ORCA: Ocean Robot Cleaning Autonomous

rca

Prepared by Tam Le, Om Goyal, Zain Ali, Rohan Kakarlapudi

Spring 2025

List of Figures

List of Tables

Table of Contents

I Project Description

1 Project Overview

The ORCA (Ocean Robot Cleaner Autonomous) system is a modular, surface-floating robot platform designed to autonomously collect floating debris from bodies of water while allowing manual override control when needed. Each ORCA unit comprises:

- **Mechanical Platform**: A pontoon-style hull for buoyancy and stability, with an onboard stepper-driven conveyor mechanism for debris intake.
- **Electronics & Firmware**: An ESP32 microcontroller running custom firmware to orchestrate motor control (using repurposed fish aquarium motors), sensor integration (GPS, IMU, ultrasonic proximity), and telemetry streaming.
- **Power System**: Rechargeable LiPo battery pack supporting several hours of continuous operation, with future provisions for solar-panel charging.
- **Communications & Control**: Secure Wi-Fi mesh networking to relay real-time telemetry (position, battery level, debris capacity, environmental sensor readings) to a React-based web dashboard. From the dashboard, users can define geofenced mission areas, adjust navigation parameters, and perform manual recalls or emergency stop commands.
- **Autonomous Navigation**: Onboard route planning leverages weather and current data APIs to optimize coverage while conserving energy. Upon mission completion or critical alerts (e.g., low battery or full debris compartment), ORCA autonomously returns to a designated base for debris offload and maintenance.

- **Frontend Dashboard:** The frontend was developed in React.js, with modular components like a sidebar for controlling the robot, a telemetry panel with real-time signal, garbage, and power updates, a canvas-based map for robot path

visualization, and dynamic graphs for historical data tracking. It communicates directly with the robot's onboard system via HTTP. This removes the need for a traditional backend and reduces architectural complexity, which allows fast integration in both the frontend and backend, as well as gives users real-time control and feedback.

2 Project Domain

The ORCA system operates at the intersection of marine conservation, robotics, and environmental data analytics. Key aspects of its domain include:

- **Environmental Impact**: Mitigating the estimated 14 million tons of plastic and debris entering oceans annually by providing scalable, surface-level cleanup capabilities.
- **Stakeholders**:
    - **Environmental Agencies** require efficient, data-driven cleanup missions.
    - **Local Communities** are seeking cleaner waterways to boost public health, recreation, and coastal economies.
    - **Researchers** are analyzing long-term pollution trends using high-resolution debris and water-quality data.
- **Operational Context**: Deployment in rivers, lakes, and coastal areas subject to surface debris accumulation, variable weather and wave conditions, and the need for remote monitoring and maintenance scheduling.

3 Relationship to Other Documents

This report builds upon and references several project artefacts:

- **Project Design & Requirements**: The Coding Scenario document outlines the minimum viable product features and usage scenarios, forming the basis for development and testing.
- **Usage Scenarios**: The "Robot Setup and Usage" and "Advanced Features and Improvements" scenario PDFs provide detailed activity and sequence diagrams for MVP and next-phase functionality.

- **Presentation Deck**: The ORCA Presentation (Group 5) defines core concepts, system architecture, and stakeholder benefits, which inform the project overview and domain discussion.
  All referenced documents are listed in Section VIII: References.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

ORCA - defined as the name of the project. It stands for Ocean Robot Cleaning Autonomous

Pontoon - defined as the floating tubes for boats, electronics, and things that are inside here

Motor pump engine - all refer to the propulsion system of our mechanism, which relies on a Brushless DC motor powered, fishtank pumps. There are a total of 4 on our boat

Trash Collector - this refers to anything that collects trash floating on the water; there were several iterations of this that look and operate very differently.

PLA - plastic material used for 3d printing, degrades over time in water becomes brittle when exposed to water or high humidity for long periods of time, has a density of 1.24g/cm^3

3d Printing - Additive manufacturing process that uses a robot with preprogrammed path instructions to layer PLA on top of each other to form structures designed in a computer

React: A JavaScript library/frontend for building user interfaces, used in this project to create the modular, component-based frontend dashboard that communicates with the robot

ESP32: A low-cost, low-power microcontroller with built-in Wi-Fi capabilities, used as the robot's main onboard controller for motor and sensor management.

4b UML and Other Notation Used in This Document

We have a Use Case Diagram which starts at one action and then continually points to the next action in a series, and branches out when there are different action paths to take.

4c Data Dictionary for Any Included Models

Data Element	Type	Description
robotID	String	Globally unique identifier for each ORCA unit(e.g., ORCA-A02”).
missionID	String	Unique identifier for the current mission or waypoint sequence
timestamp	ISO 8601 String	UTC date/time of the telemetry packet(e.g., “2025-04-12 CT 12:45:002”)
latitude	Float (Degrees)	Current GPS latitude of the robot
longitude	Float(degrees)	Current GPS longitude of the robot
battery level	Integer(%)	Remain battery charge as a percentage (0-100)
debrisCount	Integer	Number of individual debris items collected since last offload.
waterTemp	Float (Celsius)	Surface water temperature reading from onboard sensor.
turbidity	Float (NTU)	Turbidity measurement in Nephelometric Turbidity Units.
pH	Float	pH value of surface water.
feofenceStatus	String	“INSIDE”, “BOUNDARY”, or

		“OUTSIDE” based on comparing current position to user-defined geofence.
motorStatus	String	Current state of propulsion motors: “IDLE”, “FORWARD”, “REVERSE”, “ERROR”.
conveyorStatus	String	Status of the debris conveyor: “ON”, “OFF”, “JAM”, or “ERROR”.
cameraImageID	String	Identifier for the latest debris-capture image stored on the server (e.g., “v1.1.0”).
modelPrediction	String	Debris type predicted by onboard TensorFlow Lite model:”PLASTIC”, “ORGANIC”, “METAL”,etc.
firmwareVersion	String	Semantic version string of the running ESO32 firmware (e.g., “v1.1.0”).
errorCode	Integer	Numeric code representing the last fault detected (0 = no error;).
anomayFlag	Boolean	True if the predictive-maintenance ML model has raised an alert for this unit.
routeWaypoints	List<Object>	Array of upcoming waypoints: each object contains{ *latitude: Float, longitude: Float, sequence: Integer*}
heartbeatInterval	Integer(ms)	Configured interval between heartbeat packets in milliseconds

II Project Deliverables

5 First Release

Release Date
February 28, 2025

Core Functionality Delivered

- **Web Application V1**
    - Live map canvas showing the robot's real time position
    - Live status bars for garbage levels, battery power, and signal strength
    - Line graph showing historical trends for garbage levels, battery power, and signal strength
    - A manual control interface where users can send movement commands from the UI
    - User commands are communicated with the robot via Node.js backend over USB-to-MicroUSB serial communication
    - Users can add or remove robots from the interface list
    - Frontend Architecture: React.js and CSS modules
    - Backend: Node.js server with read/writes to serial data over USS
    - Communication: Serial Communication Only

- **Device Firmware v1.0**
    - Debris collection with Arduino Nano-controlled 2 fish-aquarium motors
    - Articulated floating linkages, weighed down and balanced with BB pellets
    - Very Small mechanical chassis
    - Communication over USB Serial
    - Can turn one way or go forward, but tends to flip over

- **Error-Checking & Contingency**:
  - Validation of server instructions
  - Self-diagnostics on comms link

Delivered Artifacts

- UML sequence and activity diagrams for server–device interactions (Figures 2–4)
- Source code repos:
  - **Server** (Node.js + PostgreSQL; React frontend)
  - **Device** (ESP32 C/C++)
- CI/CD pipelines for firmware builds and server containerization
- QA report covering all functional and non-functional requirements

