavlink** for Pixhawk communication, transmitting velocity and yaw commands and switching between manual and autonomous modes.

OpenCV powers the vision pipeline for **AR-tag detection** and obstacle recognition. **Matplotlib** provides real-time visualization for debugging and testing.

A custom path planner governs **velocity smoothing, target tracking, and adaptive control**, enabling responsive and precise docking in dynamic environments.

Docking Mission Logic:

The docking sequence begins by navigating to a pre-docking point (~2 m behind berth entrance), maintaining a 2 m safety buffer around poles.

The ASV advances while dynamically refining alignment based on proximity and bearing.

At the entrance midpoint, the ASV performs a **controlled reverse** (~2 m), adjusts as needed, and proceeds with a final approach—using **AR-tag alignment** where available—for a precise, low-impact docking maneuver.

After holding position for 5 s to simulate secured docking, the ASV reverses and returns to a GPS exit point to resume the mission.

GPS, vision, and proximity data are fused to guide final maneuvers. Vision-based control is triggered inside a proximity threshold where GPS alone is insufficient. Velocity and yaw are smoothed continuously for safe, accurate motion. The system adapts in real time to changing geometries and obstacles. *(Figure 16)*

4.0 Innovative Aspect

One of ASKET ASV's key innovations is its **fully 3D-printed modular hull**, designed specifically for ASVs and fabricated using a **glass fiber-reinforced PETG** on a large-format robotic pellet extruder. Unlike conventional fiberglass, metal, or molded plastic hulls, this approach allows **rapid prototyping, customizable design, and strong yet lightweight structures**—with minimal tooling cost.

- **Fast iteration:** hull production in 12 h
- **Optimized hydrodynamics** tuned digitally

- **Modular & serviceable:** independently printable hull & lid
- **Lightweight & efficient:** 9–10 mm walls reduce inertial load and improve thrust performance

Technical Implementation:

- **Material:** PETG + glass fiber (hull), PLA (electronics lid)
- **Fabrication:** ABB + Ziknes Z-Pellet One
- **Design:** solid bow, sealing flanges, interlocks, **XTC-3D** coating

Fiberglass requires tooling & curing; sheet metal is heavy/corrosive; commercial ASV hulls are proprietary and hard to modify. ASKET's printed hull is **low-cost, flexible, student-friendly, and repeatable**.

Tests showed excellent lateral stability, minimal pitch/yaw drift, and efficient, cavitation-free propulsion. The modular design was validated through multiple assembly cycles. Two full-scale hulls were printed and tested, proving the approach's viability for competition use.

5.0 Testing

5.1 Vessel Testing

Vessel development started with **~10 scale models (1:5)**, printed and tested to evaluate **buoyancy, hull geometry, and stability** under various load conditions. These models informed iterative refinements to the final hull design.

The **first full-scale prototype** was printed in three parts (nose, middle, stern) due to printing issues. Though the parts were glued and sealed, **leaks appeared during initial water tests**. The design was then reworked to allow **single-piece printing**, resolving the leakage problem.

Without access to a test pool, the team conducted all water trials in **open sea conditions**—bringing ASKET to **Barcelona's beaches** for real-world testing in **saltwater**, which also provided valuable data on durability, sealing, and system resilience.

Testing results:

- **Hull stability and buoyancy** matched expectations
- **Leaking resolved** with one-piece hull
- **Durability in saltwater** confirmed under field conditions
- **Cruising speeds (1–5 knots)** and **battery runtime (30–35 min)** achieved
- No structural or propulsion failures observed

5.2 Hardware Testing

Hardware testing followed a **bottom-up approach**: each component was first validated individually with **custom test scripts**, then gradually integrated into a complete system.

Individual subsystems—propulsion, battery, and sensors—were **bench-tested prior to integration**:

- **Thrusters** were tested across PWM ranges to confirm **smooth thrust response** and correct directionality
- **Battery** testing confirmed **stable capacity** (8000 mAh) and acceptable voltage sag under sustained load
- **Sensors** (LiDAR, GPS, IMU, sonar) were verified for **consistent connectivity, range accuracy, and refresh rates** across multiple test sessions

Hardware was then integrated step-by-step to form a complete hardware stack. This incremental method helped uncover early issues and improved overall system reliability.

During full system testing, **CPU thermal performance** emerged as a limitation: under high load, the Raspberry Pi 5 CPU experienced **thermal throttling**, which reduced camera frame rates and delayed visual feedback.

Identified improvement actions:

- Add **active cooling** or redesign sealed compartment for better passive heat dissipation
- Upgrade **cabling and connectors** for field robustness

5.3 Software Testing

ASKET's **computer vision system** was extensively tested on land and in open water. Initial testing of **color-based buoy detection** showed challenges in green buoy recognition due to water color and lighting conditions. After tuning—using adaptive thresholding, Gaussian blur, and HSV color space adjustments—detection performance improved significantly.

Red buoy detection worked reliably in good light but degraded in overcast conditions, as red wavelengths are absorbed first in water.

SIFT feature-based detection was effective for **cardinal buoys**, with some tuning needed for precise shape recognition.

Full-system water trials confirmed:

- The ASV correctly adjusted course in response to detected buoys
- Motors followed control commands from vision-based navigation
- The primary software bottleneck was **reaction time** due to low video frame rate under thermal throttling conditions

Identified software optimizations:

- Remove unnecessary image overlay rendering
- Streamline image pipeline to reduce latency
- Upgrade thermal management to sustain target frame rate (>10 FPS)

Other tested software modules:

- **TCP/IP communication stack**: verified reliable telemetry streaming and command transmission, with heartbeat and timeout safety features
- **Waypoint navigation**: tested with uploaded GPS waypoints and dynamic adjustment to minimize cross-track error
- **GUI integration**: validated live camera feed, telemetry updates, and manual override controls during field sessions

Overall, all core systems—vision, navigation, control, GUI—performed as designed. Identified software improvements included **optimizing vision processing loops** and addressing **thermal management** to sustain higher frame rates and faster reactions in autonomous mode.

5.4 Integrated Testing

Integrated field trials combined path planning, vision-based object detection, reactive control, and docking behaviors. The ASV successfully completed:

- Autonomous waypoint navigation
- Dynamic buoy detection and avoidance
- Manual override via RC controller
- Partial docking sequences with AR-tag alignment

Limitations included:

- Inconsistent docking accuracy under poor lighting
- Occasional latency in obstacle response due to CPU heat buildup
- Minor GPS drift in areas of signal reflection

Future work will focus on:

- Improving **compute cooling** to enable sustained vision performance
- Further tuning of **docking behaviors** under variable conditions
- Testing in more complex obstacle fields for competition-readiness

6.0 Contributions

The development of the ASKET ASV was a fully **collaborative, interdisciplinary effort** by the NODE team at IAAC. The project was completed for the **Njord 2025 Physical Autonomous Challenge**, with each team member contributing to key technical areas across **hardware, software, and system integration**.

