Acknowledgement

I am profoundly grateful to my academic supervisor, **Dr. Abbes Mehdi**, whose expert mentorship, insightful guidance, and steadfast encouragement were fundamental to the success of the Autonomous Cleaner Boat Minimum Viable Product (MVP) for debris collection in marinas, rivers, and lakes.

My heartfelt thanks extend to **Graines d'Entrepreneurs** for their generous support, resources, and inspiration, which empowered this innovative project. I also express deep appreciation to my enterprise supervisor, **Mr. Mohamed Nadim Touil**, for his practical advice and valuable contributions within the startup environment. I am thankful to my teammates for their collaborative efforts and shared commitment to environmental sustainability. Lastly, I owe gratitude to my family and friends for their unwavering support, which sustained me throughout this endeavor. Your collective efforts have made this project a significant step toward cleaner waterways.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABS Acrylonitrile Butadiene Styrene (hull material)

AMCL Adaptive Monte Carlo Localization (navigation algorithm)

ASV Autonomous Surface Vehicle

GPS Global Positioning System

IMU Inertial Measurement Unit

LiDAR Light Detection and Ranging

MVP Minimum Viable Product

ROS Robot Operating System

SLAM Simultaneous Localization and Mapping

URDF Unified Robot Description Format (simulation model)

OS Operating System

General Introduction

Plastic debris and floating waste in marinas, rivers, and lakes endanger aquatic ecosystems, disrupt biodiversity, and impact local communities. With millions of tons of waste polluting inland and coastal waterways annually, innovative solutions are essential to address this environmental challenge. This Final Year Project (PFE) presents a Minimum Viable Product (MVP) for an autonomous cleaner boat designed to collect floating debris in marinas, rivers, and lakes. Developed in collaboration with Graines d'Entrepreneurs, a Tunisian initiative that empowers young innovators, this project delivers a practical and technology-driven tool for sustainable waste management.

The motivation for this MVP stems from the limitations of existing marine waste collection methods, which are often manual, costly, or restricted to specific areas. Autonomous surface vehicles (ASVs) offer a transformative approach, enabling efficient navigation and waste collection with minimal human intervention. By integrating sensors, artificial intelligence, and lightweight design, this project aims to create a functional prototype that demonstrates the feasibility of autonomous waste collection, paving the way for future refinements and deployment.

The primary objective is to design, implement, and test an MVP of an autonomous cleaner boat using a catamaran configuration for stability. The system employs a Raspberry Pi 4 for processing, ROS Noetic for software integration, and sensors such as LiDAR, GPS, and cameras for navigation and waste detection. A YOLO-based vision system identifies debris in real-time, while path planning and obstacle avoidance ensure safe operation. Sustainable power systems and remote monitoring enhance the MVP's practicality. Developed over four months, the MVP was validated through simulations and initial real-world tests, focusing on functionality and adaptability in marine environments.

This project, supported by Graines d'Entrepreneurs, contributes to environmental sustainability and Tunisia's entrepreneurial ecosystem by empowering young innovators. The MVP serves as a proof-of-concept, demonstrating the potential of autonomous systems to address marine pollution. This report outlines the development process, from design to testing, providing a foundation for future enhancements in autonomous marine robotics.

The report is structured as follows: Chapter 1 introduces Graines d'Entrepreneurs and the environmental challenge of waste pollution in marinas, rivers, and lakes, comparing existing solutions like Seabin and WasteShark to justify the need for an autonomous MVP. Chapter 2 outlines the MVP's system design, detailing the catamaran's hardware, ROS Noetic-based software, and mechanical components optimized for stability and efficiency. Chapter 3 describes the development and testing of autonomous waste collection missions, covering Gazebo simulations, hardware integration, and real-world trials to validate performance. The conclusion evaluates the MVP's success in demonstrating feasibility, addresses challenges like dynamic water navigation, and suggests future enhancements like multi-boat systems, supported by a bibliography of technical and marine robotics references.

Chapter

1 Graines Academy : Overview and Project Context

Contents

1.1 Introduction

Rapid accumulation of marine debris, particularly plastics, poses a significant threat to oceanic ecosystems, coastal economies, and global sustainability. The Ocean Cleaner project aims to address this critical environmental challenge by developing an autonomous surface vehicle capable of navigating coastal and open ocean waters to efficiently collect waste. This project is undertaken in collaboration with Graines d'Entrepreneurs, a Tunisian startup dedicated to fostering entrepreneurial skills and innovative thinking among young people. Using advanced technologies such as autonomous navigation and sustainable power systems, the project seeks to create a scalable solution for the collection of marine waste. This chapter introduces the host company, outlines the problem of ocean pollution, reviews existing solutions, and details the key technologies and objectives of the Ocean Cleaner project.

1.2 Host Company

1.2.1 Company Overview

Graines d'Entrepreneurs is a Tunisian startup founded in 2018 and headquartered in La Marsa, Tunis. The organization is dedicated to fostering an entrepreneurial spirit among young people by providing them with opportunities to develop innovative ideas and projects. As a social enterprise, Graines d'Entrepreneurs places a strong emphasis on experiential learning, creativity, and practical education. Through a variety of interactive programs, including workshops, hands-on training sessions, and project-based challenges, the company inspires and empowers youth to explore entrepreneurship and leadership. By connecting participants with experienced mentors, real-world problems, and a collaborative community, Graines d'Entrepreneurs equips aspiring young leaders with the mindset, problem solving abilities, and skills necessary for success in the rapidly evolving world of entrepreneurship.

Figure 1.1: Graines d'Entrepreneurs LOGO

1.2.2 Domain of Activity

The core activity of Graines d'Entrepreneurs is entrepreneurship education. The startup organizes workshops, training sessions, and incubation programs designed for children, teenagers, and young adults. Their programs typically include hands-on learning experiences, professional mentoring, and opportunities to develop real-world projects. The objective is to cultivate creativity, leadership, and problem solving skills in young people, helping to create a dynamic entrepreneurial ecosystem in Tunisia and beyond.

1.2.3 Company Products and Services

Graines d'Entrepreneurs offers a range of educational programs tailored for young learners:

Weekly Workshops: Interactive sessions that guide students through the process of ideation, project planning, and pitching, using tools like the Business Model Canvas. These workshops simulate the stages of creating a startup or social initiative, fostering skills applicable to projects such as The Ocean Cleaner.

Startup Days and Bootcamps: Intensive programs, such as Junior Startup Day and the Seeds4Tomorrow initiative, where participants develop and pitch innovative projects in a hackathon-style format. These events often focus on real-world challenges, including environmental issues.

Training for Trainers: Programs to train educators and coaches in active pedagogy, positive education, child psychology, and entrepreneurial coaching, ensuring scalable impact across schools and communities.

Customized Educational Content: Collaborations with schools and regional institutions to integrate entrepreneurship into curricula, supported by partnerships with organizations like the Swiss Ed TECH Collider and regional economic promotion bodies. These offerings provide a robust platform for developing innovative projects like The Ocean Cleaner, enabling young entrepreneurs to apply their skills to environmental challenges through structured guidance and mentorship.

1.3 Review of Key Technologies and Components for Autonomous Navigation Boats

1.3.1 Historical Overview of Autonomous Robots

The development of autonomous robots, including those designed for marine environments, has evolved significantly over the past century. The concept of autonomy in robotics traces back to the 1940s, when William Grey Walter created the first autonomous robots, known as "turtles," which used simple sensors and circuits to navigate their environment. These early robots laid the groundwork for autonomous systems by demonstrating basic obstacle avoidance and self-directed movement.

In the 1960s, advancements in computing enabled more complex robots. The Stanford Research Institute's Shakey robot, developed between 1966 and 1972, was a landmark, combining sensors and planning algorithms to navigate indoor spaces. Marine robotics, however, lagged until the 1980s, when remotely operated vehicles (ROVs) equipped with sonar and cameras became common for underwater exploration. Though human-controlled, ROVs laid the foundation for autonomous marine systems.

The 1990s ushered in true autonomy for marine robotics, with MIT's 1993 "Sea Grant" AUV using sonar and basic navigation algorithms for underwater missions. Surface-based autonomous boats also emerged, powered by advances in GPS, inertial navigation, and computing, followed by the U.S. Navy's autonomous surface vessels in the 2000s, which integrated LiDAR and radar for tasks like mine detection.

By the 2010s, autonomous boats gained momentum in research and commercial applications, spurred by events like the 2014 Maritime RobotX Challenge, which drove innovation in obstacle avoidance and mapping. Today, these boats use AI, machine learning, and sensor fusion for robust navigation, supporting tasks from environmental monitoring to cargo transport.

1.3.2 Sensor Technologies

Sensor technologies are critical for enabling autonomous boats to perceive their environment, localize themselves, and navigate safely. The primary sensors used in autonomous navigation boats include LiDAR, GPS, and inertial measurement units (IMUs).

1.3.2.1 Lidar

LiDAR (Light Detection and Ranging) provides high-resolution environmental mapping, essential for safe navigation in cluttered or low-visibility marine settings. It detects obstacles such as other vessels, docks, or floating debris, enabling the boat to plan collision-free paths. In this project, LiDAR serves as a primary tool for obstacle detection and spatial awareness, particularly in narrow waterways or areas with poor visibility like fog or nighttime. **Key Aspects of Lidar Include:**

- **High-precision Obstacle Detection:** Accurately identifies obstacles such as other vessels, docks, and floating debris, allowing for safe navigation and collision avoidance.
- **Environmental Mapping:** Generates detailed 2D spatial maps, enabling the boat to operate in cluttered or narrow waterways.

- **All-weather Operation:** Performs reliably in various lighting and weather conditions, including fog, shadows, and nighttime, where traditional vision systems may fail.

1.3.2.2 IMU (Inertial Measurement Unit)

The Inertial Measurement Unit (IMU) tracks the boat's motion and orientation, ensuring stable navigation in the dynamic marine environment affected by waves, currents, and wind. It provides real-time data on acceleration, angular velocity, and attitude (roll, pitch, yaw), supporting precise maneuvering. In this project, the IMU integrates with other sensors to maintain course accuracy and execute tasks like aligning with waste targets or docking. **Key Aspects of IMU Include:**

- **Stabilization and Control:** Provides real-time data on acceleration and angular velocity to stabilize navigation and maintain accurate heading.
- **Dynamic Compensation:** Compensates for the effects of waves, currents, and wind, ensuring consistent performance in challenging marine environments.

1.3.2.3 GPS (Global Positioning System)

The Global Positioning System (GPS) delivers geolocation data, allowing the boat to determine its absolute position in open waters. It is vital for long-range navigation, enabling the boat to follow predefined waypoints or cover designated cleaning zones systematically. In this project, GPS enhances localization accuracy when combined with other sensor data, ensuring efficient navigation across large marine areas for comprehensive waste collection.

Key Aspects of GPS Include:

- **Global Positioning:** Provides accurate latitude, longitude, and altitude information for route planning and waypoint following across large water bodies.
- **Resilient Localization:** Maintains reliable navigation even when local visual references are unavailable, and supports sensor fusion with IMU for drift correction.

- **Autonomous Mission Execution:** Enables the boat to autonomously follow predefined routes and reach target destinations with minimal human intervention.

1.3.3 Mapping and Localization Methods

Mapping and localization are foundational capabilities in autonomous robotic systems, enabling them to perceive, interpret, and navigate within their operating environments. These processes are essential for safe, reliable, and efficient autonomous movement, particularly in unknown or dynamically changing settings.

Mapping involves constructing a spatial representation of the environment using sensor data. This map may be two-dimensional (2D), depending on the application and available sensors. Mapping enables the robot to understand environmental features, identify obstacles, and define navigable regions.