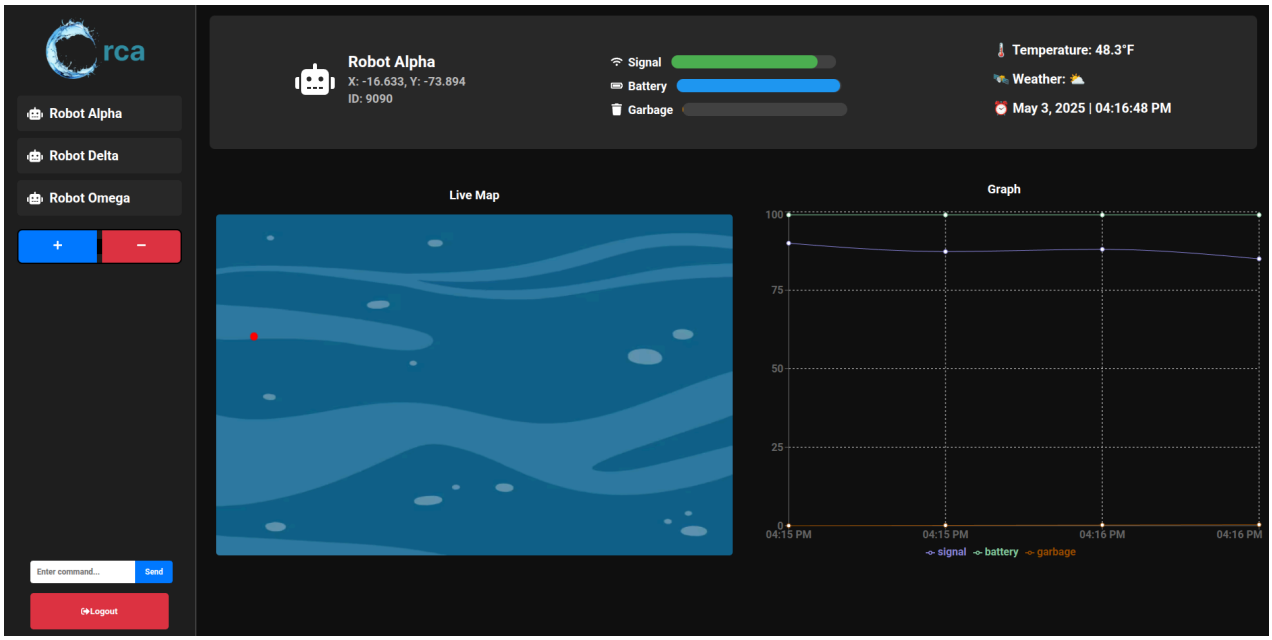

Figure 1 UI Example for first Release

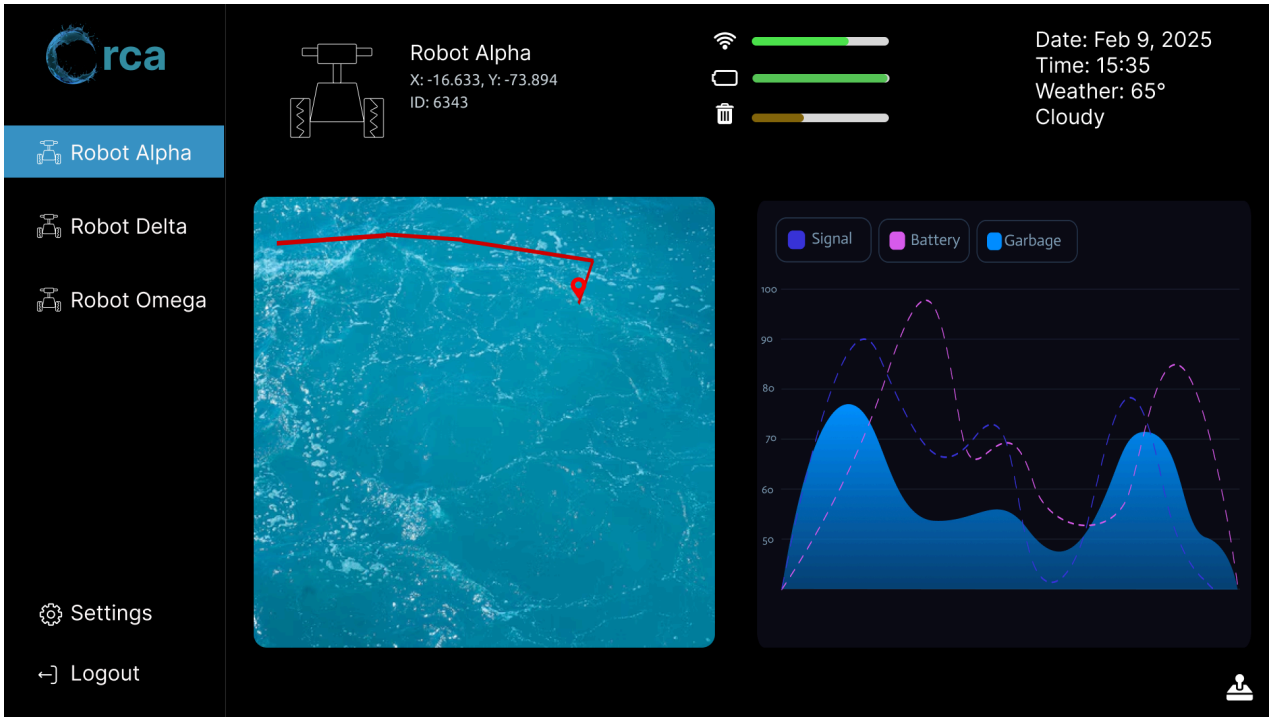

Figure 2 UI Example for first release on Robot Alpha

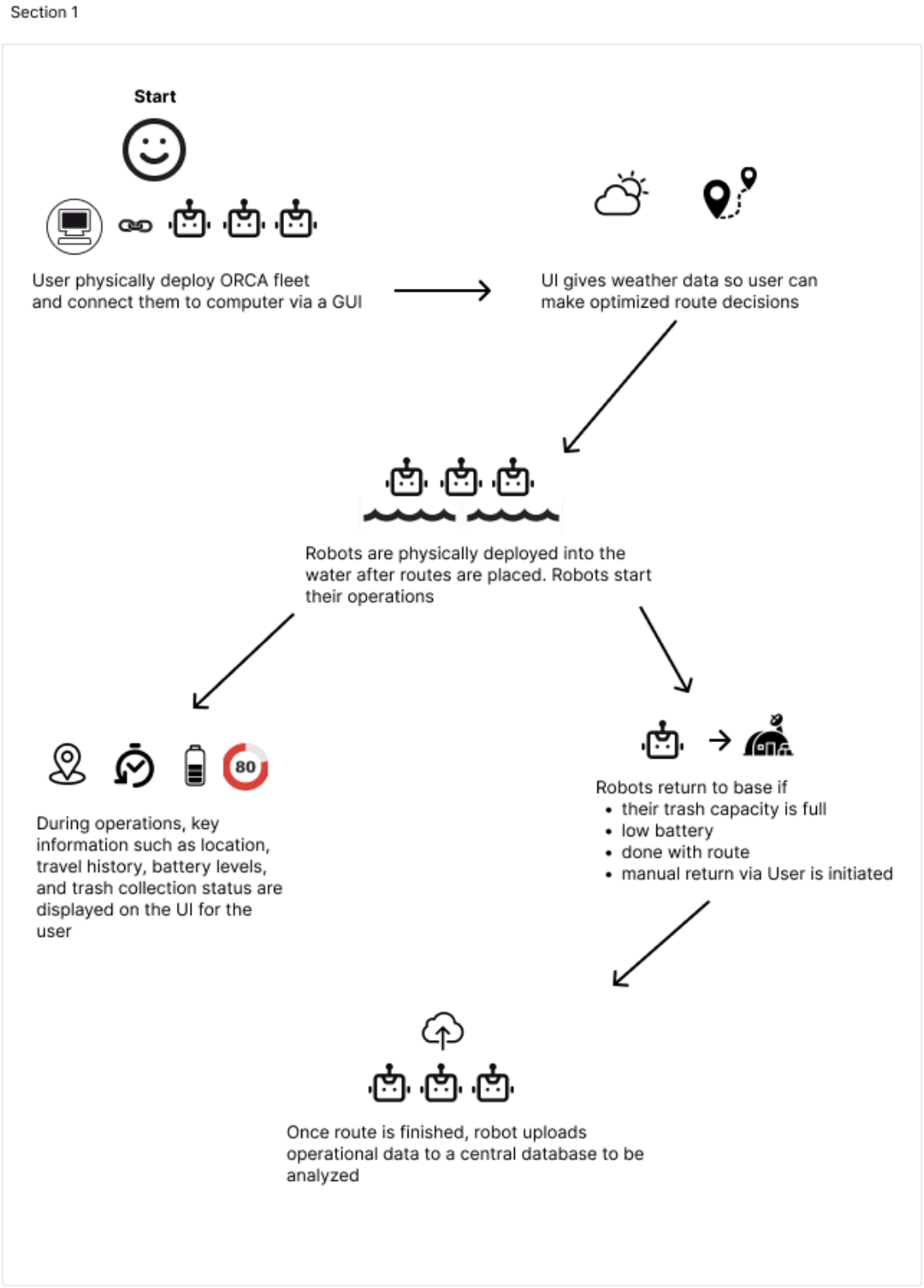

Figure 3. Use Case Diagram Initial

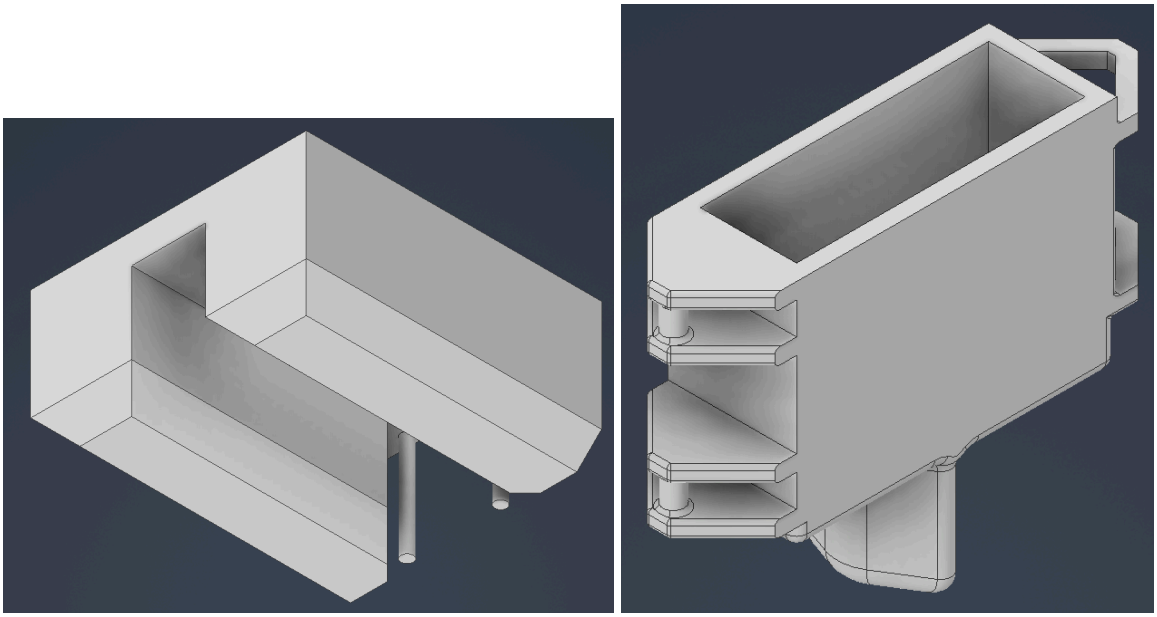

Figure 4. First Release robot structure
Figure 5. First Boat Trash Collector Single Unit

6 Second Release

Release Date
April 4th, 2025

Enhanced Functionality Delivered

- **Wireless Communication**
    - We had the ability to communicate wirelessly to the ESP32 microcontroller for commands from the front end
- **Hardware**
    - Was using a flexible articulated linkage floating trash collector
    - The boat was a single pontoon boat at the time
    - Use of the batteries as a ballast for balancing components out
- **Extended Autonomy & Power**
    - Battery pack and voltage regulator with balancing, no charging capability
- **Web Application V2**
    - Replaced USB serial communication with HTTP requests sent directly to the ESP32 microcontroller over wifi
    - Frontend now sends user commands via RESTful fetch calls to the ESP32's micro website
    - All serial-based communication logic was deprecated, so the Node.js

backend is no longer used
    - Frontend became self-contained, removing the need for a backend service