Team Structure and Roles:

System & Project Management

- **Aleksandra Kraeva** – Team Captain / Mission Manager: Led overall project planning, testing coordination, and system-level mission strategy
- **Mert Özkılıç** – Principal Engineer / Project Manager: Led technical integration across mechanical, electrical, and software systems

Hardware & Electronics

- **Auxence Daillen** – *Hardware Co-Leader*: Developed and implemented the hardware system for the autonomous marine vehicle. Led the installation and wiring of electronics, designed the onboard power systems, and managed the integration of propulsion components and internal structural layout.
- **Maximilian Severin Becht** – *Hardware Co-Leader*: Developed and implemented the hardware system for the autonomous marine vehicle. Led the installation and wiring of electronics, designed the onboard power systems, and managed the integration of propulsion components and internal structural layout.

Software Development

- **Salvador Cantuarias Brañes** – *Software Leader*: Led development of autonomy software, vision of the ASV and Software-Hardware integration.
- **Samarth Akshaykumar Pachchigar** – *GNC Leader*: Led development of the control, navigation, localisation and guidance algorithms. Established Robotic communication logic.
- **Arun Prasad** – *Robotic Communication Developer*: Implemented communication protocols.
- **Geetham Pasumarty** – *Robotic Communication & GUI Developer*: Implemented robotic communication stack and developed GUI for real-time control.
- **Chanakan Chormai** – *Localisation Enginee*r: Developed mapping and localization system using LiDAR, cameras, IMU, and GPS fusion
- **Eleni Papakosta** – *GNC Engineer*: Focused on navigation algorithms, docking maneuvers, motion control, and stabilization

Design & Fabrication

- **Yuvraj Ajay Shirke** – *Design Leader*: Led structural hull design, coordinated 3D printing workflow.
- **Gautam Vohra** – *Computational Designer*: Designed final hull geometry for optimized hydrodynamics and printability; led fabrication of 1:5 scale prototypes and coordination of the team.
- **Brian Woodtli Bahena** – *Fabrication Expert*: Designed and integrated electronics lid; ensured hardware compatibility
- **Nitsan Mor** – *Fabrication Exper*t: Led assembly and fabrication of two full-scale hull prototypes, including cable routing and hardware mounting
- **Naji Altala** – *Communications Coordinator*: Managed graphic documentation, social media, and external communications

External Support and Acknowledgments

Marita Georganta (Space Software Engineer): Mentor and Coordinator of the project. Provided regular review of system design and integration, and assisted with sponsor outreach, team building, and budget management. Contributed material resources through her involvement with Women in Aerospace – Europe.

Huanyu Li (Robotics Researcher / Digital Fabrication Designer): Acted as mentor, providing expertise on localization and robotic communication.

Alex Dubor (IAAC Robotics Lab Director): Acted as project mentor, supporting budget acquisition, communications, and logistics.

Ziknes (Large-scale 3D printing company): Printed ASKET's full-scale hull on their Z-Pellet One machine and generously contributed €900 to cover the required printing deposit.

Women in Aerospace Europe - Sponsored hardware and software investments for the project with €350

IAAC Robotics Lab: Provided RPLIDAR A2M8, Raspberry Pi 5, lab space for development, and a project budget of €500 to purchase key components for the ASV hardware stack

External Reference (Architecture and design office, Barcelona): Supported the project by allowing the team to use their workshop to print small-scale parts and 1:5 prototypes of the ASKET hull.