Localization refers to the robot's ability to determine its position and orientation (pose) relative to the map. Accurate localization ensures that the robot can track its motion over time, follow planned paths, and make context-aware decisions.

One of the most widely used approaches is **Simultaneous Localization and Mapping (SLAM)**. SLAM allows a robot to build a map of an unknown environment while concurrently estimating its own pose within that map. SLAM techniques process data from sensors such as LiDAR, cameras, IMUs, and GPS to iteratively refine both the map and the estimated location [1].

1.3.4 Path Planning

Path planning is a fundamental aspect of autonomous navigation, enabling robots to compute safe and efficient routes toward their destinations. It ensures the robot can navigate through complex environments while avoiding obstacles, respecting motion constraints, and optimizing overall performance.It aims to find an optimal path that minimizes distance, time, while

avoiding static obstacles and respecting environmental boundaries. To function reliably, autonomous path planning must address multiple essential challenges:

Obstacle Avoidance: Continuously detects and avoids both static and dynamic obstacles using data from sensors like LiDAR.

Dynamic Adaptation: Responds in real time to changes in the environment by recalculating safe and feasible paths.

Motion Constraints: Accounts for the robot's physical limitations, including turning radius, acceleration, and speed, to ensure the path is executable.

Path Efficiency: Selects paths that align with operational goals, such as minimizing time, energy consumption, or exposure to risk.

Autonomous navigation relies on integrating sensors, mapping, localization, and path planning to enable boats to operate independently in marine environments. These technologies address challenges like obstacle avoidance and environmental uncertainty. Together, they empower robust autonomous boat systems for diverse real-world maritime applications.

1.4 Project Context and Aims

1.4.1 Problematic

The escalating crisis of ocean pollution, driven by an estimated 8–14 million metric tons of plastic entering marine environments annually, poses severe threats to marine ecosystems, biodiversity, and human livelihoods. Plastic debris, including plastic garbage and microplastics, accumulates in coastal waters, ocean gyres, and remote marine regions, leading to habitat destruction, entanglement of marine life, and ingestion-related fatalities. Current waste collection methods face significant challenges:

Limited Coverage: Manual cleanups and crewed vessels are restricted to accessible coastal areas, leaving vast open ocean regions unaddressed. High Operational Costs: Large-scale systems like The Ocean Cleanup's Interceptor require substantial infrastructure

and maintenance, making them cost-prohibitive for widespread deployment, particularly in developing regions.

Environmental Footprint: Many existing solutions rely on fossil fuel-powered vessels, contributing to greenhouse gas emissions and contradicting sustainability goals.

Autonomous Navigation Challenges: Existing autonomous surface vehicles (ASVs) often lack robust navigation systems to handle dynamic marine conditions (e.g., currents, waves, obstacles) and advanced waste detection capabilities to identify diverse debris types efficiently. These issues underscore the need for an autonomous, cost-effective, and environmentally friendly solution to enhance marine waste collection, particularly in diverse and challenging marine environments.

1.4.2 Study of Existing Solutions

A diverse array of solutions has emerged over the years to tackle the persistent problem of aquatic waste collection. These approaches span from traditional manual cleanup operations and community-led initiatives to cutting-edge robotic systems and autonomous vehicles, reflecting both the urgency of the issue and the rapid evolution of technology in this field. The study of existing solutions is an essential step, as it allows us to identify the strengths and weaknesses of current projects. This will help us in the development of our project. We have chosen to analyze two existing projects:

- Project «The Seabin»
- Project «The WasteShark Boat»

1.4.2.1 The Seabin

The Seabin Project offers a practical and stationary solution for collecting floating debris in controlled aquatic environments such as marinas, docks, and yacht clubs. Designed for simplicity and efficiency, the Seabin system utilizes a submersible water pump to continuously draw in surface water along with surrounding floating waste. This water, laden with debris, is

directed into a fine mesh catch basket housed inside the device. As the water passes through the basket, solid waste is retained, while the filtered, cleaner water is expelled back into the surrounding harbor. As illustrated in Figure 1.2, the Seabin operates by passively filtering waste from the surface layer of the water, making it an effective tool in calm and enclosed waterfront environments [2].

Capable of collecting up to 20 kilograms of waste at a time, the Seabin must be emptied manually at regular intervals to maintain effectiveness. The system performs optimally in calm, sheltered waters where it can remain undisturbed and continuously powered. However, it requires a nearby electrical connection and ongoing maintenance to ensure consistent operation and prevent clogging or pump degradation. Despite these constraints, the Seabin provides a reliable, localized solution for addressing visible pollution in enclosed or semi-enclosed waterfront areas.

Figure 1.2: The Seabin Project.

Unlike the stationary Seabin, my project introduces an autonomous, mobile boat that can travel across rivers and lakes, even in choppy or changing conditions. Thanks to its ability to navigate and adapt, it can reach trash in places where fixed devices can't operate. This flexibility makes it a more versatile and responsive solution for real-time water cleanup.

Comparison with the Seabin Project

The table below outlines the main differences between the two systems:

Table 1.1: Comparison between the Seabin and Our Autonomous Cleaning Boat

Feature	**Seabin Project**	**Our Autonomous Boat**
Mobility	Stationary	Fully mobile
Debris Collection Method	Water and surface debris pulled into mesh basket by pump	Waste detected, tracked, and collected using onboard net system
Autonomy	Requires manual setup and maintenance	Operates autonomously, tracks and collects waste without human intervention
Location Suitability	Sheltered, powered areas only (like marinas)	Capable of operating in dynamic environments
Maintenance	Must be manually emptied and monitored	Periodic maintenance
Total Design Weight (Kg)	55	7.29734
Control	Manual setup only	Remotely controllable with autonomous features

As shown in Table 1.1, our project provides several functional advantages over the Seabin. While the Seabin is limited to static use in specific areas, our boat introduces **mobility**, **real-time waste tracking**, and **automated collection**. Its lightweight and compact design make it suitable for navigating tight or remote environments. Additionally, the autonomous capabilities reduce human intervention and extend operational range, making our solution more adaptive and scalable for widespread water cleanup operations.

1.4.2.2 The WasteShark Boat

The WasteShark is an environmentally friendly aquatic drone designed to remove floating waste from urban waterways, ports, and marinas. Inspired by the whale shark's filter-feeding technique, it is available in both remote-controlled and fully autonomous versions, providing a flexible solution for surface water cleanup [3].

It moves across the water's surface, collecting floating debris into an onboard basket with a capacity of up to 200 liters, as shown in Figure 1.3 . WasteShark can operate on preset routes or be steered manually and is equipped with obstacle-avoidance sensors for optimized cleaning. Once full, the basket must be manually emptied before reuse. Depending on the model, the cost ranges from $17,000 to $23,600.

Figure 1.3: The WasteShark Boat.

Comparison with the WasteShark Project

In contrast, our autonomous boat presents a more compact, lightweight, and cost-effective alternative designed to clean water bodies in real-time. The comparison is detailed in the table below:

Table 1.2: Comparison between WasteShark and Our Autonomous Cleaning Boat

Feature	**WasteShark**	**Our Autonomous Boat**
Mobility	Mobile	Fully mobile
Debris Collection Method	Floating debris collected in onboard basket	Trash tracked and collected with a net system
Autonomy	Available in manual and autonomous versions	Fully autonomous with manual override
Navigation	Preset paths and manual steering; uses obstacle sensors	Autonomous path planning and waste tracking
Waste Capacity	Up to 200 liters (approx. 200 kg)	10 kg net capacity
Operational Environment	Urban waterways, ports, and marinas	Rivers, lakes, marinas, and natural environments
Weight (Kg)	72	7.29734
Total Boat Dimensions (mm)	Approx. 1570 × 1090 × 540 (typical WasteShark spec)	630 × 756 × 423
Price Range	$17,000 – $23,600	Approx. $1,500

As shown in Table 1.2, the WasteShark, with a weight of approximately 72 kg, is a relatively large and heavy device. Its deployment and retrieval in marinas or ports typically require lifting equipment and regular maintenance, which can be costly and logistically demanding. In contrast, our autonomous boat weighs only 7.3 kg, making it over ten times lighter. This lightweight structure, combined with a compact form factor and a much lower cost of around $1,5000, enables quick and easy handling, even in remote or constrained environments. Moreover, its autonomous capabilities and real-time adaptability allow it to access hard-to-reach areas such as narrow riverbanks, small lakes, and natural water bodies, offering a practical and scalable solution for localized water cleanup operations.

1.4.3 Proposed Solutions

To address the limitations of existing solutions, such as the stationary operation of the Seabin and the high cost and weight of the WasteShark, the **Ocean Cleaner Project** proposes a lightweight, cost-effective, and fully autonomous surface robot (ASV). This vehicle is designed for versatile waste collection in rivers, lakes, and marinas. The following solutions integrate advanced technologies and sustainable engineering to enhance mobility, autonomy, and scalability. The proposed solution is based on the following key components and innovations:

- **Mechanical Design:** The prototype vessel features a mechanical design specifically aimed at providing stability and ensuring smooth navigation in various water conditions.
- **Autonomous Navigation:** The boat will utilize advanced navigation technologies, including Lidar, IMU, and computer vision, to plan and follow optimal paths while avoiding obstacles. This will enable efficient coverage of polluted areas with minimal human intervention.
- **Intelligent Waste Detection:** Onboard cameras and sensors, integrated with machine learning algorithms (such as YOLO for object detection), will allow for real-time identification and localization of floating waste.
- **Automated Collection Mechanism:** A specially designed mechanical system will be installed to collect and store various types of debris encountered during the mission, ensuring effective removal and secure storage until proper disposal.
- **Remote Monitoring and Data Collection:** The vessel will transmit live status updates and environmental data to a remote monitoring station, allowing for real-time mission control and the collection of valuable insights on pollution patterns.

1.5 Project Methodology

The development of the Autonomous Cleaner Boat Minimum Viable Product (MVP) for debris collection in marinas, rivers, and lakes adhered to the Waterfall model [4], a linear methodology ensuring each phase was fully completed before advancing. Executed in partnership with Graines d'Entrepreneurs, this approach facilitated a disciplined progression from conceptualization to validation, as detailed below:

1. **Requirements Analysis:** The project started by pinpointing the need to tackle waste pollution in marinas, rivers, and lakes. Working with Graines d'Entrepreneurs, we outlined goals for a boat that could navigate on its own, spot debris, and collect it efficiently. We studied solutions like Seabin and WasteShark to set practical targets for affordability and performance.
2. **System Design:** Next, we planned the boat's structure, choosing parts and tools to meet our goals. We selected a Raspberry Pi 4, sensors like LiDAR and GPS, and thrusters for movement, all suited for water environments. The software was built on ROS1 Noetic for control, and a catamaran shape with ABS material was designed for balance and durability.
3. **Implementation:** In this stage, we built the boat's systems. We created a virtual model using ROS1 Noetic to test navigation and debris detection in a simulated water setting. We also wired the electrical parts, connecting the Raspberry Pi and sensors with a power bank and batteries, using secure cables to handle water exposure and ensure smooth operation.
4. **Testing and Integration:** Finally, we checked the boat's performance. We tested each part (like sensors and motors) individually, then combined them to ensure they worked together. Virtual tests in a simulated environment and real-world trials in marinas or rivers confirmed the boat could navigate and collect debris, with tweaks made to handle challenges like water currents.

This Waterfall methodology ensured a structured development process, delivering a reliable MVP that demonstrates the feasibility of autonomous waste collection, with insights for future scalability in marine environments.