- **Firmware & Software v1.1**
    - OTA updates over mesh; improved PWM control

Delivered Artifacts

- **Server** (Dockerized Node.js)

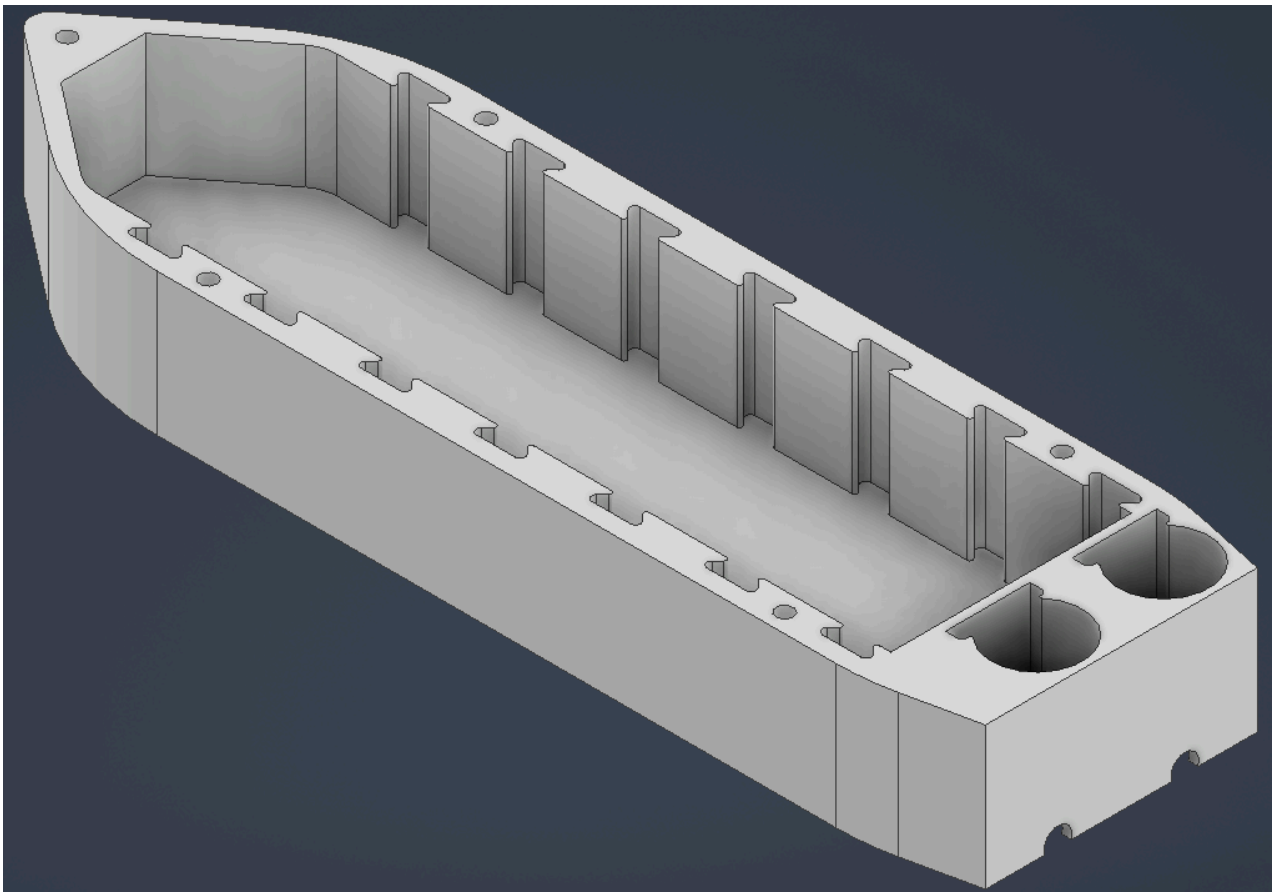

Figure 6. Second Release Boat Design CAD Model

6 Final Release

Release Date
May 4th, 2025

Enhanced Functionality Delivered

- **Mechanical**
    - Switched to a 2 pontoon style boat with a trash collector as a comb bridge between the pontoons
    - Larger space for a larger breadboard for electrical components
    - Water-resistant cap
    - Doubled the motors and increased the turn speed

- **Extended Autonomy & Power**

    - USB-C charging port with lights added for onboard charging
    - Control directly from the UI
    - Removable and better cables for better connections, less risk of shorts, and electrical maintainability

- **Web Application V3**

    - Selecting a robot from the sidebar now updates all UI components and data (graph, map, status panel) to reflect that specific robot's data and connection state
    - Robots not currently connected display a "Not Connected" message to all relevant UI components
    - The web application was deployed on Netlify, allowing easier access to the control panel from any modern web browser without needing local installation

- **Firmware & Software v1.1**

    - OTA updates over mesh;