Golden Sponsor

Silver Sponsor

Bronze Sponsor

Support

Figure 1 - ASKET ASV - photo from Bogatell Beach, Barcelona

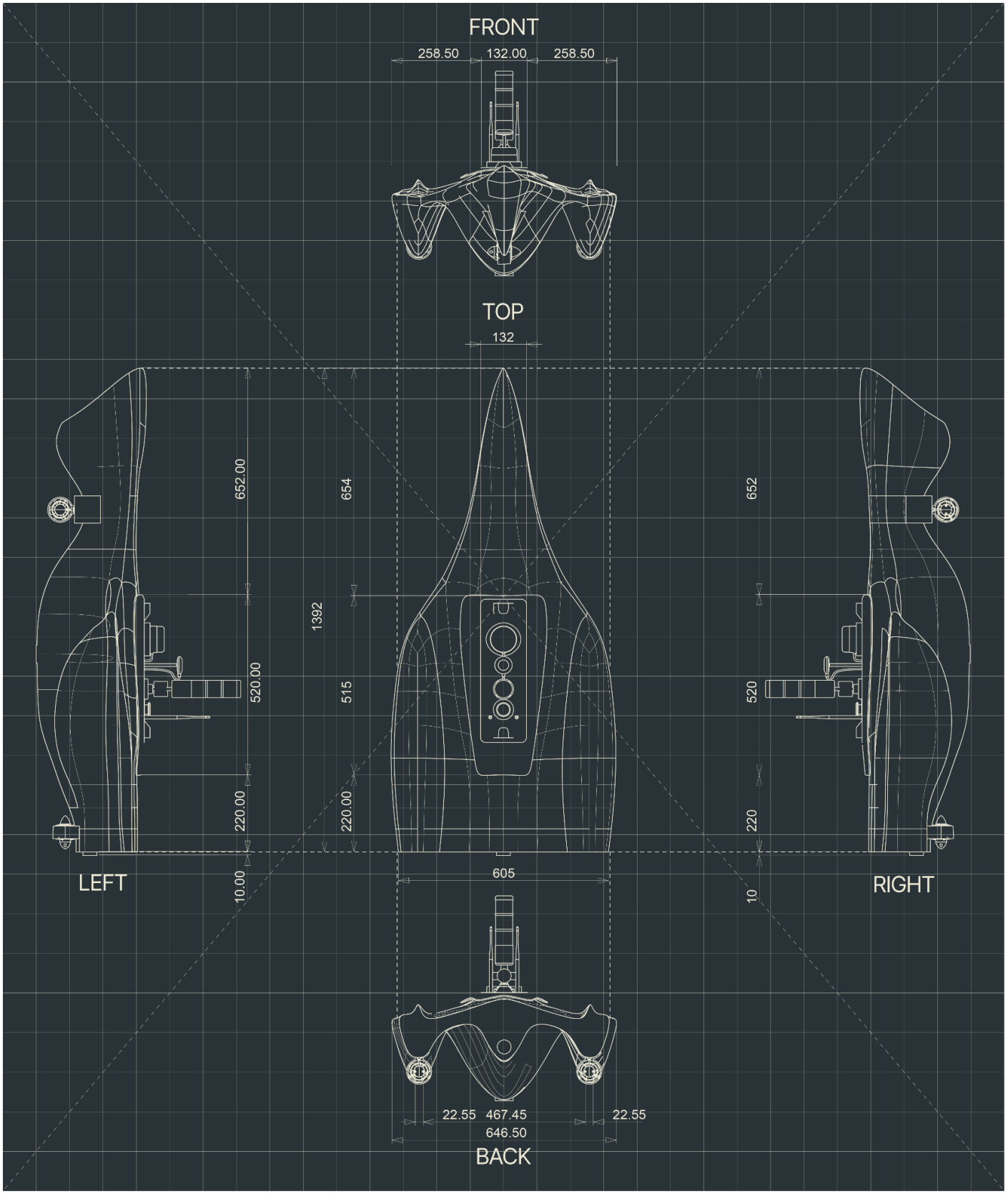


Figure 2 - ASKET ASV Blueprint - diagram

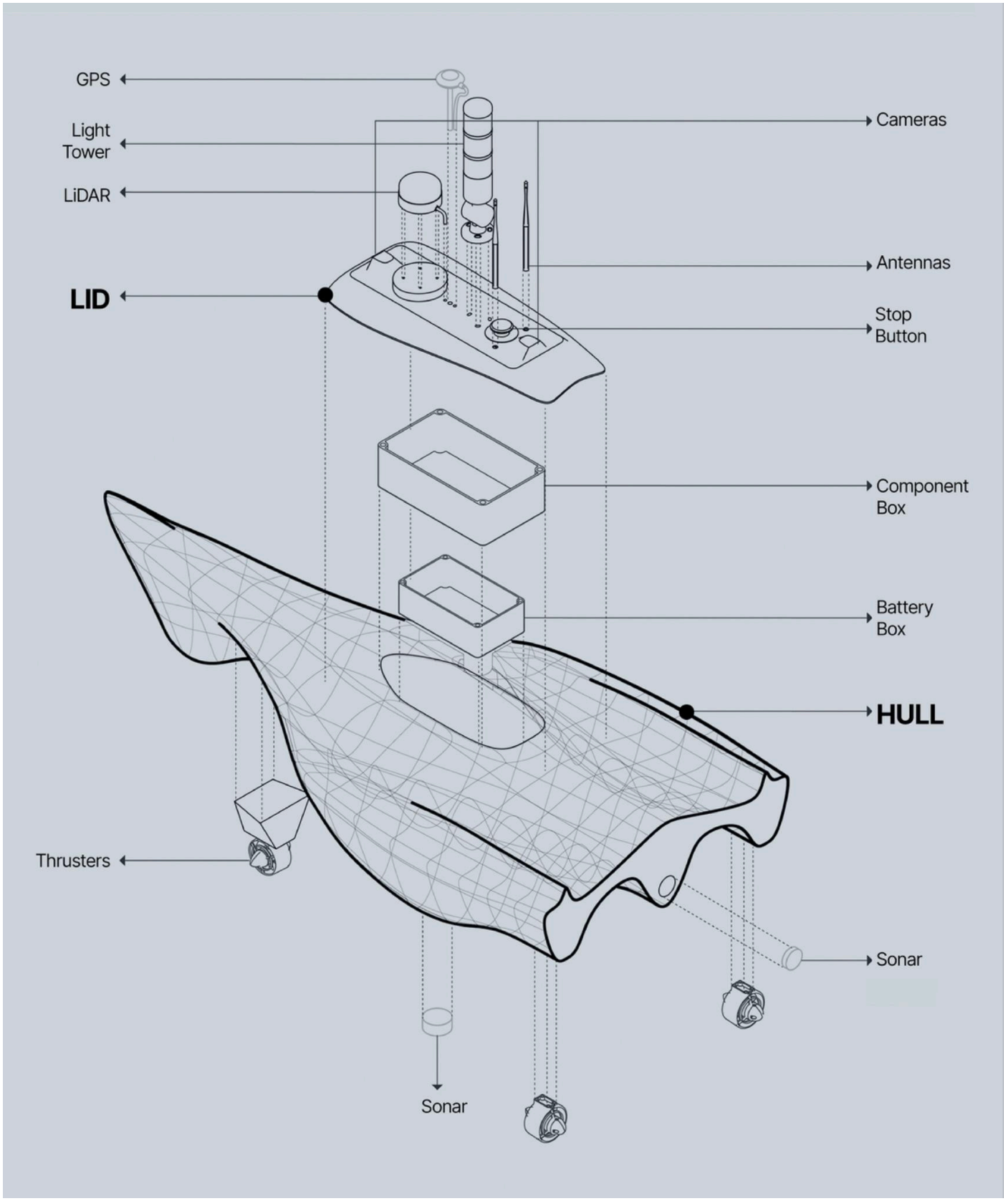


Figure 3 - ASKET ASV - Exploded Axonometric View showcasing main elements - diagram