1.6 Conclusion

This chapter has outlined the critical need for innovative solutions to address the rapid accumulation of marine debris in marinas, rivers, and lakes, which threatens ecosystems and sustainability. The Autonomous Cleaner Boat project, developed in collaboration with Graines d'Entrepreneurs, responds to this challenge by leveraging autonomous surface vehicle technology to collect waste efficiently. Supported by the startup's mission to foster youth entrepreneurship, the project integrates advanced technologies, including LiDAR, GPS, IMU, SLAM, and path planning, to enable robust navigation and debris detection. By reviewing existing solutions like Seabin and WasteShark, the project identifies opportunities for a cost-effective, environmentally friendly alternative tailored to confined waterways. This introduction sets the stage for the project's development, emphasizing its potential to contribute to cleaner aquatic environments through innovation and collaboration.

Chapter

2 System Design and Software Architecture

Contents

2.1 Introduction

Autonomous navigation has become a cornerstone of modern mobile robotics, enabling robots and autonomous surface vehicles (ASVs) to operate reliably in complex and dynamic environments. This chapter provides a comprehensive overview of the key principles and system components underpinning autonomous navigation, with a particular focus on Simultaneous Localization and Mapping (SLAM) and localization techniques. The integration of advanced sensor technologies, probabilistic algorithms, and robust control architectures is explored, highlighting their role in achieving precise and adaptive robot movement. Furthermore, the chapter details the practical implementation of these concepts using widely adopted ROS packages such as ‘`slam_gmapping`‘ and ‘AMCL‘, demonstrating how theoretical foundations are translated into real-world solutions for autonomous waste collection in marine environments.

2.2 Hardware Component

2.2.1 Introduction

In any robotics project, hardware serves as the foundation that turns innovative ideas into a functional reality. For our MVP, an autonomous boat designed to collect waste in marine and lake environments, the hardware is the physical core that determines how effectively it can navigate waterways, detect debris, and execute its environmental cleanup mission. Selecting the right hardware isn’t about choosing the most advanced or expensive components, it’s about finding the optimal combination that aligns with Cleaning Boat’s goals, balancing performance, compatibility, cost, and resilience against water and debris.

2.2.2 Processing unit

The processing unit must handle multiple critical tasks: running the Robot Operating System (ROS Noetic) for seamless component integration, executing an AI model for real-time waste detection from camera images, processing sensor data for navigation, and controlling motors for propulsion. To optimize performance and distribute workloads, we propose dual-microcontroller architecture. The primary microcontroller will manage ROS, the AI model, and high-level processing, while a secondary microcontroller will handle GPS data processing and motor control, ensuring efficient task separation and system reliability.

2.2.2.1 Microcomputer Selection

The microcomputer must offer sufficient computational power to run ROS, support AI model inference (e.g., for enhancing camera image quality and waste classification), and process sensor data in real time. We evaluated two popular options: the Raspberry Pi 4 (8GB RAM) and the Raspberry Pi 5 (8GB RAM), both capable of meeting Boat Cleaning's demands in marine environments. The table below compares their key features to determine the best fit.

Table 2.1: Comparison between Raspberry Pi 4 and Pi 5 (both 8GB RAM)

Feature	**Pi 4 (8GB RAM)**	**Pi 5 (8GB RAM)**
CPU	Quad-core Cortex-A72 @ 1.8GHz	Quad-core Cortex A76 @ 2.4GHz
RAM	8GB LPDDR4-3200	LPDDR4X-4267
Ubuntu 20.04 Support	Yes, official support and easy setup	No official support, requires custom setup
ROS Noetic Support	Yes, fully compatible	Yes, but may have compatibility issues

The Raspberry Pi 4 (8GB), as shown in Figure 2.1, is the better choice for running Ubuntu 20.04 and ROS Noetic due to its official support and straightforward installation. The Raspberry Pi 5 (8GB), while offering superior performance, lacks official Ubuntu 20.04 support, potentially complicating ROS Noetic setup. Additionally, there are other micro-computers like the NVIDIA Jetson Nano, but we can't choose them due to their higher cost (approximately $99) and potential setup challenges.

Figure 2.1: The Raspberry Pi4 8GB.

2.2.2.2 Microcontroller Selection

The microcontroller is tasked solely with reading GPS data and controlling motors, operating as a slave to the Raspberry Pi 4. This division of responsibilities allows Raspberry Pi 4 to manage complex computations while the secondary microcontroller handles these specific real-time tasks. By offloading GPS data processing and motor control, the system achieves efficient performance. The Arduino Uno has been selected for this role due to its simplicity and suitability

Table 2.2: Arduino Uno Characteristics

Feature	**Arduino Uno**
CPU	ATmega328P, 16 MHz
Memory	32 KB Flash, 2 KB SRAM, 1 KB EEPROM
I/O Pins	14 digital (6 PWM), 6 analog inputs
Communication	UART, SPI, I2C
GPS Shield Compatibility	Compatible with dedicated shields like the Adafruit Ultimate GPS Shield for seamless GPS integration

The Arduino Uno(see Figure 2.2) efficiently handles Boatcleaninge's GPS data reading (via UART with a Ublox NEO-6M) and motor control (using PWM outputs), acting as a reliable slave to the Raspberry Pi 4. The Uno's simplicity and affordability make it ideal for real-time tasks in marine environments.

Figure 2.2: The Arduino Uno.

2.2.3 Sensor

Sensors are the perceptive tools that allow The MVP to understand its environment and maintain situational awareness. They collect critical data for navigation, obstacle avoidance, and waste detection in challenging aquatic settings.

2.2.3.1 Lidar Selection

To ensure The Robot can effectively navigate and detect obstacles in marine environments, we evaluated three sensor types: RPLIDAR (a LiDAR-based sensor), ultrasonic sensors, and Time-of-Flight (ToF) sensors. The table below compares their key characteristics to determine their suitability for the boat's autonomous waste collection mission.

Table 2.3: Comparison of Distance Sensors for Autonomous Cleaning Boat

Feature	**TOF Sensor**	**Ultrasonic Sensor**	**RPLidar 360°**
Working Principle	Emits infrared light pulses and measures return time for distance or depth mapping.	Emits high-frequency sound waves and measures echo return time for distance.	Emits laser pulses and measures time-of-flight to create a 360° point cloud for mapping.
Range	0.1 m – 12 m	0.02 – 4 m	0.15 – 12 m
Field of View	2°	30–40°	360°
Communication Protocol	UART / I2C	GPIO Timing	UART

Each sensor offers unique benefits for boat cleaning. TOF sensors are accurate for short-range waste detection but limited by a narrow 2° field of view, requiring multiple units. Ultrasonic sensors are useful for nearby obstacles but suffer from limited coverage and sensitivity to water interference. RPLIDAR, with its 360° field of view, provides wide coverage and reliable mapping, making it the most suitable for open-water navigation and obstacle avoidance.

The RPLIDAR(see Figure 2.3) remains the top choice for the MVP due to its 360° field of view and high accuracy, enabling comprehensive environmental mapping and navigation in complex marine settings.

Figure 2.3: The RPLidar A1.

2.2.3.2 Camera Selection

A camera serves as Boat Cleaning ‘s key sensor for visual waste detection, capturing images to identify and classify debris in marine environments. Integrated with an AI model, enabling the boat to accurately target waste during its cleanup mission. I have selected a Raspberry Pi Camera, as it integrates seamlessly with Raspberry Pi 4.

The Raspberry Pi Camera with a 5MP(see Figure 2.4) resolution and 60° field of view is chosen for its ability to deliver clear images optimized for the AI models with resolution

640x640, ensuring precise waste detection. Its focused field of view supports the boat's mission by enabling accurate identification and targeting of debris in marine environments.

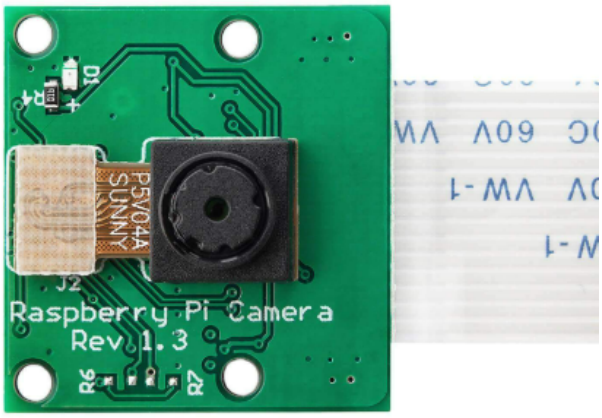

Figure 2.4: The Raspberry Cam V1(5MP).

2.2.3.3 The Imu

The Imu is vital for autonomous navigation, measuring angular velocity to track rotational movements and ensure stability. It compensates for wave-induced motions, enabling precise heading control in challenging marine environments.Two types of IMUs were considered: the 6-axis IMU, which includes an accelerometer and gyroscope, and the 9-axis IMU, which adds a magnetometer for absolute orientation.

While the 6-axis IMU suits basic motion tracking, the 9-axis IMU provides better navigation accuracy by using Earth's magnetic field for heading estimation. For this reason, the 9-axis IMU(see Figure 2.5) was chosen to ensure reliable orientation in challenging marine environments.

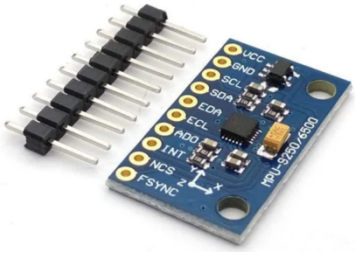

Figure 2.5: The Imu 9-axis.

2.2.3.4 GPS

GPS is essential for the Ocean Cleaner autonomous navigation, enabling the system to follow predefined map points programmed for waste collection routes. It ensures precise self-localization, allowing robots to determine their position in marine environments accurately.

The SIM808 module(see Figure 2.6) was selected for its dual functionality, combining GPS positioning with GSM/GPRS communication. This integration allows the Ocean Cleaner to determine its location accurately. ts high GPS sensitivity ensures reliable performance even in weak signal condition.

Figure 2.6: The SIM808 Module.

2.2.4 Actuators

Actuators are the components that enable the MVP to move and execute its mission in marine environments. They translate control signals from the processing unit into physical actions, ensuring precise navigation and operational efficiency.

To ensure optimal performance of the project, the type of motor selected should possess specific characteristics. We recommend using underwater DC motors that can be seamlessly integrated into the mechanical conception without risking water ingress. These motors should support differential steering, enabling precise navigation and obstacle avoidance in marine environments. Choosing such motors ensures reliable and controlled propulsion, which is essential for the autonomous waste collection mission. The table below compares

the APISQUEEN U2 MINI 1.3Kg 16V 130W and APISQUEEN U01 12V-16V 2Kg 390W thrusters, evaluating their suitability for Ocean Cleaner's propulsion needs:

Table 2.4: Comparison of APISQUEEN U2 MINI and U01 Thrusters

Feature	**APISQUEEN U2 MINI**	**APISQUEEN U01**
Thrust	1.3 kg	2 kg
Voltage	12–16 V (3–4S LiPo)	12–16 V (2–4S LiPo)
Max Power	130 W	390 W
Max Current	8 A	17 A
Weight	210 g	178 g
Arduino Uno Compatibility	Yes, via PWM through motor driver	Yes, via PWM through motor driver

The APISQUEEN U2 MINI 1.3Kg 16V 130W(see Figure 2.7) thrusters were selected for Ocean Cleaner because their lower power consumption (130 W) compared to the U01's 390 W significantly reduces battery demands, addressing critical power supply constraints for the system.

Figure 2.7: The APISQUEEN U2 MINI .

2.2.5 Power System

The power system is vital for the Ocean Cleaner project, ensuring operational efficiency and endurance in marine environments. Initially, a single battery was selected to power all components. However, the motors' high power demands (130 W each, up to 8 A) risked overheating, compromising system reliability.

To address this, a distributed power system was adopted, using dedicated LiPo batteries for each motor and a power bank for the Raspberry Pi and SIM808 module. This approach

distributes the power load, minimizes overheating, and ensures stable operation, enhancing the Ocean Cleaner's effectiveness in marine cleanup missions.