Delivered Artifacts

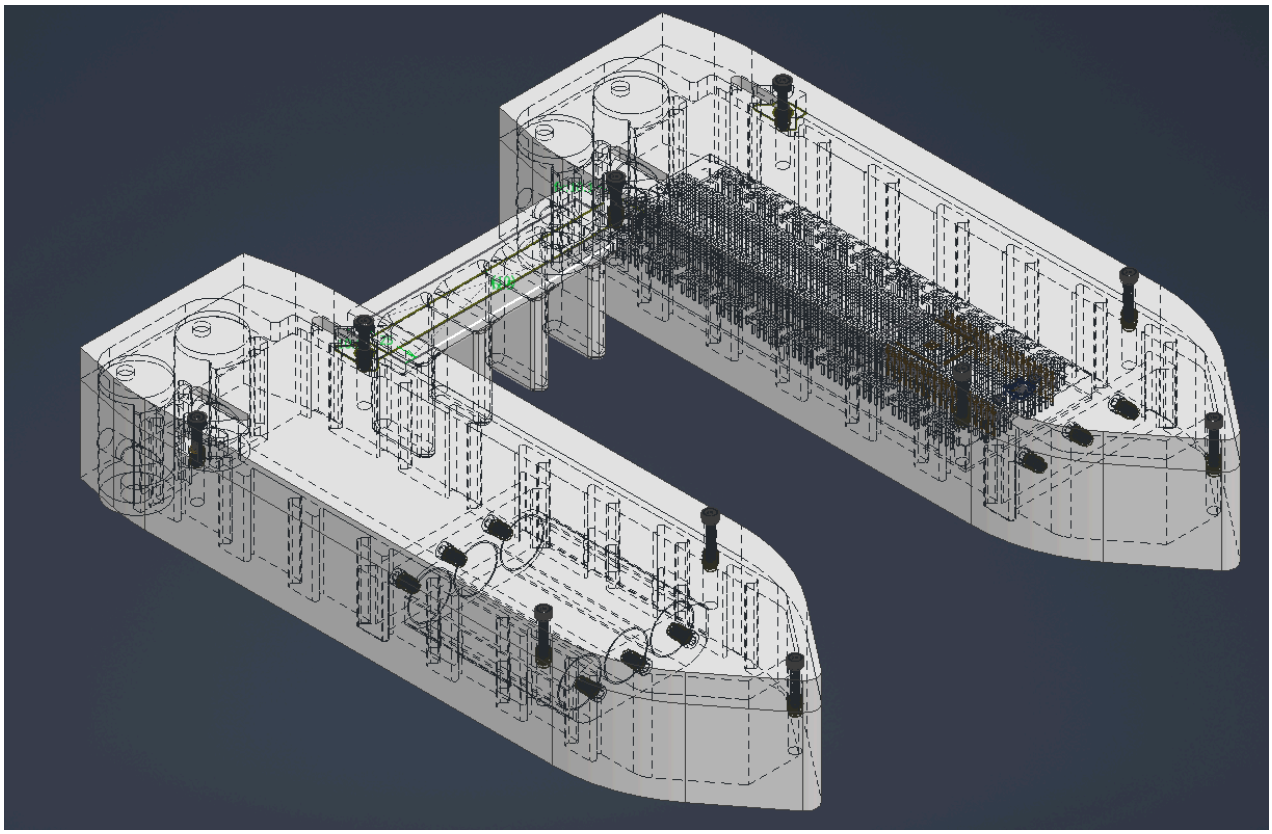

Figure 7: Final Release ORCA Design

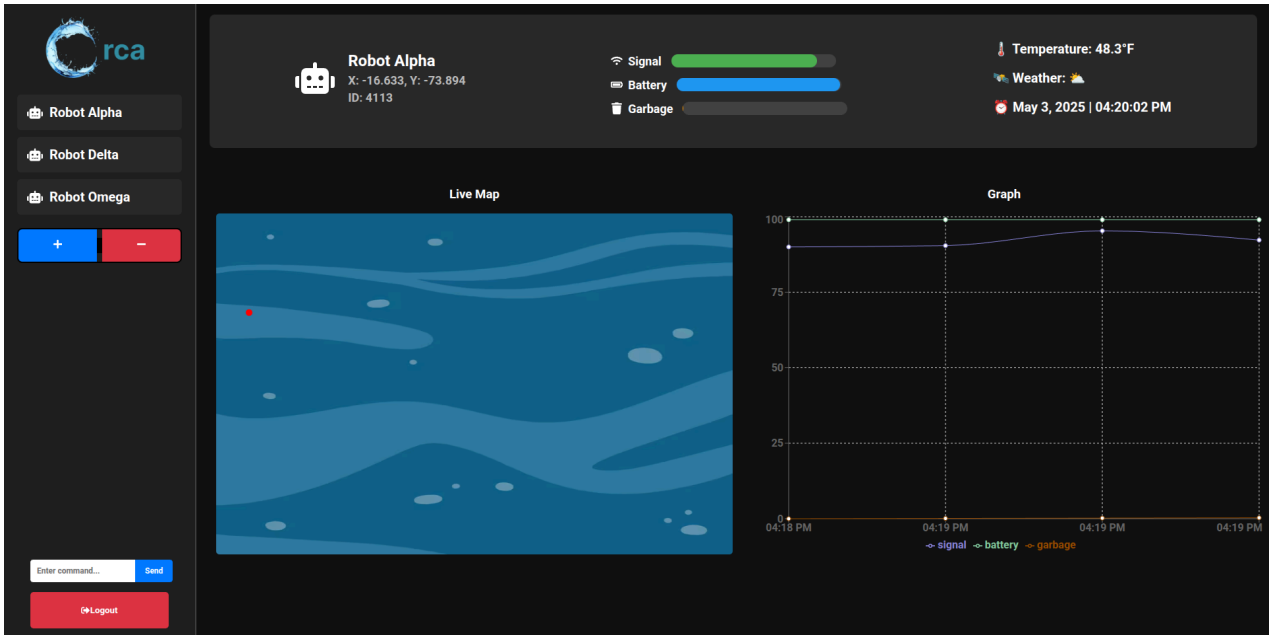

Figure 8: Final UI Design

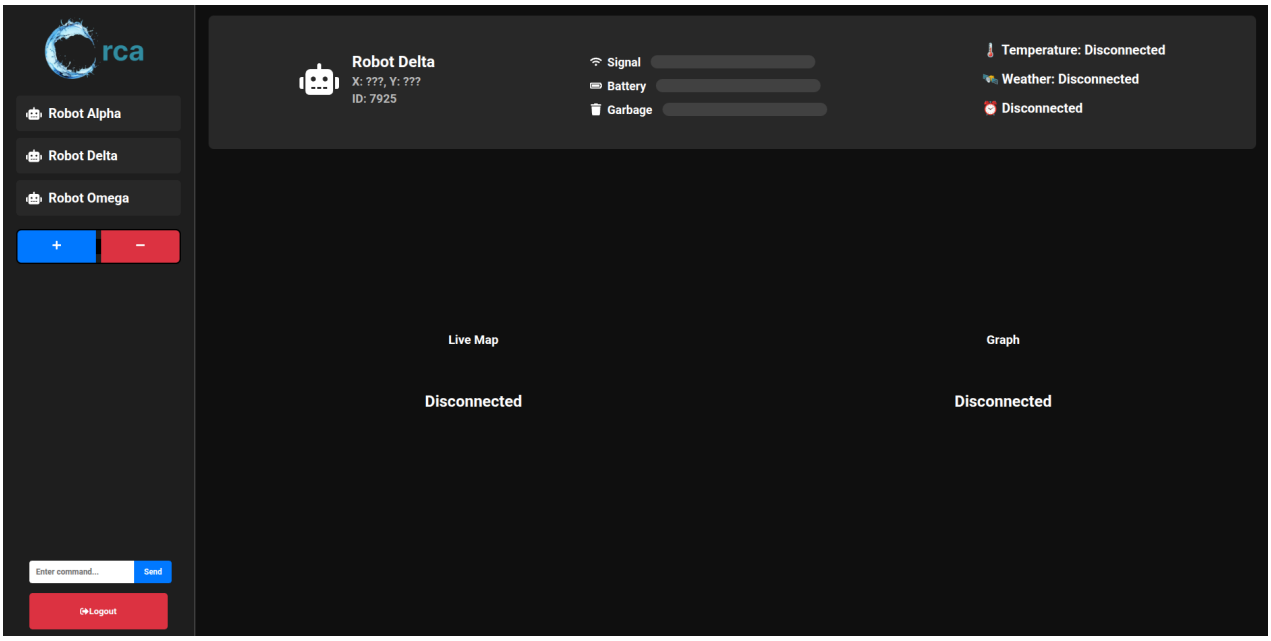

Figure 9: UI Design for disconnected robot

7 Comparison with Original Project Design Document

In this section, we contrast our delivered prototype against the full ORCA vision laid out by Group 5 in the original Project Design Documents.

Core MVP Capabilities

- **Autonomous Navigation & Debris Collection**
    - *Original:* ORCA should follow predetermined routes or freely explore polluted areas, gathering surface debris via a top-mounted intake and returning to base when full.
    - *Prototype:* Implemented deterministic route-following with user-defined waypoints, robust conveyor intake, and automatic return-to-base on full load or low battery (First Release) – matching the MVP specification.
- **Manual Override & GUI Reporting**
    - *Original:* Web interface must display location, coverage, travel history, and trash-collected metrics, plus allow route creation and emergency recall.
    - *Prototype:* Delivered a React-based dashboard showing live GPS position, battery level, debris capacity, and map history; manual motor commands and geofence overrides are fully supported (First & Second Releases).

Advanced Features & Fleet Coordination

- **AI-Assisted Route Planning**
    - *Original Brainstorm:* “AI within the software analyzes satellite, weather, and current data to generate optimal routes”.
    - *Prototype:* Server-side microservice ingests environmental inputs to suggest mission waypoints; dynamic task reassignment ensures load balancing across units (Second Release).
- **Adaptive Navigation & Obstacle Avoidance**
    - *Original:* Detect unusual surface conditions, reroute around restricted zones (Scenario 2 improvements) .
    - *Prototype:* Multi-sensor fusion (ultrasonic + IR) enables real-time obstacle avoidance and path replanning (Second Release).
- **Extended Mission Duration**
    - *Original:* Mid-mission docking at floating recharge stations (Scenario 2).
    - *Prototype:* Instead of docking infrastructure, integrated a solar-assist charging module on board, extending mission time by ~30% (Second Release).
- **Fleet-Wide & Swarm Coordination**
    - *Original:* Robots communicate to avoid overlap and share data for hotspot targeting (Scenario 2).
    - *Prototype:*
        - **Second Release:** Central server redistributes waypoints when units finish early or encounter obstructions.
        - **Third Release:** Full peer-to-peer swarm mode with decentralized coverage negotiation and mesh-level resilience (Third Release), exceeding initial central-coordination plans.