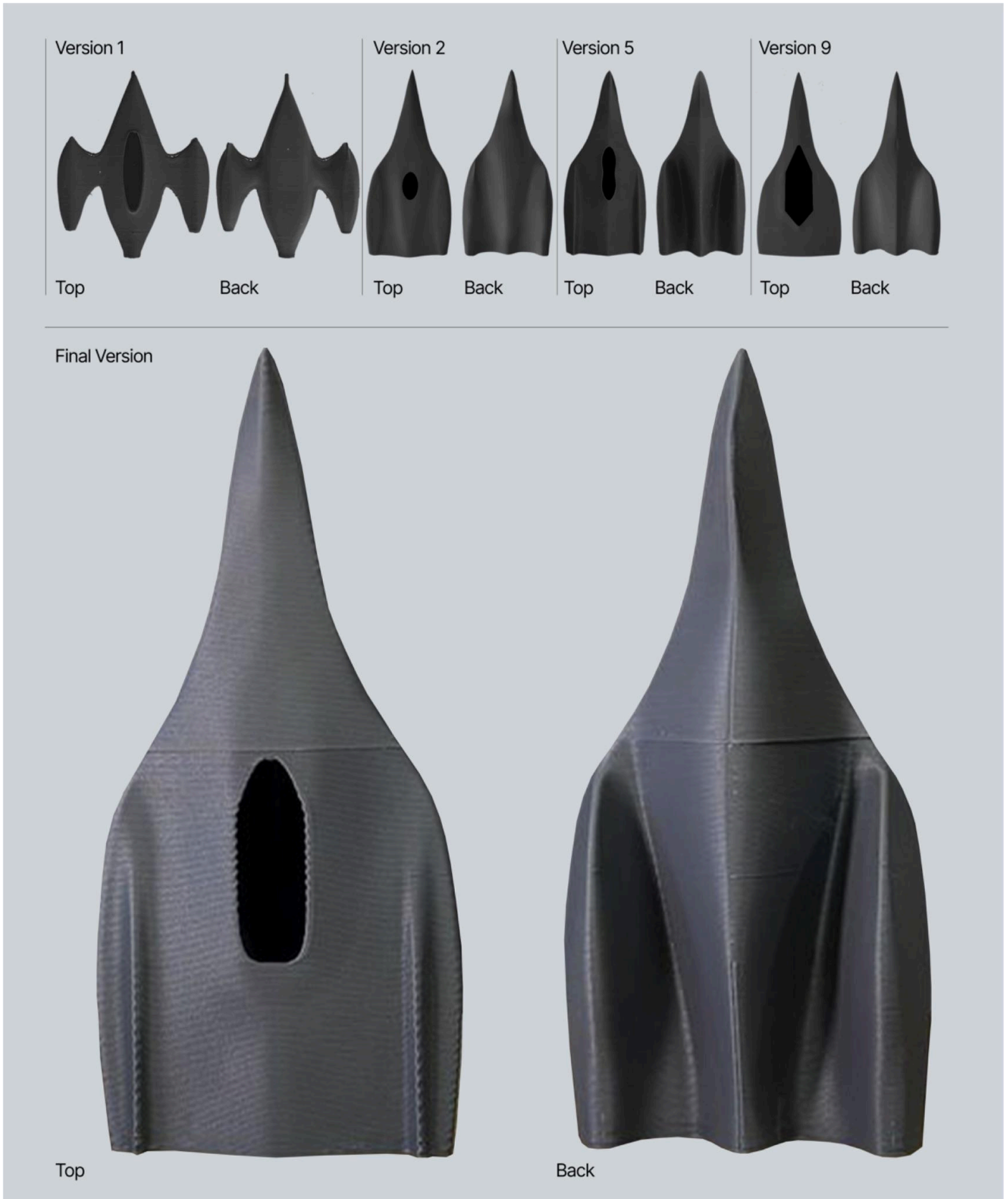


Figure 4 - ASKET ASV - 3D printed prototypes - photos made throughout the year 2024-2025

Figure 5 - ASKET ASV - 3D printing fabrication process - photos from Ziknes factory