2.2.5.1 LiPo Battery Selection

For the Ocean Cleaner project's propulsion system, we chose the Zeee 4S 14.8V 5200mAh 100C LiPo Battery with EC5 Connector. This battery was selected to meet the high-performance demands of the underwater thrusters while ensuring reliability and efficiency in a marine environment.

• Voltage (14.8V): The 4S configuration delivers 14.8V, which is ideal for powering the motors efficiently, providing the necessary thrust for navigation.

• Capacity (5200mAh): With a 5200mAh capacity, this battery strikes a balance be-tween weight and runtime, allowing Ocean Cleaner to operate for extended peri-ods without adding excessive bulk.

• Discharge Rate (100C): The 100C rating enables the battery to supply up to 520A of current (5200mAh × 100C / 1000), far exceeding the 16A needed for two thrusters (8A each). This ensures consistent performance during high-demand situations like wave navigation or rapid maneuvers.

The Zeee 4S LiPo(see Figure 2.8) battery's robust design and high discharge capability make it an excellent match for Ocean Cleaner's propulsion needs.

Figure 2.8: The Zeee 4S Lipo.

2.2.5.2 Power Bank Selection

For the control electronics of the Ocean Cleaner project, a TECTIN 20000mAh 66W Power Bank(see Figure 2.9) is selected, featuring wired connectivity, 66W power output, 20000mAh battery capacity, USB Type-C input, two USB outputs, and fast charge capability. This power bank is chosen to provide a stable and versatile power source for the Raspberry Pi and SIM808 module. Its key specifications are outlined below:

- **Capacity (20000mAh):** The 20000mAh capacity ensures long-lasting power for the Raspberry Pi and SIM808 module, supporting extended. missions without frequent recharging.
- **Power Output (66W):** The 66W output comfortably meets the power demands of the Raspberry Pi (15–20W), Arduino (1–2W), and SIM808 module (2–10W), with sufficient reserve capacity.
- **Two Ports:** The two USB outputs enable simultaneous powering of the Raspberry Pi and SIM808 module, simplifying the power setup.
- **Fast Charge:** The fast charge capability minimizes downtime by rapidly recharging the power bank between missions

Figure 2.9: The Power Bank.

2.2.6 Conclusion

The careful selection of hardware components ensures the boat meets its performance, reliability, and operational requirements. By evaluating and choosing appropriate materials,

sensors, actuators, and control systems, the design achieves an optimal balance between efficiency, cost, and functionality. This robust hardware foundation supports successful integration with the mechanical and software systems, paving the way for effective prototyping and testing.

2.3 Software Components

This section outlines the software components utilized in the design, simulation, and operation of the waste collection robot. The software stack includes tools for mechanical design validation, robot control, navigation, and waste detection, ensuring a cohesive system capable of autonomous operation in a marine environment.

2.3.1 SolidWorks CAD Software

SolidWorks is a comprehensive computer-aided design (CAD) and computer-aided engineering (CAE) software used for modeling, simulation, and assembly of mechanical systems. Although primarily mechanical, its role in the software workflow of this project was essential.

Figure 2.10: Solidworks Logo.

2.3.1.1 Overview of SolidWorks

In this project, SolidWorks was utilized to create detailed 3D models of the robot's chassis and other mechanical components. Its simulation capabilities, including finite element analysis (FEA) and motion studies, were critical for validating the structural integrity and functionality of the designed components under marine conditions.

2.3.1.2 Applications in the Project

- **3D Modeling and Assembly:** SolidWorks was used to design the individual pieces of the MVP, including the robot's chassis, waste collection mechanism, and mounting interfaces for hardware components. The software's assembly environment enabled precise integration of these components
- **Structural Analysis:** Using SolidWorks Simulation, finite element analysis (FEA) was conducted to evaluate the structural integrity of the robot's shape and components. The analysis accounted for various loads, such as water pressure at different depths and mechanical stresses encountered during navigation.
- **Material Selection Support:** SolidWorks facilitated material selection by allowing simulations with different material properties. This helped optimize the design for corrosion resistance, weight reduction and buoyancy line level.

SolidWorks played a pivotal role in bridging the mechanical design and software integration phases, providing a robust platform for validating the physical design before hardware and software implementation. Furthermore, SolidWorks provides tools that support the next development steps by enabling the creation of accurate 3D models, which can be exported to use in simulation environments.

2.3.2 Robot Operating System (ROS)

2.3.2.1 Overview of ROS

ROS (Robot Operating System) is an open-source framework designed to simplify the development of complex and modular robotic systems. It offers essential tools, libraries, and communication mechanisms that enable developers to build robot software in a structured and scalable manner [5].

Figure 2.11: ROS Logo.

The development environment was set up on Ubuntu 20.04 (Focal Fossa), which is the only officially supported version for ROS1 Noetic. This ensures compatibility with key simulation tools such as Gazebo 11 and RViz, which are essential for modeling physical interactions, testing sensor data, and visualizing system behavior.

2.3.2.2 ROS1 Simulation Tools

Gazebo 11 is a powerful open-source 3D robotics simulator that provides a realistic environment for testing robot models with accurate physics, sensor simulation, and dynamic interactions. It is the final long-term support (LTS) release of the Gazebo classic series and is fully compatible with ROS1 Noetic. Gazebo 11 allows developers to simulate complex scenarios, making it ideal for evaluating robot behavior before physical implementation.

Figure 2.12: Gazebo 11 Logo.

RVIz (ROS Visualization) is a 3D visualization tool for ROS that allows developers to view sensor data, robot models, and coordinate frames in real time. It is especially useful for debugging and understanding how a robot perceives its environment and executes navigation tasks. In this project, RViz was used to visualize the boat's position, orientation, sensor outputs, and planned paths during simulation.

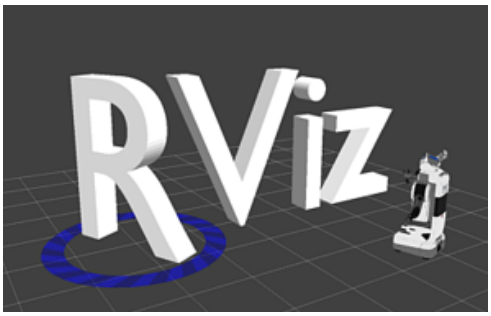

Figure 2.13: RVIz Logo.

2.3.2.3 ROS1 Node Organization and Communication

The fundamental building blocks of any robotic system are called nodes. A node is an independent executable that performs a specific task, such as reading sensor data or controlling actuators. These nodes are designed to work collaboratively by communicating with one another through topics, services, or actions

Language Compatibility for ROS Nodes

ROS1 Noetic provides robust support for developing node files in both Python3 and C++. This flexibility allows developers to choose the programming language that best fits each specific task or their own expertise. Python3 is often preferred for rapid prototyping, scripting, and ease of integration with existing Python libraries, while C++ is ideal for performance-critical applications and low-level hardware interaction.

Role of the ROS Master

The ROS Master acts as the central coordination service in any ROS1 Noetic system. Its main role is to manage the registration of nodes, topics, and services. When a node wants to publish or subscribe to a topic, it first communicates with the ROS Master to register its intent. The master then facilitates the connection between publisher and subscriber nodes by sharing

their network addresses, enabling them to establish a direct peer-to-peer communication link. Although the master is essential for establishing these connections, it does not handle or forward any actual message data once the nodes are connected

Figure 2.14 illustrates this architecture in detail, showing how nodes interact with the ROS Master to initiate communication.

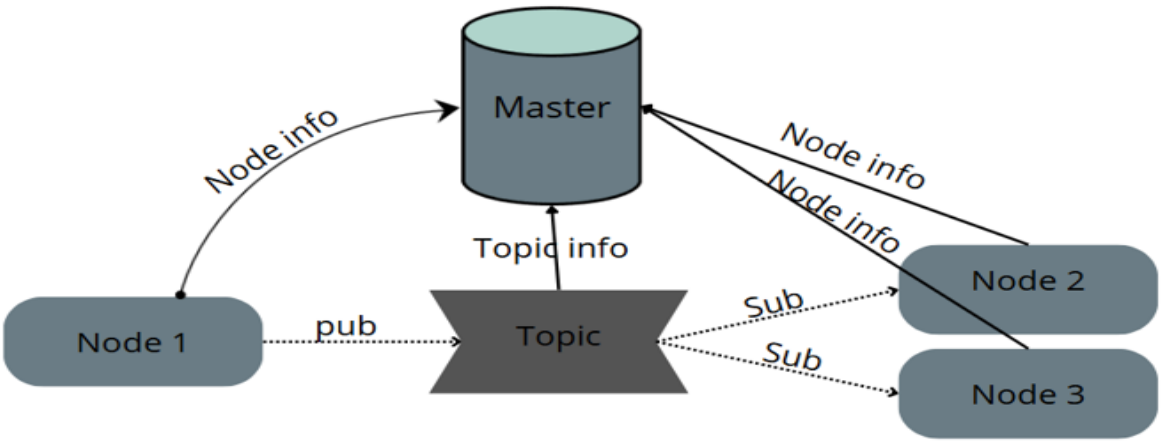

Figure 2.14: ROS Master Architecture.

Communication Between Nodes Communication in ROS1 Noetic is primarily handled using a publish/subscribe model via topics. A node that generates data publishes messages to a named topic, while other nodes that require this data subscribe to the same topic. Figure 2.15 illustrates this process, showing how nodes exchange messages through a shared topic channel.

Figure 2.15: Communication Between Nodes.

Importance of Handling Multiple Publishers on One Topic When multiple nodes publish to the same topic, such as a velocity command topic, the subscriber will always receive only the latest message sent. ROS does not provide built-in arbitration or prioritization between publishers.

2.3.2.4 ROS Packages

The waste collection robot's software architecture leverages several standard ROS1 Noetic packages to enable autonomous navigation, localization, and mapping in a marine environment. The following packages were selected for their robust functionality and compatibility with the project's requirements. Figure 2.16 illustrates the overall organization of the ROS packages employed in this system.

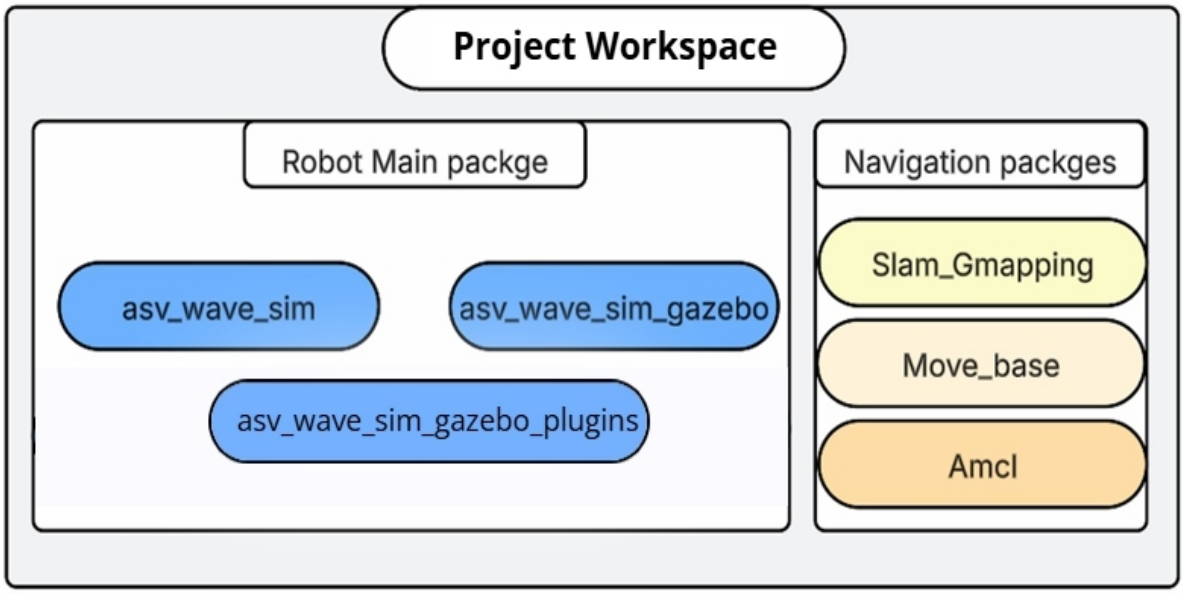

Figure 2.16: Ros Package Hierarchy Overview.