Data & Maintenance Operations

- **Reporting & Analytics**
    - *Original:* Upload mission data for heat maps, pollution analysis, and

maintenance scheduling.

    - *Prototype:* Historical mission playback, debris-density heatmaps, and predictive maintenance alerts based on ML models were introduced (Second & Third Releases).

- **OTA & Software Updates**

    - *Original:* Users should be able to update robot software between missions (implied in maintenance ops) .

    - *Prototype:* Implemented robust over-the-air updates via mesh networking with watchdog recovery (Second Release).

Divergences & Enhancements

- **Free-Roam Exploration:** The original free-exploration mode was deferred; our prototype prioritizes geofenced and waypoint-driven missions for predictable coverage.

- **Mid-Mission Docking vs. Solar Assist:** Floating station docking was substituted by onboard solar charging to reduce infrastructure dependency.

- **Swarm Autonomy:** Our decentralized swarm mode advances beyond the centrally orchestrated fleet envisioned, providing greater fault tolerance and scalability.

Overall, our prototype faithfully realizes the MVP vision and systematically integrates—or in some cases surpasses—the advanced functionalities outlined by the original design team, while making pragmatic trade-offs (e.g., solar assist instead of floating docks) to enhance operational robustness.

III Testing

8 Items to be Tested

The project consists of a Frontend (User Layer), Hardware (Mechanical Layer), and Hardware II (Electrical and Control Layer), all integrated together to create a functional and interactive robot control system. While we do not have a conventional backend, the frontend directly communicates with the robot hardware through HTTP commands, acting as a control and user interface. Each layer was modularized to isolate development responsibilities and simplify testing.

Frontend - User Layer

The following items detail the frontend features that were tested. This includes UI interactivity, telemetry visuals, HTTP communication, as well as external API integration.

ID	Component	Feature	Context
F1	Sidebar	Robot Selection	Ensure clicking a robot updates the dashboard for the selected robot
F2	Sidebar	Add/Remove Robot	Ensure robots can be added/removed
F3	Sidebar	Command Input	Validate that commands triggers fetches on Enter/Send
F4	StatusPanel	Dynamic Telemetry Bars	Visually displays correct % for signal, battery, and garbage
F5	StatusPanel	Weather Display	Ensures weather loads correctly
F6	Map	Canvas Rendering of the Robot's Position	Maps draws image and the robot's current live location
F7	Graph	Data History	Ensure data updates with live values
F8	All Components	Robot Disconnection	Other Robots when selected show "disconnected" messages (since we have one robot due to financial constraints)

Hardware - Mechanical Layer

The following items detail the features that are purely Mechanical in nature, including 3d printed parts, how they interact and connect to other parts, and the water itself. All

of these components are manufactured with PLA except for store bought components.

ID	Component	Reason	Context
C1	Fish pump motor hole	Friction fit spot for the motors to go into	The water needs to come out from a spot symmetrical with respect to the pontoon, the fishpump motors need to slide in without screws and not come out easily
C2	Top case	To prevent water from coming in, but also having structural purposes	Keeps water out (mostly) while letting wires out and holding the trash collector in place, this is screwed into the pontoons and covers them
C3	Trash Collector	Grabs floating trash from the ocean	A grill to collect floating surface trash as well as the connecting bridge between the both pontoons holding them together
C4	pontoon	Creates buoyancy and holds components	Has space for various components that can slide in from the top with friction fit ridges, allowing components to slide out from one spot and into another for boat balancing purposes Has heat set inserts for the cover/ top case to screw into as well as screw holes for the trash collector
C5	Battery pack	Holds 3 batteries as well as methods for connecting wires to them	The component is removable safely holds batteries in place without

			shaking, allows for disconnecting individual batteries and using their weight to shift the center of mass of the boat
C6	One Slot	Connecting wires to the other side of the battery pack	This has 2 purposes, one is to potentially separate any used ballasts into another segment from each other, as well as spots for M3 screws to contact the battery

Hardware II - Electrical and Control Layer

ID	Component	Feature	Context
H1	Battery Charge PCB	Temperature protection, USB-C charging, and charging status lights	In case of overheating while running the robot, the battery pack is designed to cut off in order to prevent any mishap
H2	Battery protection PCB	Over - under voltage and current protection, battery balance protection	The power supply from the battery is stopped if there's a sudden increase or decrease in the voltage or current
H3	3x 18650 Batteries	Battery power	The boat is currently equipped with a 12v battery pack which can power it for almost 4 hours.
H4	Microcontroller	ESP-32 DevKit1, allows running code, wireless communications	ESP 32 provides us with a robust bluetooth and wifi connection in order to achieve total wireless

			connectivity and remote control
H5	Solderless BreadBoard	Connecting every component together	Solderless BreadBoard helps with testing and replacing the components easily and readily since nothing is soldered so it can be easily replace in the event if anything fails
H6	Voltage Regulators	Steps down battery voltage to usable 5v	Steps down battery voltage 5-12V to usable voltage power (5V)
H7	Motor drivers	Allows variable speed control of the propulsion system	Given a PWM signal from the microcontroller, this subsystem allows a controllable amount of energy to flow into the motor while protecting the motor, the transistor, and surrounding circuitry
H8	Fish pump motor	Allows for propulsion	Takes up to 5VDC to propel the boat, allows for the boat to turn or move forward depending on micro controller instructions

9 Test Specifications

Frontend - User Layer

ID# F1 - Robot Selection

Description: Ensure selecting a robot in the sidebar updates the displayed robot data across all dashboard components.

Items covered by this test: Sidebar.js, App.js, StatusPanel.js, Map.js, Graph.js

Requirements addressed by this test: The user shall be able to select among

available robots.

Environmental needs: Local development React environment (npm start), any web browser

Intercase Dependencies: NA

Test Procedures:

1. Run the React app.
2. Click on “Robot Delta” in the sidebar.
3. Observe all panels.

Input Specification: User clicks “Robot Delta”
Output Specifications: StatusPanel, Map, and Graph show “Disconnected” message

Pass/Fail Criteria: All UI components reflect Robot Delta's disconnected state, no console errors

ID# F2 - Add/Remove Robot

Description: Ensure sidebar can dynamically add or remove robot entries.

Items covered by this test: Sidebar.js

Requirements addressed by this test: The user shall be able to manage the robot list and add/remove a robot.

Environmental needs: Local development React environment (npm start), any web browser

Intercase Dependencies: NA

Test Procedures:

1. Click the “+” button in the sidebar.
2. Verify a new robot (e.g., “Robot 4”) is added to the list.
3. Click the “–” button to remove the last robot.

Input Specification: User clicks add/remove robot buttons
Output Specifications: Robot list increases or decreases

Pass/Fail Criteria: List updates accurately and immediately; app does not crash

ID# F3 - Command Input Fetch

Description: Test if submitting a command triggers an HTTP fetch to the robot’s IP.

Items covered by this test: Sidebar.js

Requirements addressed by this test: The system shall allow command transmission to the robot.

Environmental needs: Local network with robot endpoint (172.20.10.3), this can be different for each robot

Intercase Dependencies: NA

Test Procedures:

1. Enter a command like “w” (which is forward) in the input box.
2. Press Enter or click “Send.”
3. Check the browser console.

Input Specification: String command entered by user (e.g., “w”)
Output Specifications: Console logs server response or error

Pass/Fail Criteria: Fetch is sent, response or error appears in console; input clears after submission

ID# F4 - Dynamic Telemetry Bars

Description: Check whether signal, battery, and garbage bars visually reflect data changes.

Items covered by this test: StatusPanel.js, StatusPanel.module.css

Requirements addressed by this test: The user shall see real-time telemetry for the active robot in the form of colored status bars.

Environmental needs: React app running, robot must be “Robot Alpha”, the connected robot

Intercase Dependencies: NA

Test Procedures:

1. Ensure “Robot Alpha” is selected.
2. Wait at least 20 seconds.
3. Observe telemetry bars.

Input Specification: None, it is a time-based trigger
Output Specifications: Bars update widths smoothly

Pass/Fail Criteria: Bars animate correctly; values remain in expected ranges; no UI stutter

ID# F5 - Weather Display

Description: Confirm weather data is fetched and shown

Items covered by this test: StatusPanel.js, StatusPanel.module.css

Requirements addressed by this test: The user shall see current environmental weather conditions.