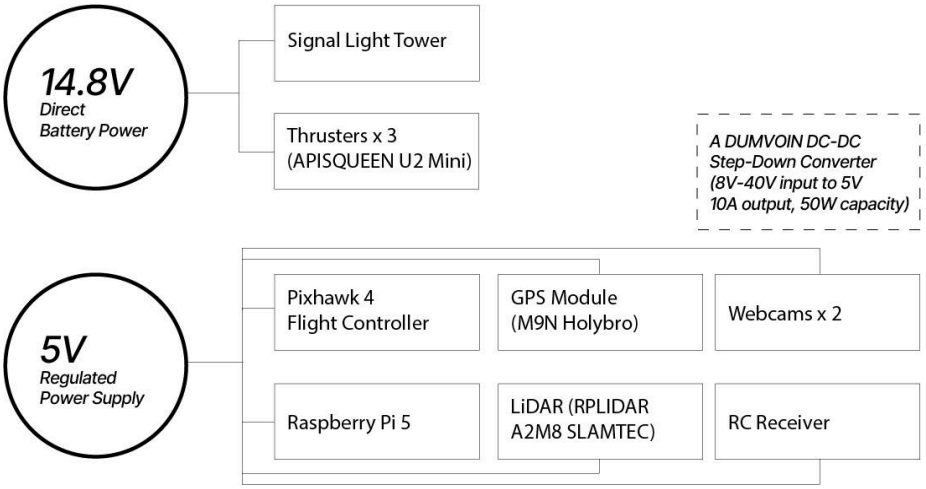


Figure 6 - ASKET ASV - Power Distribution - diagram

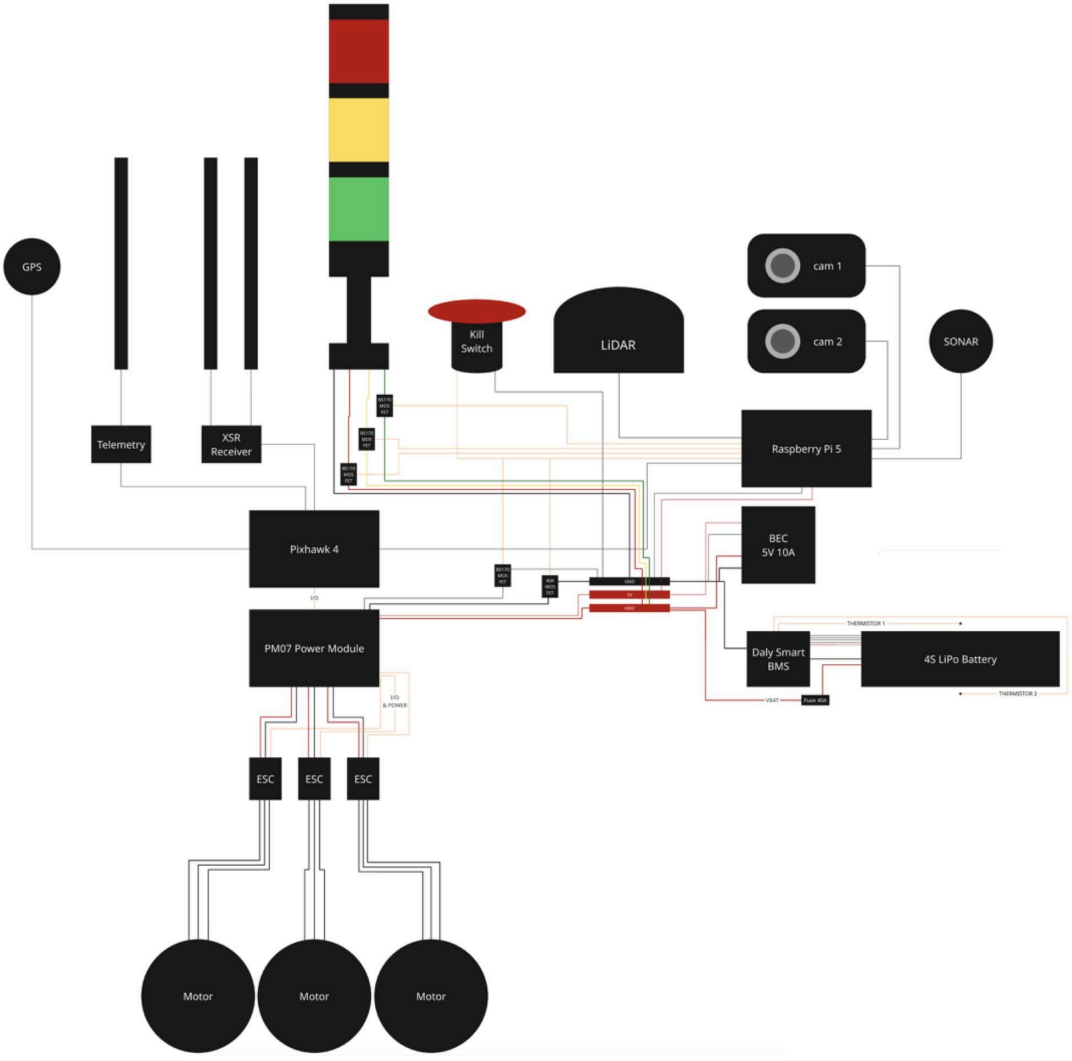


Figure 7 - ASKET ASV - Hardware Implementation - diagram

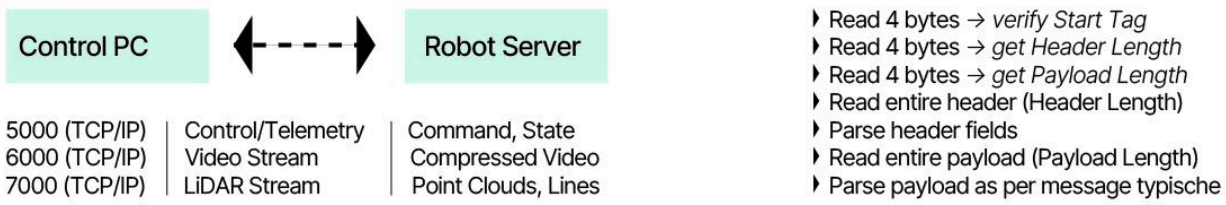


Figure 8 - ASKET ASV - Stream Protocol and Receive Logic Loop - diagram

Control & Telemetry Socket (Port 5000)

Field	Description	Data Type
Header	Synchronization and ID	Fixed Length String
Message Type	Command / Status / Error	Integer
Payload Length	Size of data payload	Integer
Payload	Command or telemetry data	JSON / Binary
Checksum	Error detection	Integer

Camera Stream Protocol (Port 6000)

Field	Description	Data Type
Start Tag	Fixed string identifier (VIDF)	Fixed 4-byte string
Timestamp	Epoch time or system clock	Float
Frame ID	Sequential frame number	Integer
Resolution X	Image width in pixels	Integer
Resolution Y	Image height in pixels	Integer
Frame Rate	FPS of stream	Float
Format	Image encoding format	String
Compression	Optional field	Float
Payload Length	Size of Frame Data	Integer
Frame Data	Compressed image data	Binary blob

Example:
Start Tag: 'VIDF'
Timestamp: 1720665600.270 # Unix Epoch time
Frame ID: 35420
Resolution X: 1920
Resolution Y: 1080
Frame Rate: 30.0
Format: 'H.264'
Compression: 0.85 # Optional: compression ratio if used
Payload Length: 204800 # Size of compressed frame in bytes
Frame Data:
b'\x89PNG\r\n\x1a\n\x00\x00...'
 # (Binary JPEG/H.264 compressed frame data)

LiDAR Stream Protocol (Port 7000)

Field	Description	Data Type
Start Tag	Fixed string identifier (LIDR)	Fixed 4-byte string
Timestamp	Epoch time or system clock	Float
Frame ID	Sequential scan/frame number	Integer
Scan Type	2D / 3D / Full Sweep	Enum (Integer)
Resolution	Angular resolution (deg)	Float
Range Min	Minimum measurable distance	Float
Range Max	Maximum measurable distance	Float
Point Format	Format of point data	String (XYZI, XYZRGB, etc.)
Payload Length	Size of Point Data field	Integer
Point Data	Binary point cloud data	Binary Array (e.g. float32[])

Example:
Start Tag: 'LIDR'
Timestamp: 1720665600.235 # Unix Epoch time
Frame ID: 1258
Scan Type: 3 # (3D Scan for example)
Resolution: 0.1 # Angular resolution in degrees
Range Min: 0.5 # Minimum scan distance (meters)
Range Max: 60.0 # Maximum scan distance (meters)
Point Format: 'XYZI' # X, Y, Z, Intensity
Payload Length: 131072 # Size of binary point data (bytes)
Point Data:
[(1.234, 5.678, 3.210, 0.98),
(1.111, 5.555, 3.333, 0.90) ... # Thousands of points in binary]