`asv_wave_sim`**:** This package simulates an Autonomous Surface Vehicle (ASV) in a wave-affected water environment within Gazebo [6]. It models water physics, including waves, currents, and buoyancy, using plugins like libHydrodynamicsPlugin.so. In this project, `asv_wave_sim` tested the robot's behavior under maritime conditions, such as hydrodynamic forces and interactions with floating debris. The `asv_wave_sim_gazebo` package is central, enabling the implementation and testing of various functionalities in a simulated environment using ROS tools such as RVIZ and Gazebo. Additionally, the `asv_wave_sim_gazebo_plugins` package incorporates pre-existing code to simulate water physics and ocean behavior. These packages streamline development and minimize risks of material damage or unforeseen errors during early testing phases.

`move_base`**:** This package provides a high-level interface for autonomous navigation, combining global and local planners for path planning and obstacle avoidance. It used for local obstacle avoidance, integrating with costmaps configured via YAML files to handle static and dynamic obstacles. Recovery behaviors, like rotating in place, ensured robust navigation in cluttered waters.

amcl: The Adaptive Monte Carlo Localization (AMCL) package enabled probabilistic localization, estimating the robot's pose within a known map using sensor data (e.g., sonar, LiDAR) and odometry. AMCL employed particle filtering to localize the boat accurately, integrating GPS and IMU data, and was configured to align with the robot's base link and map frame.

`slam_gmapping`**:** This package facilitated Simultaneous Localization and Mapping (SLAM), building 2D occupancy grid maps of unknown marine environments while tracking the robot's position. Using laser scan data and odometry, it employed a particle filter algorithm, configured via YAML files for map resolution and update rates, to support navigation and localization tasks. These packages provided a robust framework for simulating and controlling the waste collection robot, ensuring adaptability to marine conditions and supporting seamless transitions from simulation to real-world deployment.

2.3.2.5 File Model Types Used in ROS

The development and simulation of the ASV within the ROS ecosystem required the use of several specialized file formats, each serving a distinct role in robot modeling, visualization, and simulation. The following file types were utilized in this project:

- **URDF (Unified Robot Description Format):** This file serves as the primary description of the ASV's physical configuration, including its links, joints, sensors, and coordinate frames. This XML-based format allows for the modular construction of the robot model and references external mesh files for both visualization and collision detection.
- **SDF (Simulation Description Format):** These files are utilized primarily within the Gazebo simulator to provide a more detailed and flexible representation of both the robot

and the simulation environment. SDF supports advanced features such as sensor plugins, environmental properties, and extended physics configurations.

2.3.3 Navigation Stack

This section explores the key concepts, algorithms, and tools used in autonomous robot navigation, focusing on perception, localization, mapping, and path planning. It covers autonomous navigation frameworks, Simultaneous Localization and Mapping (SLAM), localization techniques, and trajectory planning algorithms, with an emphasis on their implementation in the Robot Operating System (ROS).

2.3.3.1 Autonomous Navigation

Autonomous navigation enables robots to move through environments without human intervention, relying on sensor data, path planning, and obstacle avoidance. Key components include global and local planners, sensor integration (e.g., LIDAR, cameras), and ROS-based frameworks like `move_base`.

- Overview of global and local path planning.
- Sensor requirements and data processing.

This diagram illustrates the core components of an autonomous navigation system for a mobile robot or ASV. The process begins with localization, where sensors such as GPS and IMU determine the robot's position. Mapping modules then build an environmental map using data from sensors like LiDAR and cameras. Path planning algorithms compute optimal trajectories to reach the goal while avoiding obstacles. Finally, the control module adjusts the robot's actuators to follow the planned path. The entire system relies on continuous sensor data to update localization and mapping, enabling robust and adaptive autonomous movement.

The figure 2.17 illustrates this process.

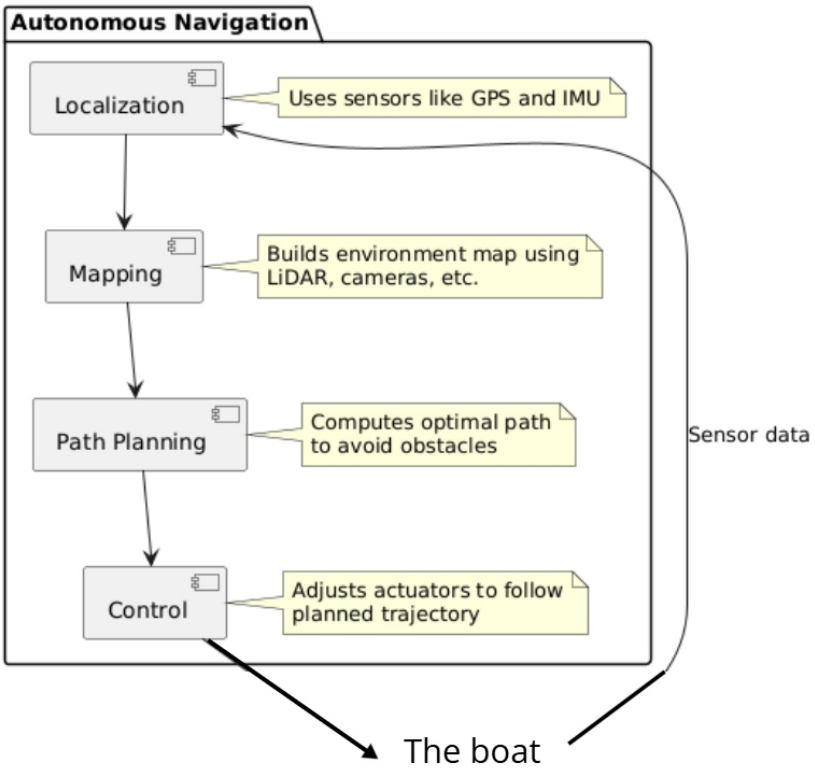

Figure 2.17: Autonomous Navigation Flow Diagram

2.3.3.2 SLAM and Localization

Simultaneous Localization and Mapping (SLAM) enables a robot to construct a map of an unknown environment while determining its own pose. Localization focuses on accurately estimating the robot's position within a known map, which is essential for reliable autonomous navigation.

In this project, SLAM and localization were implemented using the `slam_gmapping` and AMCL ROS packages, respectively. These packages provided the essential capabilities for environment mapping and real-time, probabilistic pose estimation, allowing the robot to operate effectively in dynamic marine conditions. The integration of these tools ensured robust navigation, adaptability, and a seamless transition from simulation to real-world deployment.

Gmapping Integration in ROS: Gmapping is a widely used ROS package for 2D LiDAR-based Simultaneous Localization and Mapping (SLAM). The integration process in a ROS-based robotic system involves several coordinated steps, illustrated and numbered as in Figure 2.18:

1) **Sensor Node:** The process starts with the sensor node, which collects raw data from sensors such as LiDAR.
2) **Teleoperation Node:** The `teleop_node` enables remote control or manual driving of the robot, publishing velocity commands.
3) **Robot Control and Odometry:** The `robot_control` node receives velocity commands and manages the robot's movement. It also publishes odometry data as the robot moves, which is essential for mapping.
4) **Gmapping Node:** The core gmapping node subscribes to both the laser scan and the odometry (via tf transforms) to perform SLAM. It processes the incoming data to generate a real-time 2D occupancy grid map.
5) **Map Server:** Once mapping is complete, the `map_server` node is used to save the generated map files (map.pgm and map.yaml), making them available for future localization and navigation tasks.

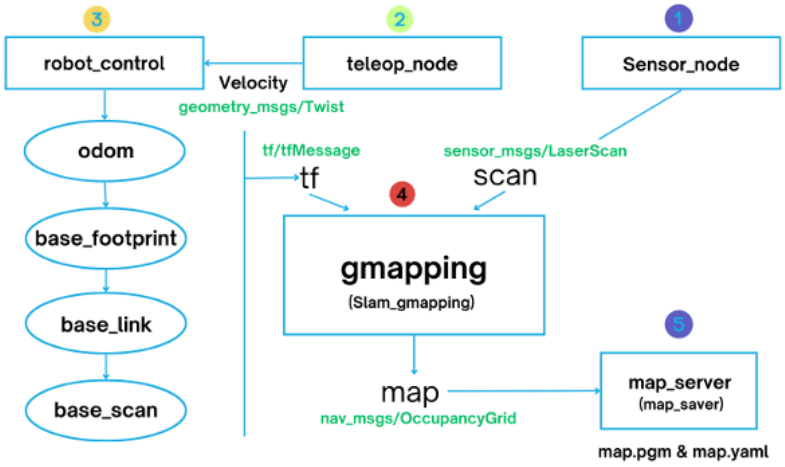

Figure 2.18: Gmapping ROS Integration Diagram

2.4 Mechanical Design

2.4.1 Design Overview

The mechanical concept at the core of this project is focused on designing an autonomous boat with exceptional stability, efficient movement, and reliable waste collection capability.

Each design decision aims to fulfill the essential requirements: maintaining steady buoyancy, enabling smooth navigation, and securely supporting the waste collection net and associated systems.

2.4.2 Catamaran Configuration and Benefits

The adoption of a catamaran configuration provides superior stability compared to monohull designs, owing to its broader beam and dual hulls, which distribute buoyancy across a larger area. This reduces rolling and pitching, particularly when the waste collection net at the stern accumulates debris, causing uneven weight distribution. According to Carlton (2018), the catamaran's wide stance enhances initial stability, minimizing rolling under shifting loads or wave action [7]. Experimental studies confirm that catamarans can safely handle heavier or uneven loads, critical for autonomous operation [8]. Key advantages include:

- **Enhanced Stability:** Twin hulls evenly distribute weight, ensuring the boat remains upright in rough or unevenly loaded conditions.
- **Efficient Navigation:** Reduced hydrodynamic resistance and minimized rolling support predictable and energy-efficient movement.
- **Increased Load Capacity:** The wide stance allows the boat to carry substantial loads, such as a full collection net, without compromising stability.

2.4.3 Boat Model and Component Layout

Figure 2.19 illustrates a 3D model of the autonomous waste-collecting boat, showcasing its catamaran configuration and key structural features. The model highlights the twin-hull design, a central platform housing electronics and propulsion systems, and a dedicated stern area for the waste collection net. This layout optimizes space utilization, ensuring that navigation and propulsion mechanisms remain unhindered while the net efficiently captures debris. The central platform provides a stable mounting surface for critical components, facilitating integration with the hardware and software systems.