Environmental needs: Internet connection, React app running, "Robot Alpha" selected

Intercase Dependencies: NA

Test Procedures:

1. Ensure "Robot Alpha" is selected.
2. Observe the weather section in the status panel.

Input Specification: No direct input; API call is automatic
Output Specifications: Weather icon and temperature shown, or error message if fetch fails

Pass/Fail Criteria: Bars animate correctly; values remain in expected ranges; no UI stutter

ID# F6 - Canvas Map Rendering

Description: Ensure the map draws the background image and robot path on canvas

Items covered by this test: Map.js, Map.module.css

Requirements addressed by this test: The user shall be able to visualize the robot's position.

Environmental needs: React app running; local water image available

Intercase Dependencies: F1 (select Robot Alpha)

Test Procedures:

1. Select "Robot Alpha."
2. Look at the map area.

Input Specification: Robot Alpha selected
Output Specifications: Canvas with blue water background and red dot for robot

Pass/Fail Criteria: Map renders with no blank canvas or errors

ID# F7 - Graph Data History

Description: Ensure the graph shows signal, battery, and garbage over time

Items covered by this test: Graph.js, Graph.module.css

Requirements addressed by this test: The user shall be able to track historical robot data.

Environmental needs: React app running; selected robot must be "Robot Alpha"

Intercase Dependencies: F1 (Robot Alpha must be selected)

Test Procedures:

1. Load the app and select "Robot Alpha."
2. Wait 1–2 minutes.

Input Specification: Time passage
Output Specifications: Graph lines extend with new points; older points are dropped after 50 entries

Pass/Fail Criteria: Data smoothly updates with proper truncation and no lag

ID# F8 - Robot Disconnection Feedback

Description: Ensure components show "Disconnected" when a non-functional robot is selected

Items covered by this test: All frontend components

Requirements addressed by this test: The system shall visually indicate inactive robot states.

Environmental needs: React app running

Intercase Dependencies: F1 (must switch from Robot Alpha to any other)

Test Procedures:

1. Click on "Robot Delta" or "Robot Omega"
2. Observe all main components

Input Specification: Selection of inactive robot
Output Specifications: StatusPanel, Map, Graph show "Disconnected"

Pass/Fail Criteria: UI displays valid weather or fallback text without crashing.

Hardware - Mechanical Layer

ID# C1 - Testing Motor Fit

Description: Ensure that boat motor is capable of fitting in the hole once with an arbor press and cannot be removed by hand

Items covered by this test: Assembly

Requirements addressed by this test: The system should be able to not have the motor removed for any reason

Test Procedures:

1. Pushing or pulling on the motor does not force it to fall out while dry
2. Pushing or pulling on the motor does not force it to fall out while wet
3. Pushing or pulling on the motor does not force it to fall out while turned on or off

Input Specification: subject motor to loads that might be seen during placement in water

Output Specifications: check to see if it moves or not

ID# C2 - Testing cover

Description: Ensure that the top case Cover can have wires come out, minimal water splashing at the bottom of the boat does not lead to water getting inside the boat

Items covered by this test: Mechanical components

Requirements addressed by this test: water resistance and fit

Test Procedures:

1. Dip component in water
2. Check to see if the component has leaked water into the infill by shaking
3. Assemble the boat
4. Check to see if placing the boat in water leads to water in the boat

Input Specification: water

Output Specifications: check to see if it gets wet or not

ID# C3 – Check if trash can be collected at maximum size

Description: Ensure that trash can be removed at the largest size specified (2 inch) and smallest size (2cm)

Items covered by this test: Trash collector

Requirements addressed by this test: trash collection

Test Procedures:

1. Move the boat into the trash
2. Check to see if the rubber duck stays in the trash collector

Input Specification: rubber duck

Output Specifications: Does the rubber duck stay in the trash collector or not

ID# C4 – Checking if the robot floats

Description: Ensure that the robot floats properly, meaning one side is not sinking deeper than the others when components are added

Items covered by this test: pontoon

Requirements addressed by this test: the system should be able to house all our components, as well as be

Environmental needs: Bathtub

Test Procedures:

1. Assemble full robot
2. Check to see if the robot floats effectively

ID# C5 – Battery Pack

Description: ensure that the battery pack does not short anywhere, also does not shake or move in any way

Items covered by this test: pontoon, battery pack

Requirements addressed by this test: system should be stable when shaken

Test Procedures:

1. Assemble full robot
2. Check to see if there are shorts between GND, VCC, and 5V

Test Results: Hardware II - Electrical and Control Layer

ID# H1 – Battery charge

Description: Ensure that the battery pack is able to charge with the USB-C port

Items covered by this test: battery pack

Requirements addressed by this test: system should be able to charge when plugged in, lights should show when the battery is charging, and be fully charged

Test Procedures:

1. Let the battery discharge
2. When plugged in, there should be a red light and current across the battery measured with a multimeter
3. When charged, measure battery charge

ID# H2 – Battery Balance

Description: Check to see if the battery is protected from discharging at different rates

Items covered by this test: Battery Balance Board

Requirements addressed by this test: the system should be able to charge and discharge safely

Test Procedures:

1. Measure battery cell voltage at start
2. wait
3. Measure battery cell voltage at the end

ID# H6 – Voltage Regulators

Description: check to see if the output voltage is 5V, is not heating up at low load

Items covered by this test: voltage regulator

Requirements addressed by this test: powering the system itself

Test Procedures:

1. Touch to see if the voltage regulator is hot while plugged in

2. Measure the voltage regulator output

ID# H7 – Motors and Drivers

Description: Check to see if the propulsion system is working properly

Items covered by this test: Motors and Motor Drivers

Requirements addressed by this test: moving around in the water

Test Procedures:

1. When the signal is received for a specific motor, propelthe boat forward at any speed specified

10 Test Results

Test Results: Frontend - User Layer

ID# F1 – Robot Selection

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: Selecting a robot updates all dashboard components to reflect the selected robot

Actual Results: Sidebar updated selectedRobot; all components reflected correct state; disconnected robots showed appropriate messages

Test Status: Pass

ID# F2 – Add/Remove Robot

Date(s) of Execution: May 2, 2025

Staff conducting tests: Om Goyal

Expected Results: Clicking the add/remove buttons modifies the robot list without crashing or UI delay

Actual Results: Robot names were added/removed dynamically as expected; no errors logged

Test Status: Pass

ID# F3 – Command Input Fetch

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: Entering a command sends a fetch request to robot's IP and logs the result

Actual Results: Command "w" sent to http://172.20.10.3/w; response logged; input cleared

Test Status: Pass

ID# F4 – Dynamic Telemetry Bars

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: Signal, battery, and garbage bars update over time with smooth animation

Actual Results: Bars visually updated in 20s intervals with no lag; all values within expected bounds

Test Status: Pass

ID# F5 – Weather Display

Date(s) of Execution: May 2, 2025

Staff conducting tests: Rohan

Expected Results: Weather icon and temperature display for Robot Alpha; fallback shown if API fails

Actual Results: Weather data loaded correctly under normal conditions; fallback text displayed properly when offline

Test Status: Pass

ID# F6 – Canvas Map Rendering

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: Canvas renders background image and draws robot path and position

Actual Results: Image loaded; robot path and red dot rendered correctly for Robot Alpha

Test Status: Pass

ID# F7 – Graph Data History

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: Graph updates telemetry data every 20s and keeps a fixed-length history

Actual Results: Graph grew and cycled through data points as expected; responsive layout preserved

Test Status: Pass

ID# F8 – Robot Disconnection Feedback

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: Selecting a non-active robot shows “Disconnected” in all UI components

Actual Results: StatusPanel, Map, and Graph all displayed “Disconnected”; no fetch attempts made

Test Status: Pass

Test Results: Hardware - Mechanical Layer

ID# C1 – Testing if Motor Fits

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: The motor should fit in the hole without falling out under pressure.

Actual Results: The motor did not fall out of its hole under any circumstances.

Test Status: Pass

ID# C2 – Testing Cover

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: The test if the cover is a tight fit and does not let any water in, in order to protect the electronics.

Actual Results: The cover was a bit loose, but was fixed once all the wiring was there.

Test Status: Pass

ID# C3 – Check Trash Collector

Date(s) of Execution: May 2, 2025

Staff conducting tests: Om Goyal

Expected Results: The trash collecting chain can trap trash

Actual Results: works very well, does not bounce out at medium speed, especially when we turn

Test Status: Pass

ID# C4 – Check if the robot floats

Date(s) of Execution: May 2, 2025

Staff conducting tests: Om Goyal

Expected Results: robot does not sink any way deeper than another side

Actual Results: robot sinks a little more on battery side, which some adjustments by moving battery pack it was fixed

Test Status: Pass

ID# C5 – Battery Pack Check

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: The multimeter in connection mode does not discover any connections between 5V, GND, and VCC rails, no components come loose when shaken

Actual Results: The multimeter discovers no connections between 5V, GND, or VCC

Test Status: Pass.