Figure 9 - ASKET ASV - Control, Video, and LiDAR Protocols - table

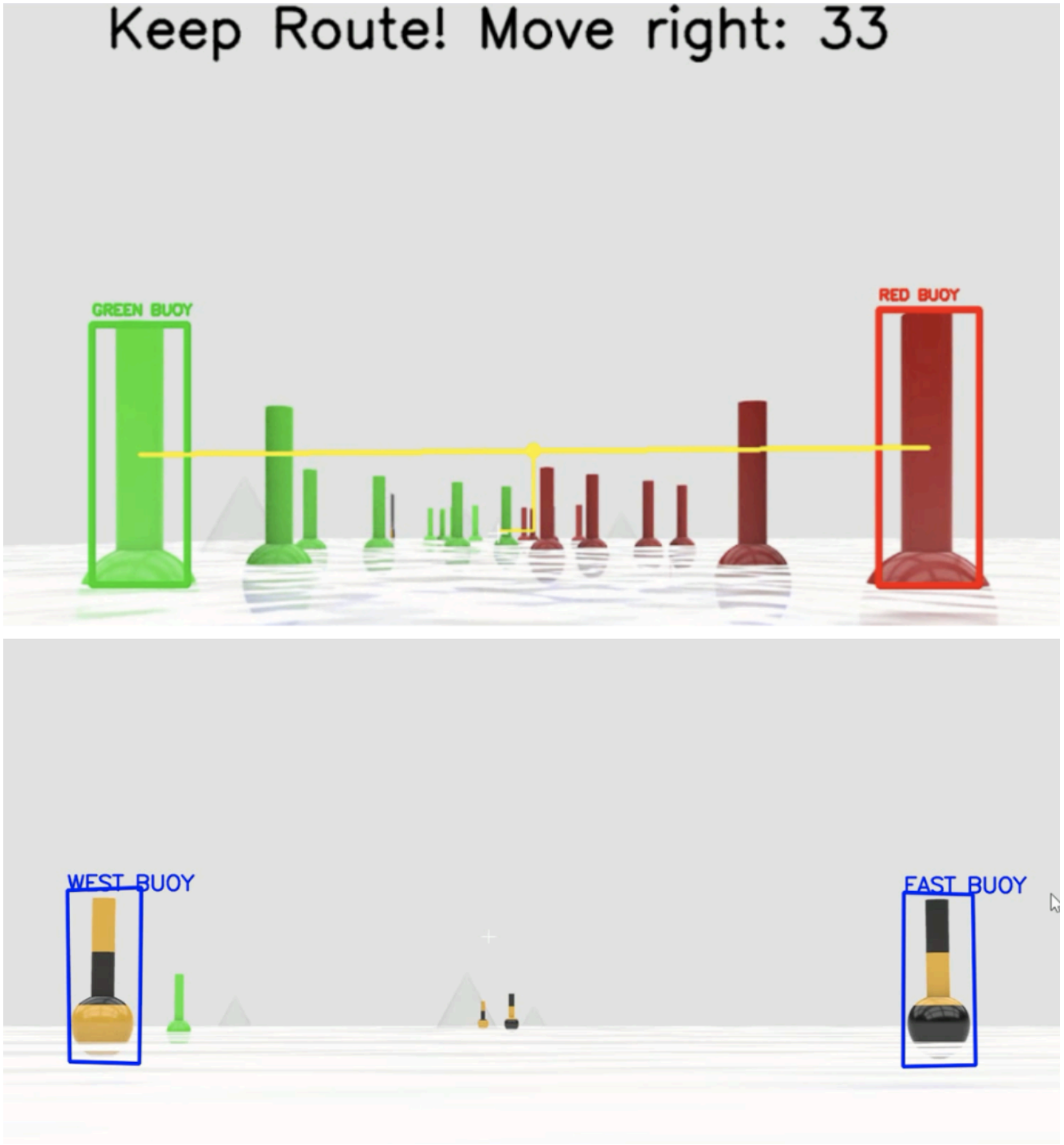


Figure 10 - ASKET ASV - Object Detection - screenshot

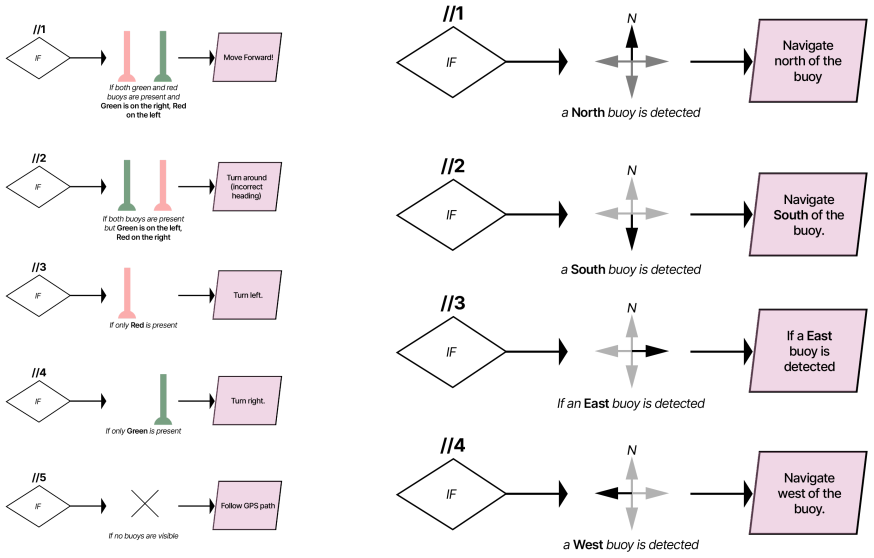


Figure 11 - ASKET ASV - Rules for Object Detection and Action - diagram