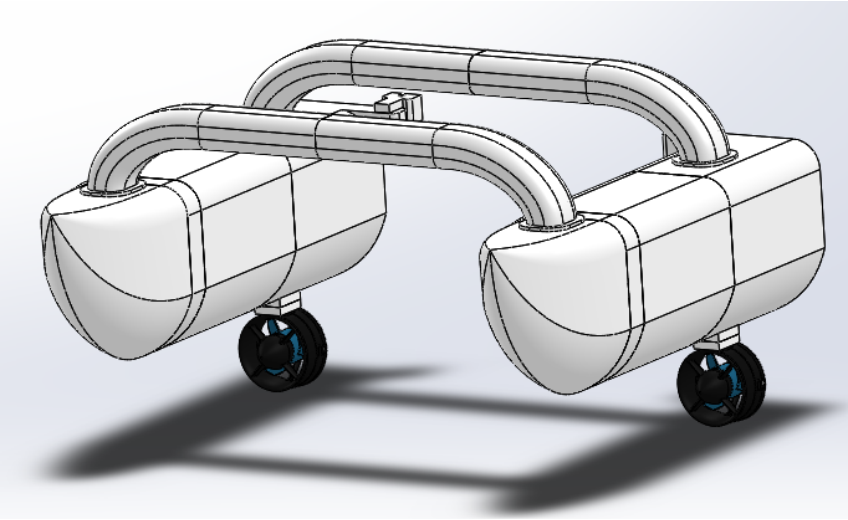

Figure 2.19: 3D model of the autonomous catamaran waste-collecting boat

2.4.4 Structural Design and Validation

The structural design of the boat ensures robustness, hydrodynamic efficiency, and compliance with marine standards, validated through simulations and engineering analyses. The following subsections detail the hull material selection, geometric design, hydrodynamic performance, and structural integrity of critical components.

2.4.4.1 Hull Material and Thickness

The boat's hull is constructed from acrylonitrile butadiene styrene (ABS) plastic with a uniform wall thickness of 4 mm. ABS is selected for its durability, impact resistance, and suitability for marine environments, as widely recognized in small-scale watercraft construction [9]. This reference indicate that ABS hulls with thicknesses between 3 mm and 6 mm provide sufficient mechanical strength for typical operational loads. Given this alignment with industry standards, detailed structural analysis of the hull thickness was deemed unnecessary, confirming the material's adequacy for the boat's requirements.

2.4.4.2 Bow and Hull Geometry

The bow features an elliptical cross-section,as shown in Figure 2.20, to minimize hydrodynamic drag, allowing water to flow smoothly around the vessel and enhancing

propulsion efficiency. This geometry, compliant with hydrostatic principles and marine standards such as ISO 12217-1 [10], ensures stable buoyancy and balance under varying loads. Smooth surfaces and rounded transitions throughout the hull reduce frictional losses, while careful weight distribution maintains adequate freeboard and transverse stability, critical for autonomous waste collection tasks [11].

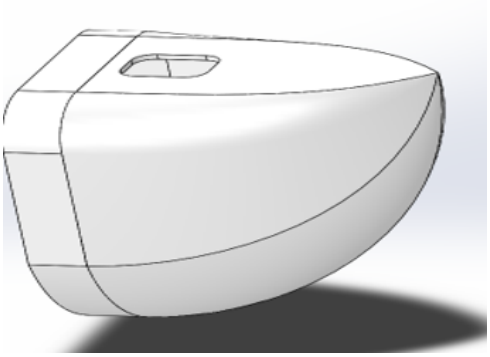

Figure 2.20: Elliptical bow cross-section

2.4.4.3 Hydrodynamic Flow Analysis

To validate the boat's hydrodynamic performance, an external flow simulation was conducted using SolidWorks Flow Simulation. The boat's buoyancy line (waterline) was determined by applying Archimedes' principle, ensuring the hull displaces a volume of water equal to the vessel's total mass. The total mass comprises the boat's material weight (2,161 g) and the operational load (5,136.34 g), yielding a combined loaded weight. The hull's total volume, measured from the SolidWorks 3D model, is 5,215,112.46 mm³ (5.215 liters), resulting in an equilibrium draft of approximately 41 mm, validated by the simulation's free surface results.

The simulation utilized an "External" analysis type, excluding internal spaces, with a transient analysis over 10 seconds under standard Earth gravity. The "Free Surface" option modeled the water-air interface, and an inlet water velocity of 4 knots (2.06 m/s) was set to evaluate fluid behaviour at high operational speeds. The calculation domain defined the virtual space surrounding the boat for accurate analysis.

Pressure Distribution: Figure 2.21 illustrates the pressure distribution on the catamaran hulls at 4 knots, with red areas indicating high-pressure zones and blue-to-green areas

representing lower pressure. The balanced pressure distribution confirms the dual-hull design's stability under dynamic conditions, though peak pressure zones suggest potential stress points requiring reinforcement.

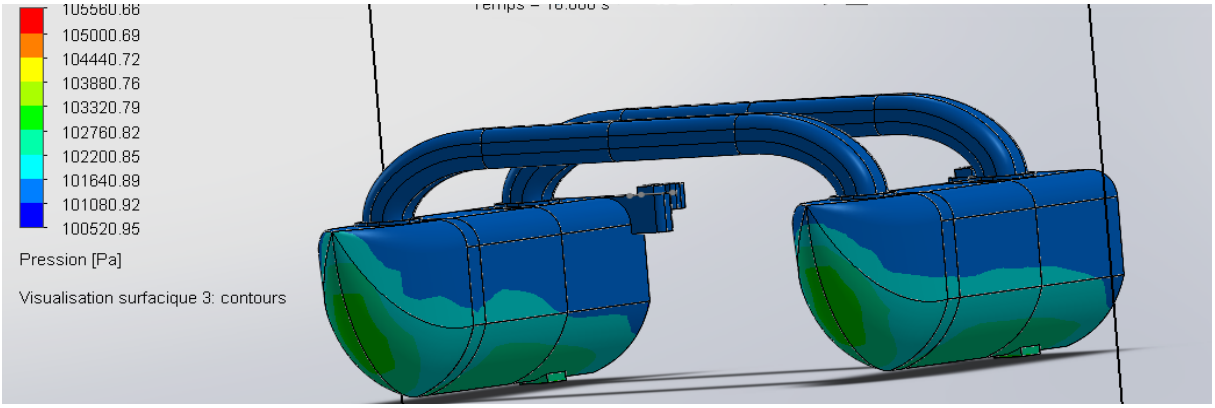

Figure 2.21: Pressure Distribution on Hulls.

Velocity distribution: As shown in Figure 2.22, the velocity distribution reveals fluid dynamics with velocities reaching up to 5.334 m/s near the bow and along the hull. These high-velocity regions indicate areas of reduced pressure that influence drag. The results affirm the hull's hydrodynamic efficiency, promoting smooth water flow and low resistance. However, the high-velocity zones near the bow suggest opportunities for further optimization to reduce drag and enhance movement efficiency.

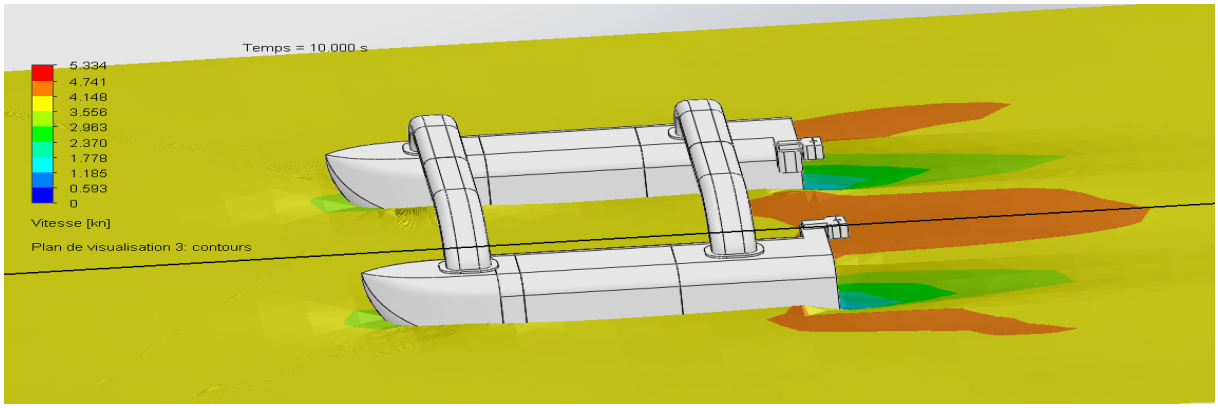

Figure 2.22: Cut Plot – Velocity Distribution.

Net Support Finite Element Analysis: To ensure that the net support structure can withstand a load of 10 kg, a static finite element analysis (FEA) was performed using SolidWorks Simulation. The support, made of ABS plastic and fixed to the stern of the boat, was subjected to a pressure load of 98.6 N/m² directed rearward, as indicated by the red arrow in

Figure 2.23. The attachment point was constrained (fixed), denoted by a green arrow. The von Mises stress distribution ranged from 1.67×10^{-8} N/m² (minimum, blue) to 7.51 × 10³ N/m² (maximum, red), with most areas showing around 3.753 × 10³ N/m² (green). These stresses are significantly below the ABS yield strength, which confirms the ability of the support to handle the load without material failure or excessive deformation.

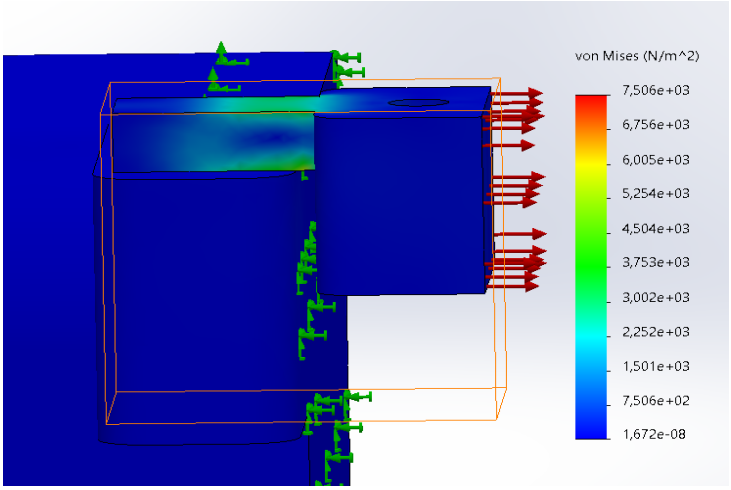

Figure 2.23: Static simulation results for net support structure.

2.4.5 Conclusion

Mechanical design establishes a solid foundation for the performance and reliability of the autonomous boat. Through strategic material selection, structural validation, and iterative CAD modeling, the design meets the project's objectives of strength, stability, and operational efficiency. The catamaran configuration, optimized hull geometry, and reinforced net support ensure that the boat can effectively collect waste in dynamic marine environments. These validated mechanical components pave the way for successful prototyping, testing, and deployment, as outlined in the subsequent sections.

2.5 Conclusion

The system design and software architecture detailed in this chapter establish a comprehensive framework for the Ocean Cleaner autonomous boat, integrating advanced hardware,

sophisticated software, and robust mechanical design. The carefully selected hardware components, including the Raspberry Pi 4, RPLIDAR, and APISQUEEN thrusters, ensure efficient processing, precise navigation, and reliable propulsion in marine environments. The software stack, built on ROS Noetic and supported by tools such as SolidWorks and Gazebo, enables the seamless integration of the navigation, mapping, and waste detection functionalities. Mechanical design, with its catamaran configuration and validated structural components, guarantees stability and operational efficiency under varying conditions. Together, these elements fulfill the project's objectives of creating a lightweight, cost-effective and autonomous solution for marine waste collection, paving the way for successful testing, refinement and eventual deployment in real-world aquatic environments.