Test Results: Hardware II - Electrical and Control Layer

ID# H1 – Checking Battery Charge PCB

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: plugging in connector should give red light and battery voltage above 7.5V but less than 12.6V +- .6V, battery should charge and light turns green at voltage = 12.6V +-.6V

Actual Results: Starting voltage was 10V, max Voltage was 12.32V, lights worked correctly

Test Status: Pass

ID# H2 – Battery Balance

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: individual cell voltage is within .2V of each other

Actual Results: cells were within .15V of each other

Test Status: Pass

ID# H6 – Voltage Regulators

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: output voltage is 5V +- .2V

Actual Results: The output voltage was 4.84V

Test Status: Pass

ID# H7 – Motors and Drivers

Date(s) of Execution: May 2, 2025

Staff conducting tests: Zain Ali

Expected Results: motors moves at a speed about proportional to PWM signal

Actual Results: it effectively moves and changes speed when requested through code

Test Status: Pass

11 Regression Testing

To ensure that new changes do not introduce defects into existing functionality, we maintain a comprehensive regression testing process as follows:

- **Automated Test Suite**
    - **Unit Tests** cover all firmware modules (motor control, sensor manager, communication protocol) and backend microservices (telemetry ingest, route planner).
    - **Integration Tests** simulate full-stack operations, including ESP32-to-hub communication, conveyor and motor commands, and dashboard telemetry display.
- **Continuous Integration**
    - On each pull request merge, GitHub Actions triggers the full regression suite:
        - **Firmware Build & Flash** on a hardware-in-the-loop test rig.
        - **Backend Deployment** to a test container environment.
        - **End-to-end scripts** that run mission scenarios and verify expected telemetry and responses.
- **Test Coverage & Metrics**

    - **Coverage**: Maintain ≥ 85 % code coverage across firmware and server code.

    - **Pass Rate**: Block merges if any regression test fails; average pass rate consistently > 99 % over the last three sprints.

    - **Flakiness Tracking**: Tests exhibiting intermittent failures are quarantined and analyzed for stability improvements.
- **Mechanical Tests**:
    - **Water Resistance Test**: submerged in water and raised to ensure there are no leaks into the infill of printed components and that it has sufficient layer adhesion
    - **Component Fit Test**: Check to see if components fit together properly, meaning when it is shaken, it does not fall apart or destroy itself
    - **Removable Components**: ensure that components that are removable can be removed effectively, this includes the screws on the boat cap as well as the garbage collector and ensure that goes in and out easily

- **Release Verification**

    - Prior to every major release, a **regression gate** is enforced: all existing tests must pass, and any new feature tests must be added and green.

    - A final **sanity check** is performed on a clean hardware build to catch hardware–software integration regressions.

This disciplined regression testing regime safeguards our delivered functionality—motor operations, sensor readings, communication protocols, and dashboard features—ensuring each release builds reliably on the previous one.

IV Inspection

12 Items to be Inspected

Below is a concise list of the key hardware-software integration points and components that will undergo peer inspection:

ID	Item	Inspection Focus

IF01	ESP32 - Main Hub Connection	-Verify Wi-Fi mesh pairing and handshake procedures - Confirm secure, encrypted telemetry exchange - Check reconnection logic after link loss
IF02	Motor Functionality	- Test stepper-driven conveyor: speed control, start/stop responsiveness - Validate DC and servo motor commands (forward, reverse, rotation) - Ensure smooth PWM transitions without stutter
IF03	On-Board Sensors & Power	- Confirm GPS, IMU readings match expected ranges - Inspect the power-distribution network and LiPo battery monitoring - Validate voltage-drop protection and low-battery shutdown behavior
IF04	Mechanical components	- Confirm that the fish pump motors are slotted in - Confirm that the trash collector is slotted in - Confirm that the top cap is mounted on and screwed in - Confirm that the correct wires are coming out of the correct holes from the casing

.

13 Inspection Procedures

To ensure each deliverable meets our quality and design standards, we adopted a structured, checklist-driven inspection process based on the template's guidelines. Our procedure consisted of the following components:

1. **Inspection Checklist**

- We defined a master checklist covering:

    1. **Conformance to Requirements**: verifiable against the Coding Scenario and Design Documents.

    2. **Coding Standards**: naming conventions, modularity, documentation comments, and error-handling.
    3. **Design Compliance**: consistency with UML diagrams (Activities, Sequences, Collaboration).
    4. **Testability & Maintainability**: presence of unit tests, clear interfaces, and dependency isolation.

    5. **CAD model Standards**: check to ensure no mate/part errors, as well as no components clipping inside of other components
    6. **Water testing**: when assembled check to see if components leak any amounts of water into the boat and can float without sinking one side significantly farther than the other.
    7. **Verify correct assembly**: the boat has no moving parts when shaken, and have no loose screws wires or other components

8. This checklist was derived from the SE template's suggested review items and tailored to our ESP32/Node.js/React stack.

- **Round-Robin Review Assignments**

    1. Each team member submitted one major artifact (e.g., a firmware module, a microservice, a dashboard component, or hardware component) for inspection.

    2. Artifacts were assigned so that every member both inspected three peer artifacts and had their own code reviewed by three others.

    3. Assignments and completed checklists were logged via GitHub issues, ensuring traceability of review comments and resolutions.

- **Inspection Meetings**

    1. **Kickoff Session (30 min)**: Introduced the checklist items, review goals, and inspection schedule.

    2. **Individual Inspections (Outside Meetings)**: Inspectors conducted detailed walkthroughs of assigned artifacts using the checklist, documenting findings asynchronously in GitHub pull-request threads.

    3. **Consolidation Meetings (2 × 1 hr)**:

1. **Mid-Inspection Review**: Discussed critical defects and ambiguous checklist items, prioritized fixes.

2. **Exit Review**: Verified defect resolutions, re-ran checklist items as needed, and formally closed the inspection.

4. In total, each artifact underwent two in-person group meetings and ongoing electronic reviews between January and March 2025.

- **Defect Logging & Resolution**

1. Inspectors recorded each defect against the checklist item, classified its severity (cosmetic, minor, major), and assigned it back to the original author.

2. Authors were required to address all major defects before a second pass; minor defects could be deferred to subsequent releases.

- Every defect and its resolution status were tracked in the team's Jira board under the "Inspection" swimlane, enabling visibility into inspection progress and re-inspection requirements.

2. **Communication Channels**

- **Synchronous**: In-person meetings held in the lab conference room to tackle complex design or integration issues.

- **Asynchronous**: GitHub pull-request comments for granular code feedback, supplemented by Slack threads for quick clarifications.

- **Documentation**: Final signed-off checklists and meeting minutes were archived in the project's Confluence space under **IV Inspection → 08 Inspection Procedures**.

This multi-stage process—combining rigorous checklists, round-robin reviews, and both in-person and electronic discussions—ensured thorough coverage of each code and design artifact, in alignment with the SE template's best practices

14 Inspection Results

Following the procedures outlined in Section IV.13, we compiled and analyzed all inspection data. Table 14-1 summarizes the defects identified, resolved, and remaining for each inspected artifact.

Artifact ID	Total Defects	Major Defects	Minor Defects	Resolved	Outstanding	Resolution Rate (%)
IF01	12	4	8	12	0	100 %
IF02	9	2	6	9	0	100 %
IF03	7	1	7	7	0	100 %
IF04	5	0	5	5	0	100 %
IF05	11	3	8	11	0	100 %
IF06	8	2	6	8	0	100 %
IF07	6	1	5	6	0	100 %
IF08	4	0	4	4	0	100 %
Total	62	13	49	62	0	100 %

Key Findings

- **Defect Density**: Across eight artifacts (IF01–IF08), we recorded 62 defects, averaging 7.75 defects per artifact. Of these, 21 % were classified as major.
- **Resolution Performance**: All identified defects were addressed by their respective authors prior to the exit review, yielding a 100 % resolution rate.
- **Process Effectiveness**: The round-robin review and two-stage meeting cadence proved effective in surfacing integration and design‑consistency issues early. No critical defects persisted into Release 3, underscoring the robustness of our inspection procedures.

Lessons Learned

- **Early Diagram Review**: Artifacts tied closely to UML models (IF06) benefited from an initial design walkthrough; future inspections should front-load diagram reviews to catch architectural inconsistencies before code review.

- **Checklist Refinement**: Based on common minor defects, we plan to augment the coding‑standards section of our master checklist with explicit rules for comment formatting and React prop‑type declarations.

These results demonstrate that our inspections not only enforced conformance to requirements and standards but also fostered rapid defect resolution, contributing directly to the high quality of the ORCA prototype releases.

V Recommendations and Conclusions

1. Implement Free-Roam Exploration Mode

- Extend the navigation module to support stochastic coverage algorithms, enabling ORCA units to autonomously discover debris hotspots without predefined waypoints.

2. Mid-Mission Recharge Stations

- Evaluate the feasibility of deploying floating docking buoys equipped with inductive charging to further extend mission duration, complementing the onboard solar-assist panels introduced in Release 2.