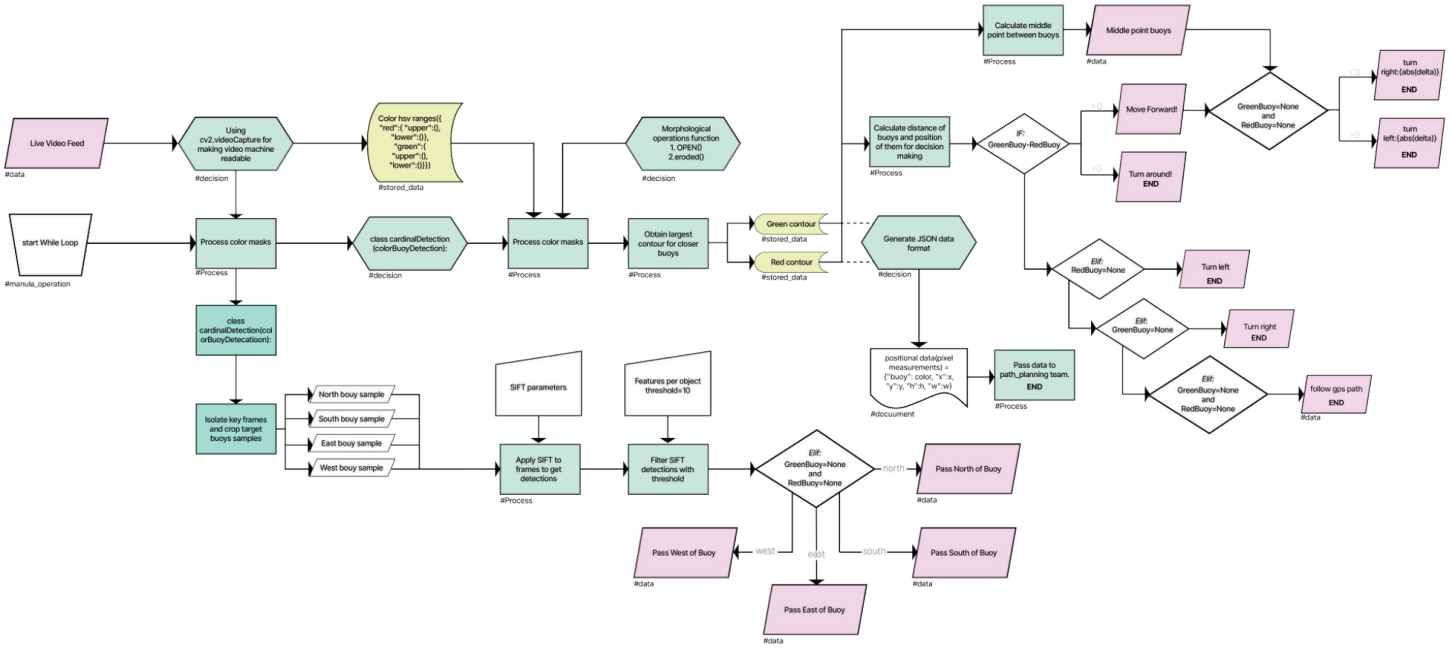


Figure 12 - ASKET ASV - Object Detection Pseudocode - diagram

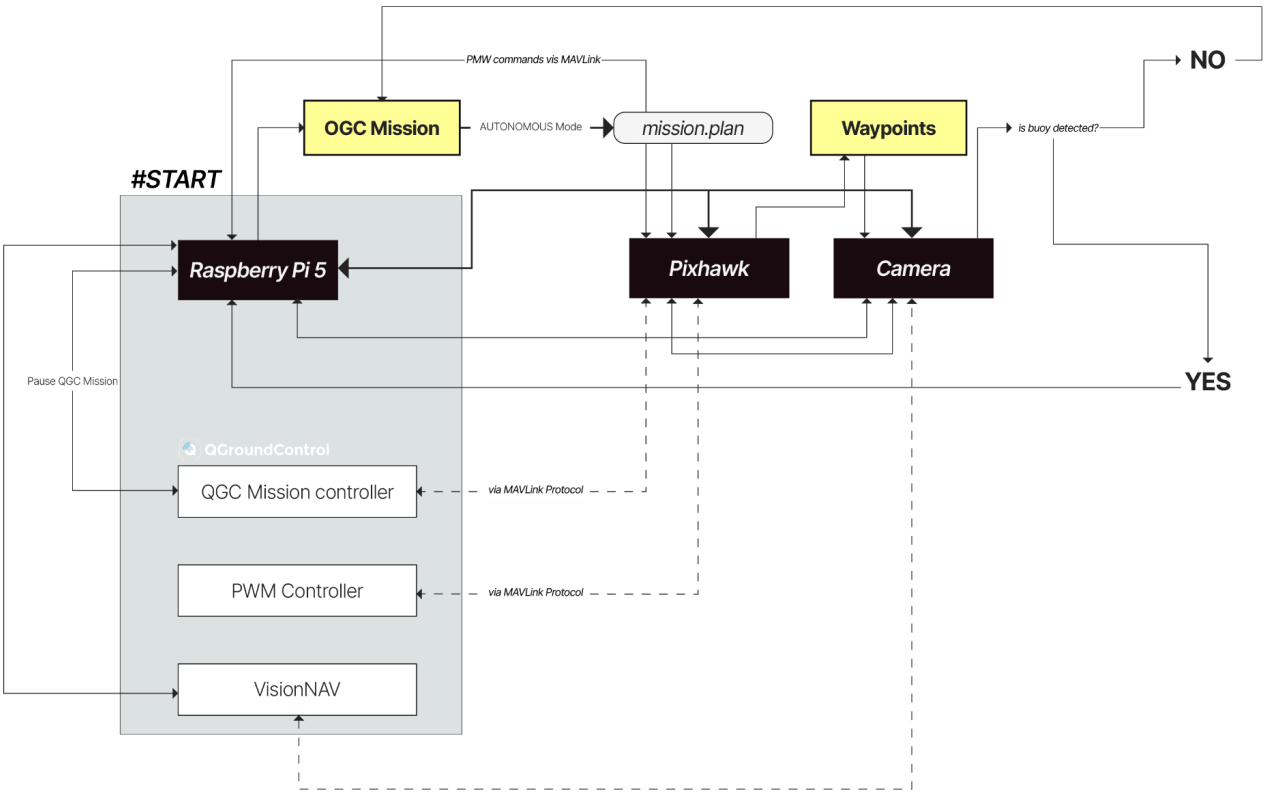


Figure 13 - ASKET ASV - Path Planning and Manual Override decision tree - diagram