Chapter

3 Development and Implementation of Autonomous Boat Missions

Contents

3.1 Introduction

The development and validation of the Autonomous Cleaner Boat's Minimum Viable Product (MVP) for waste collection in marinas, rivers, and lakes rely on robust electrical integration, sophisticated simulation, and rigorous testing. This chapter outlines the meticulous design and implementation of the boat's electrical cabling, ensuring reliable power distribution and communication among sensors, actuators, Raspberry Pi, and Arduino, using protocols like UART for low-latency data exchange in dynamic aquatic environments. It also details the creation of a virtual simulation environment, selecting ROS1 Noetic over Webots for its modular framework and extensive support, to model navigation, vision-based debris detection, and trajectory planning. The chapter further describes comprehensive testing, encompassing unit tests for individual components, integration tests for module interactions, and system-level evaluations in simulated and real-world settings, to confirm the MVP's effectiveness in autonomous waste collection under challenging marine conditions.

3.2 Virtual Simulation Development

This section describes the virtual simulation framework created to evaluate the essential capabilities of the autonomous marine waste collection system, such as navigation, obstacle evasion, and debris identification.

3.2.1 System Architecture

The autonomous system is structured into several interconnected components, primarily leveraging the modularity inherent in the Robot Operating System (ROS) architecture. To fully understand the different components of the system, we illustrate it in this figure, which shows a clear representation of the connection and the flow of information between the sections and subsections of this system.

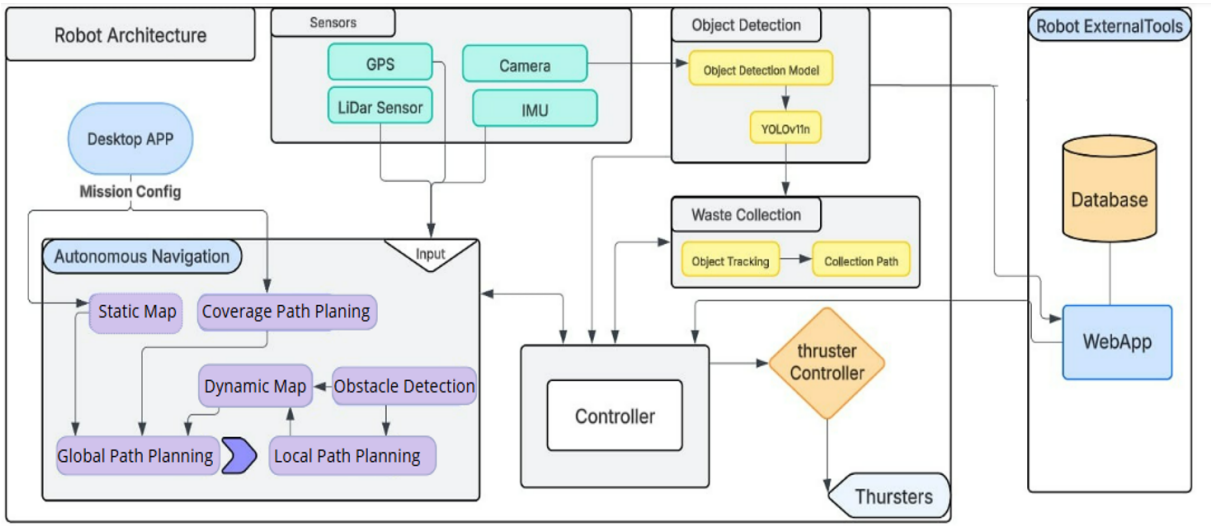

Figure 3.1: Robot System Architecture

Figure 3.1 illustrates that Our autonomous system is divided into multiple sections in the ROS architecture. This will allow us to develop each section independently, allowing us to focus on the development of specific functionalities.

3.2.2 Simulation Environment

The Gazebo simulator was chosen for its powerful physics engine and smooth compatibility with ROS Noetic, making it well-suited for marine robotics projects. A tailored marine setting was developed to mimic coastal and open ocean scenarios, featuring water surfaces, currents, and obstacles such as buoys and debris. The ‘ `asv_wave_sim_gazebo`‘ package, based on the ‘`asv_wave_sim`‘ framework, utilized plugins such as ‘**libHydrodynamicsPlugin.so**‘ to simulate water physics, including buoyancy, damping, and drag forces. Supplementary plugins, like ‘`asv_wave_sim_gazebo_plugins`‘, modeled ocean waves and surface movements to enhance authenticity. RViz was used to visualize the environment, allowing real-time observation of the boat performance and sensor outputs.

3.3 Vessel Design and Simulation Model

This section details the step-by-step process of creating and integrating the boat model into the ROS 1 Noetic simulation environment, including exporting the model from SolidWorks, importing it into the ROS workspace, configuring sensors (camera, LiDAR, and GPS) using URDF plugins, and testing the navigation simulation.

3.3.1 URDF export

To ensure compatibility with ROS and Gazebo, the SolidWorks model was exported using the 'SolidWorks to URDF Exporter' tools, a plugin developed by the ROS community. This plugin generates an URDF (Unified Robot Description Format) file along with the associated mesh files. Before exporting, all parts were properly named, assembled, and assigned coordinate frames in SolidWorks to ensure correct transformations and joint placement in ROS. In our case, the exportation method provided mesh files specifically in the STL format as shown in figure 3.2.

Figure 3.2: Meshes Folder Structure

3.3.2 Sensor Plugins Integration

Key sensors were integrated into the boat model using Gazebo plugins within the URDF description, as part of the Boatcleaning folder added to the workspace. These sensors enable simulation of autonomous capabilities by providing essential data for perception, localization,

and mapping. The configuration was achieved using <gazebo> tags in the URDF file, specifying plugin types, sensor parameters (such as update rate, resolution, and noise), and each sensor's pose relative to the boat's base link.

- **Camera:**The camera was integrated using the `libgazebo_ros_camera.so` plugin, which allows Gazebo to simulate an RGB camera and publish its data to ROS topics for use in perception tasks.

```
<plugin name="camera_controller" filename="libgazebo_ros_camera.so"> //Gazebo ROS camera plugin
  <alwaysOn>true</alwaysOn> //Keeps camera active continuously
  <updateRate>30</updateRate> //Update rate (30 Hz)
  <cameraName>camera</cameraName> //Name of the camera
  <frameName>camera</frameName> //Reference frame for the camera
  <imageTopicName>image_raw</imageTopicName> //ROS topic for raw image data
  <cameraInfoTopicName>camera_info</cameraInfoTopicName> //ROS topic for camera metadata
</plugin>
```

Figure 3.3: Camera Code Plugin

- **Lidar Plugin:** The LIDAR sensor was integrated using the `libgazebo_ros_laser.so` plugin, which enables the simulation of a 2D laser scanner and publishes scan data to a ROS topic for tasks such as obstacle detection and mapping.

```
<plugin name="gazebo_ros_laser" filename="libgazebo_ros_laser.so"> //Gazebo ROS laser plugin
  <topicName>/scan</topicName> //ROS topic for laser scan data
  <frameName>lidar_center</frameName> //Reference frame for the laser sensor
</plugin>
```

Figure 3.4: Lidar Code Plugin

3.3.3 Hydrodynamic Navigation Simulation

Realistic boat bounce in simulation is achieved using the "`asv_wave_sim`" package, which provides hydrostatics plugins to model buoyancy.This package supports ROS Noetic, enabling accurate hydrodynamic responses and advanced navigation testing, including obstacle avoidance and waste collection scenarios.

- **hydrodynamics plugin:** This plugin models the hydrodynamic forces acting on the boat, enabling accurate floating behavior. The configuration included damping, viscous drag,

and pressure drag, three key components that contribute to the realistic motion of a vessel in water.

```
<plugin name="hydrodynamics" filename="libHydrodynamicsPlugin.so"> //Gazebo ROS hydrodynamics plugin
  <wave_model>ocean_waves</wave_model> //Wave model for water surface dynamics
  <damping_on>true</damping_on> //Enables damping forces
  <viscous_drag_on>true</viscous_drag_on> //Enables viscous drag forces
  <pressure_drag_on>true</pressure_drag_on> //Enables pressure drag forces
  <markers> //Visualization markers
    <update_rate>30</update_rate> //Marker update rate (30 Hz)
    <water_patch>false</water_patch> //Disables water surface patch visualization
    <waterline>false</waterline> //Disables waterline visualization
    <underwater_surface>false</underwater_surface> //Disables underwater surface visualization
  </markers>
</plugin>
```

Figure 3.5: hydrodynamics Code Plugin

With the integration of the libHydrodynamicsPlugin.so, the boat model successfully navigated in the Gazebo 11 simulation environment. The hydrodynamics plugin provided realistic water interaction, ensuring stable floating behavior and accurate responses to hydrodynamic forces such as damping. Figure 3.6 below illustrates the boat's successful navigation within the simulated environment.

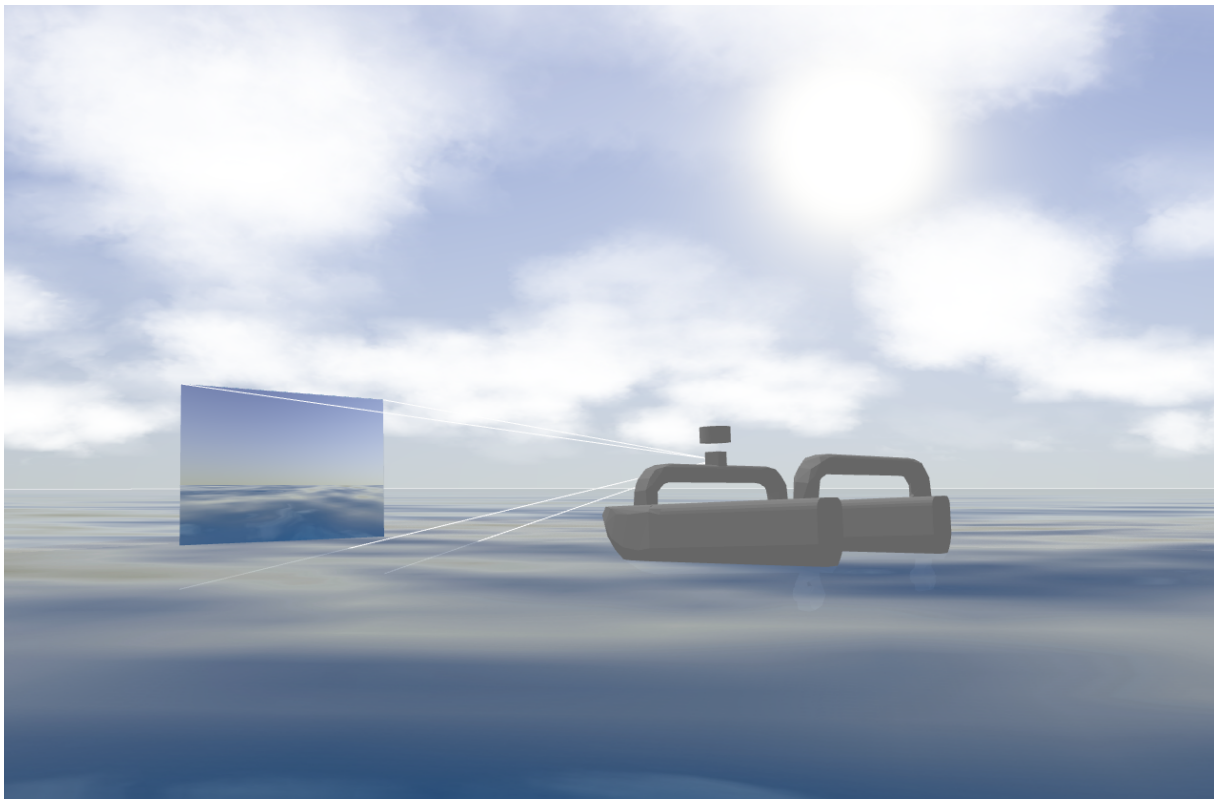

Figure 3.6: hydrodynamics Code Plugin

3.3.4 Autonomous Navigation Simulation

The autonomous navigation system was implemented in the "asv_wave_sim_gazebo" simulation environment, serving as the primary platform for developing and testing boat navigation. Key ROS packages were integrated to achieve autonomy: Slam_Gmapping for real-time SLAM, Move_Base for path planning and obstacle avoidance, and AMCL for precise localization within generated maps using sensor data.