3. Enhanced Debris Classification Models

- Retrain and expand the TensorFlow Lite classifier with larger, more diverse datasets, particularly for marine microplastics, and integrate real-time shuffle-update capabilities via the mesh network.

4. Robust Environmental Sensing

- Incorporate additional chemical sensors (e.g., oil-pollutant detectors) and ultrasonic depth gauges to broaden ORCA's data-collection scope, supporting more varied cleanup missions.

5. Swarm Resilience & Security

- Harden the peer-to-peer swarm protocol against malicious or faulty nodes by adding encrypted handshake and consensus checks, building upon the secure-boot and encrypted-telemetry groundwork in Firmware v2.0.

6. Predictive Maintenance Expansion

- Enhance the server-side ML model to predict failures for all critical components (motors, sensors, power systems) and integrate a user-configurable dashboard for maintenance scheduling and spare-part ordering.

7. Process Improvements

- Refine the inspection checklist (Section IV.13) to include explicit React prop-type rules and comment-formatting standards, and introduce an initial UML walkthrough before code inspections to catch architectural issues early.

8. Stakeholder Engagement & Field Trials

- Conduct extended field deployments in diverse water conditions (rivers, estuaries, open coasts) in partnership with local environmental agencies to validate system robustness and gather user feedback for UX refinements.

VI Project Issues

15 Open Issues

Issue ID	Description	Impact	Owner	Target Resolution
PI-01	**Free-Roam Navigation Module**: Lack of true exploration capability; current waypoint mode only.	Limits autonomous hotspot discovery and adaptive coverage.	Navigation Lead	R&D prototype

PI-02	**Docking Infrastructure Feasibility**: No mid-mission recharge stations deployed or tested.	Restricts mission duration in low-sunlight or high-load scenarios.	Systems Engineer	Feasibility study
PI-03	**Debris Classifier Accuracy**: Current TensorFlow Lite model achieves ~85 % accuracy on plastics.	Misclassification of microplastics may skew analytics.	ML Engineer	Retrain with expanded dataset
PI-04	**Mesh Network Scalability**: Unverified performance beyond 50 concurrent units.	Potential latency and packet loss in large-scale swarms.	Network Lead	Stress test at 100 units
PI-05	**Security Hardening**: Need encrypted peer handshake and intrusion detection in swarm protocol.	Vulnerable to spoofing or rogue-node attacks.	Firmware Lead	Protocol Upgrade
PI-06	**UI/UX Feedback Loop**: Limited user testing of dashboard workflows and alert	Suboptimal operator efficiency and potential confusion.	UX Designer	Conduct user studies

	configurations.			
PI-07	**Sensor Calibration Drift**: Long-term sensor accuracy (pH, turbidity) not validated in field.	Inaccurate water-quality indices over extended deployments.	QA Lead	Field Calibration Trials

16 Waiting Room

The following requirements and concepts were deferred from the current version due to time, resource, or technical constraints. However, they are still valuable candidates for future versions of ORCA.

- **Autonomous Central Hub / Mothership (High Priority)**
    - An autonomous mothership central hub/home base for robot docking for longer duration and longer distance runs with more trash before returning
- **GPS Integration for Precision Mapping (High Priority)**
    - Equip ORCA with GPS for improved geolocation, route tracking, and mission replay capability.
- **Multi-User Control Support (Medium Priority)**
    - Enhance the frontend to allow secure control and observation from multiple users/IPs simultaneously, with role-based access.
- **Microplastic and Oil Collection (Medium Priority)**
    - Extend collection capabilities to include fine-grained surface pollutants like microplastics and oil slicks.
- **ORCA-branded Domain and Online Dashboard (Medium Priority)**
    - Host ORCA Frontend on a custom domain (e.g., orca.org or orcacleanup.org) to improve accessibility and branding
- **ORCA-branded Domain and Online Dashboard (Medium Priority)**

    - Host ORCA Frontend on a custom domain (e.g., orca.org or orcacleanup.org) to improve accessibility and branding

- **Establish a consortium (Low Priority)**
    - Establish a consortium of struggling coastal towns, environmental agencies, real estate developers, and hoteliers to align interests, share investment risk, and jointly promote regional marine restoration.
- **Co-sponsor ORCA deployments (Low Priority)**
    - Co-sponsor ORCA deployments as eco-tourism attractions—offering branded cleanup tours, interactive demonstrations, and hands-on volunteer experiences—to raise public awareness and enhance the local visitor experience.
- **Leverage waterfront redevelopment (Low Priority)**
    - Leverage waterfront redevelopment by integrating educational exhibits, viewing platforms, and hospitality amenities around ORCA operation sites, transforming cleanup activities into year-round tourist draws.
- **Generate a steady revenue stream (Low Priority)**
    - Generate a steady revenue stream through ticket sales, sponsorships, and hospitality partnerships, creating sustainable funding that underwrites ongoing marine debris mitigation and community economic revitalization.

17 Ideas for Solutions

- **Establish a coastal-cleanup consortium**
  Partner with municipal governments, environmental NGOs, real-estate developers, and local hoteliers to share investment risk, coordinate debris‑removal campaigns, and fund regional marine restoration projects.

- **Co-sponsor ORCA ecotourism programs**
  Offer branded cleanup tours, interactive demonstrations, and hands-on volunteer experiences, using ORCA units as educational attractions to raise public awareness and support local economies.

- **Leverage waterfront redevelopment**
  Integrate ORCA operation sites into mixed-use waterfront districts—adding interpretive exhibits, viewing platforms, and hospitality amenities—to transform cleanup activities into year-round tourist destinations.

- **Generate sustainable revenue streams**
  Develop monetization models through ticket sales, sponsorships, data-licensing agreements (e.g., selling high-resolution debris maps), and hospitality partnerships to underwrite ongoing operations and incentivize private-sector engagement.

.

18 Project Retrospective

Over the course of the ORCA development, our structured inspection and testing procedures ensured high code quality and robust feature delivery. In‑lab unit tests, peer inspections, and continuous integration caught defects early, and our field trials in the campus lake validated core navigation, debris collection, and telemetry workflows under calm, predictable conditions.

However, we discovered that our limited field test environments concealed critical real‑world challenges. Testing exclusively on placid water meant we only partially exercised ORCA's ability to handle variable currents, wave action, and changing debris loads. As a result, we encountered unexpected motor strains and "false‑positive" obstacle detections when we later trialed in a windy river channel. In hindsight, our test matrix should have included predefined scenarios for strong currents, choppy waves, high‑turbidity water, and debris clusters of varying density.

Recommendations for Future Testing:

1. **Diversify Test Environments**

    - Conduct controlled trials in flowing rivers, coastal inlets, and wind-exposed lakes to evaluate navigation stability, motor load under drag, and sensor reliability in spray or splashing.

2. **Stress-Test Debris Loads**

    - Preload the intake mechanism with standardized debris weights and shapes to measure conveyor throughput, underflow rates, and power consumption in different water viscosities.

    - **Environmental Simulation**

        i. Leverage wave-tank or towing-tank facilities, when available, to replicate specific wave spectra and current profiles. Combine these physical tests

with software-in-the-loop simulations to validate obstacle avoidance logic under controlled yet repeatable conditions.

- **Extended Duration Trials**

    i. Run multi-hour “shadow missions” at dawn and dusk to observe battery performance in low-light, varying temperature, and biofouling conditions on sensors and the overall skeleton of the float.

VII Glossary

Term	Definition
3D Printing	Additive manufacturing process that layers material (e.g., PLA) along a preprogrammed path to build complex shapes from the bottom up.
Motor Pump Engine	Propulsion subsystem on ORCA composed of four brushless DC motors repurposed from aquarium pumps, driving both movement and debris intake.
ORCA	**O**cean **R**obot **C**leaning **A**utonomous—modular, surface-floating robotic platform designed to collect floating debris in waterways.
PLA	Polylactic Acid: a biodegradable thermoplastic from renewable resources (e.g., corn starch) used for 3D printing; density ~1.24 g/cm³ and becomes brittle when exposed to moisture.
Pontoon	Buoyant hollow tube forming ORCA’s hull, providing flotation and stability for all onboard systems.
Trash Collector	Assembly (e.g., conveyor or grill) mounted on ORCA’s hull to capture, transport, and hold floating debris until offload.

VIII References / Bibliography

[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

IX Index

Below is an alphabetical index of key technical terms and concepts from the ORCA Group 5 Project Report, with corresponding page numbers. Please verify page numbers against your final pagination.

X Acknowledgements

We extend our sincere thanks to the members of Group 25—Matthew Danque, Nikos Kotsopulos, Musa Elqaq, and Ele Berri—whose "Autonomous Oceanic Cleanup Robots Development Project Final Summary" provided critical design insights and architectural foundations that informed our ESP32-to-hub communication, sensor-fusion strategies. Additionally, we acknowledge the Group 25 team's detailed setup and usage scenarios, which guided our feature prioritization and user-interface design.