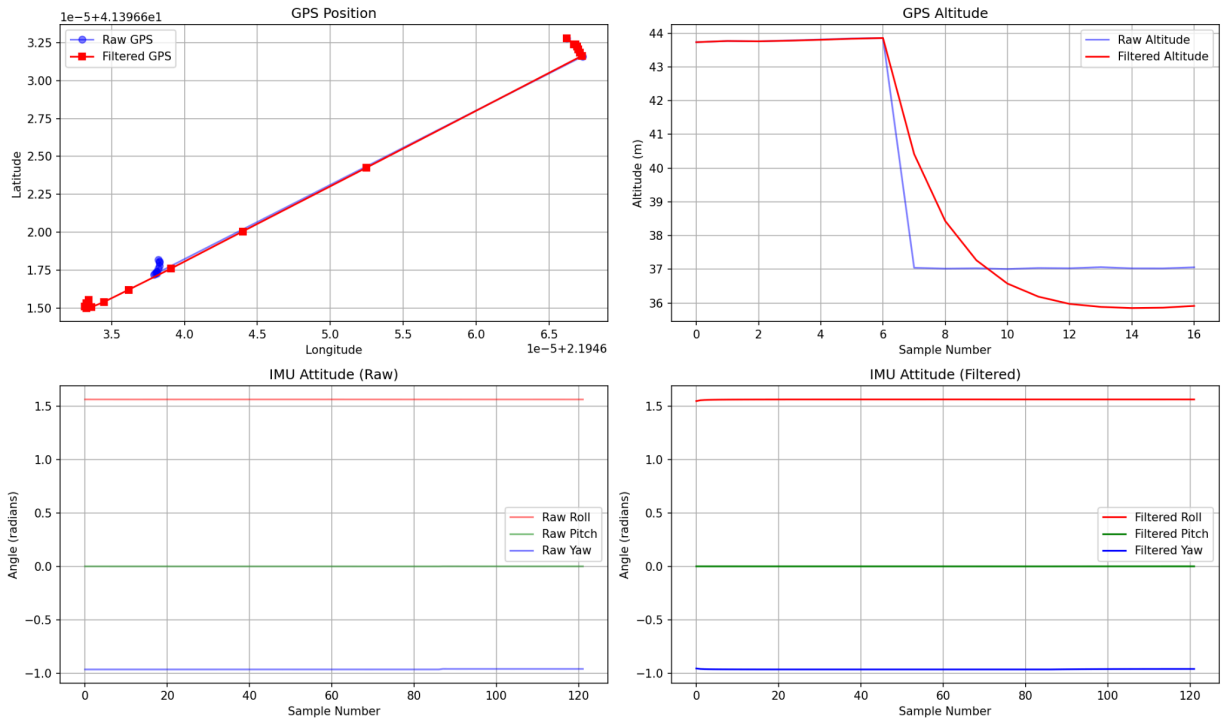


Figure 14 - ASKET ASV - Kalman Filter results - diagram

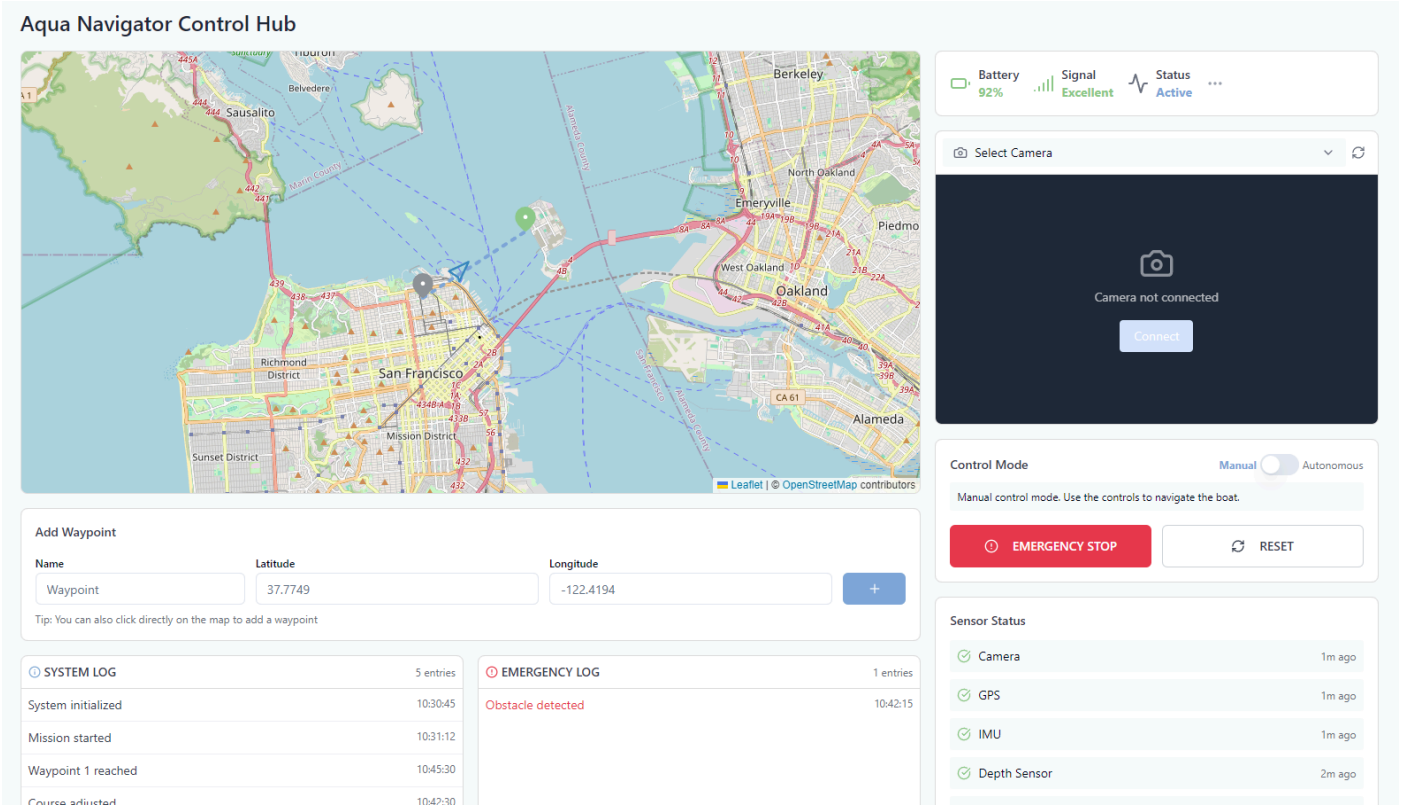


Figure 15 - ASKET ASV - GUI Layout - screenshot

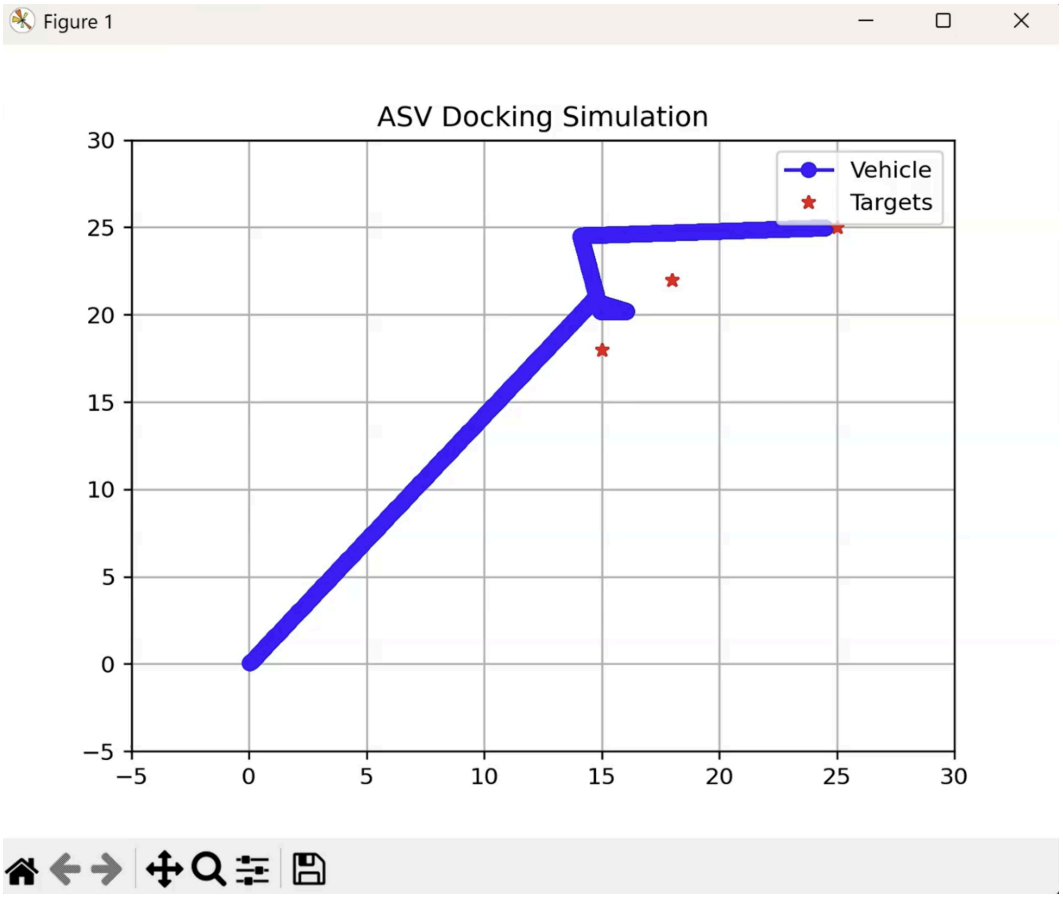


Figure 16 - ASKET ASV - Docking Simulation - screenshot