3.3.4.1

The "asv_wave_sim_gazebo" package, which I developed, organizes the required functionalities of system in a clear and structured manner, shown in Figure 3.7 .

Figure 3.7: Overview of the main directories in the ROS package structure

- **config:** This folder holds YAML files for system parameters like navigation and simulation settings. These files enable fine-tuning of the robot's behaviour for various marine environments.
- **launch:** This directory contains launch files that orchestrate the startup of multiple ROS nodes and configurations. It ensures smooth initialization of the system for simulation and deployment.
- **maps:** This directory stores PGM-format virtual maps essential for the navigation stack. These 2D occupancy grids support mapping and localization in static and dynamic marine settings.
- **media:** This folder includes visual assets like water textures to enhance Gazebo simulation realism. These assets improve the visual accuracy of the marine environment for testing.

- **models:** This folder contains 3D models and URDF files of objects like debris for Gazebo simulations. It supports testing of navigation and waste detection functionalities.
- **msg:** This directory defines custom ROS message types for efficient node communication. Messages like Detected Objects enable seamless data exchange for waste detection.
- **scripts:** This folder houses executable nodes for autonomous navigation and waste collection.
- `world_models` **and worlds:** These folders contain simulation maps and models defining ocean environments for Gazebo. They provide realistic scenarios for testing robot performance.

3.3.4.2 Mapping Package Configuration

The configuration of `slam_gmapping` is an essential step in enabling robust simultaneous localization and mapping (SLAM) for ROS-based robotics projects. This package utilizes laser scan data and odometry information to incrementally construct 2D occupancy grid maps while estimating the robot's pose in real time. Configuration is primarily managed through YAML files, typically located in the project's configuration directory. As illustrated in Figure 3.8, these YAML files allow users to clearly set and adjust parameters such as map resolution, update rates, particle filter settings, and sensor topic names in an organized manner.

```yaml
local_costmap:
  global_frame: odom          # Matches your odom plugin's frameName
  robot_base_frame: baseboatclening  # Matches your URDF base link

  update_frequency: 5.0       # Update at 5 Hz
  publish_frequency: 4.0      # Publish at 4 Hz

  static_map: false
  rolling_window: true        # Keep the costmap centered on the boat
  width: 6.0                 # 6 m x 6 m local window
  height: 6.0
  resolution: 0.05           # 5 cm grid resolution
```

Figure 3.8: Costmap YAML Configuration

3.3.5 Map Generation and Visualization

The process of map generation and visualization is fundamental in autonomous robotics, as it enables the robot to understand and navigate its environment effectively. In this project, the configured ‘`slam_gmapping`‘ package was used to collect laser scan and odometry data, allowing the robot to construct an occupancy grid map of its surroundings in real time. Map generation not only provides essential spatial information for navigation but also plays a crucial role in planning and obstacle avoidance.

To ensure accurate localization within the map generated, the AMCL package was integrated. AMCL uses particle filtering to continuously estimate the robot's position on the map using sensor data, providing robust and reliable localization for autonomous navigation.

The static map represents a snapshot of the environment created after the mapping process is complete, as shown in Figure 3.9. This type of map is especially useful in environments where the arrangement of obstacles and features remains unchanged over time, such as offices, warehouses, or laboratories. By relying on a static map, the robot can efficiently plan routes and avoid obstacles without the need for ongoing map updates, which reduces computational demands during navigation. Furthermore, static maps are valuable for defining the boundaries of the robot's operational area. This allows users to specify precise limits for navigation, ensuring the robot focuses only on designated zones, an important consideration for tasks like surface cleaning, where it is essential to restrict the robot's activity to targeted areas.

In contrast, **the dynamic map** is continuously updated as the robot moves and as the environment evolves. This approach is crucial in scenarios where the environment is subject to change, such as public spaces, warehouses with moving objects, or outdoor settings where obstacles may appear or disappear. Dynamic mapping allows the robot to adapt in real time, ensuring that its understanding of the environment remains accurate and up to date. This capability is especially important for applications requiring high levels of autonomy and safety, as it enables the robot to respond to unexpected changes and maintain reliable navigation even in unpredictable conditions.

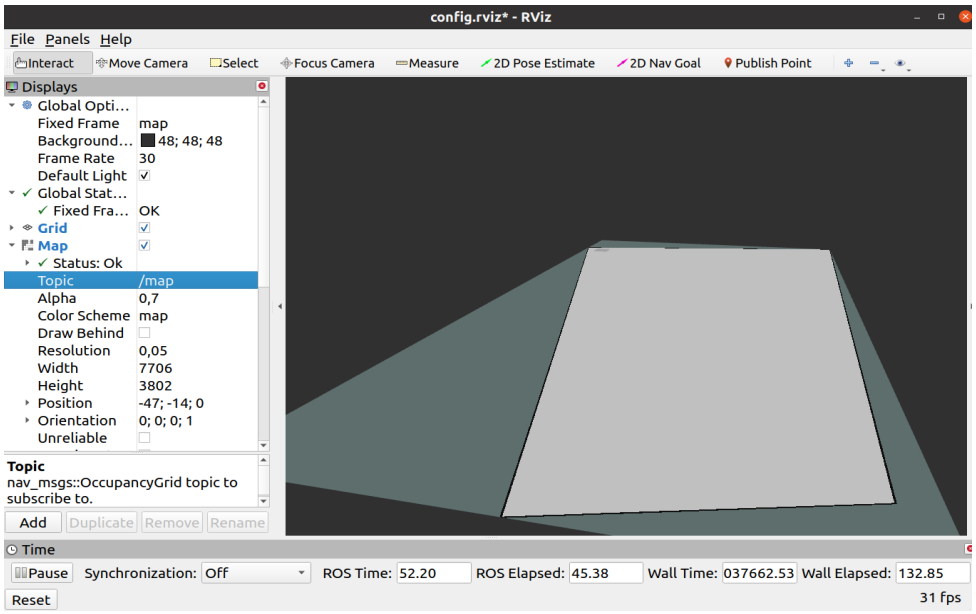

Figure 3.9: RViz Static Map Visualization

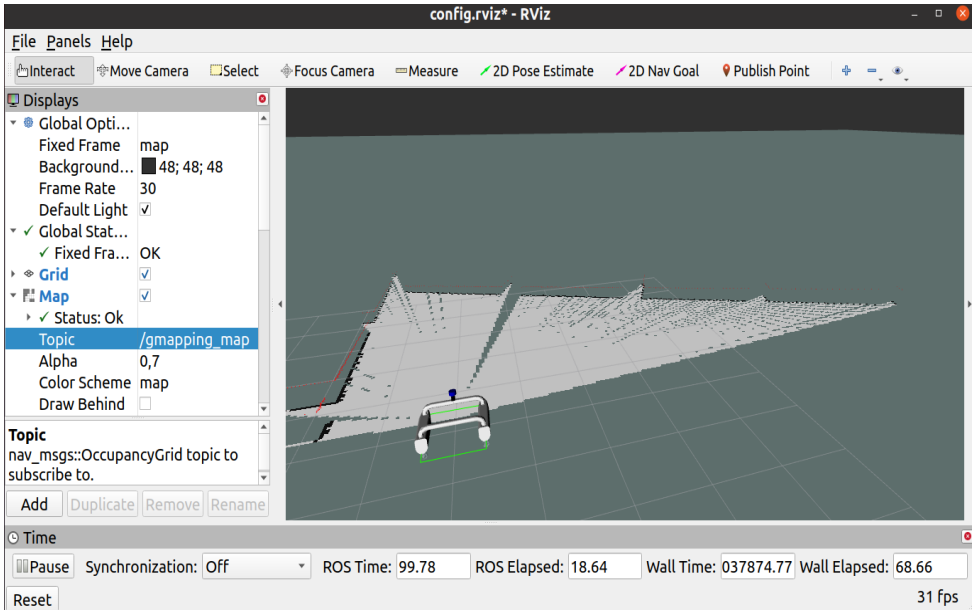

Figure 3.10: RViz Static Map Visualization

3.3.5.1 Obstacle Avoidance

Obstacle avoidance is a crucial capability for any autonomous vehicle, ensuring not only the successful completion of navigation tasks but also the safety of the platform and its environment. In this simulation, obstacle avoidance was achieved through the integration of the `move_base` package, real-time sensor feedback, costmaps, and interactive goal setting using RViz.

The `move_base` node is the core of the ROS navigation stack for autonomous path planning and obstacle avoidance.As shown in Figure 3.12, it operates by integrating two planners and two types of costmaps:

- Global Planner & Global Costmap: The global planner uses the static (global) costmap, built from the known map, to compute an initial path from the robot's current position to the goal while avoiding fixed obstacles.
- Local Planner & Local Costmap: The local planner relies on the local costmap, which is continuously updated with real-time sensor data. This allows the robot to detect and react to dynamic obstacles, such as moving objects or people, by adjusting its trajectory in real time.

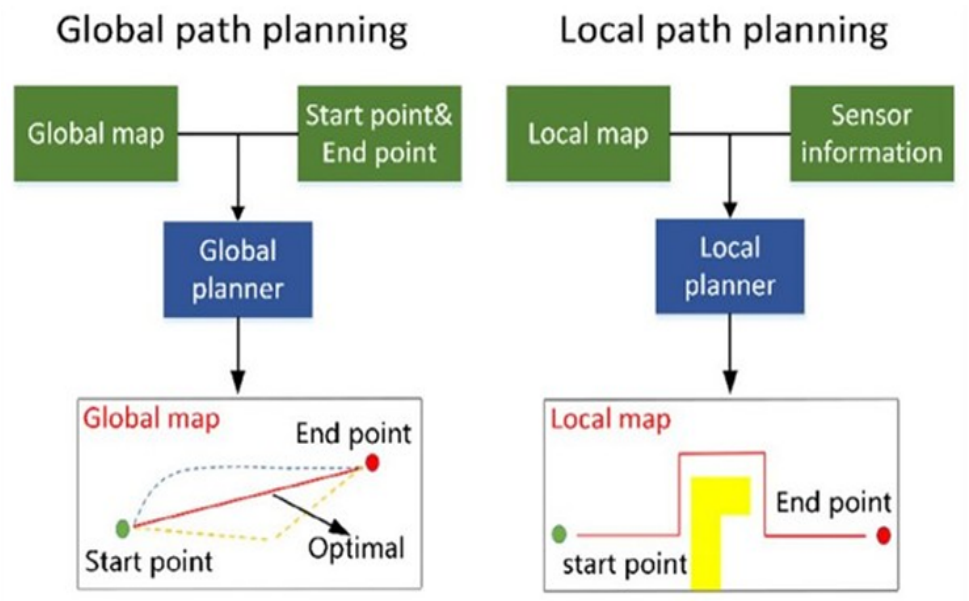

Figure 3.11: Global and Local Path Planning Process

In situations where no valid path is available, such as when the robot is trapped or surrounded by obstacles, the `move_base` node can trigger recovery behaviors. These include actions like rotating in place to search for an escape route or clearing the local costmap to remove outdated obstacle data. Such strategies help the robot resolve navigation deadlocks and continue toward its goal, increasing the overall robustness and reliability of autonomous navigation.

3.3.6 ROS Node Architecture and Messaging

The autonomous marine robot's ROS system, which I developed, leverages a network of interconnected nodes to achieve modular functionality. This design organizes the system into independent yet cooperative components, each handling specific tasks such as sensor processing or actuator control, seamlessly integrated within my custom boatcleaning package. As shown in Figure below, the primary nodes developed for this system are described in the following sections.

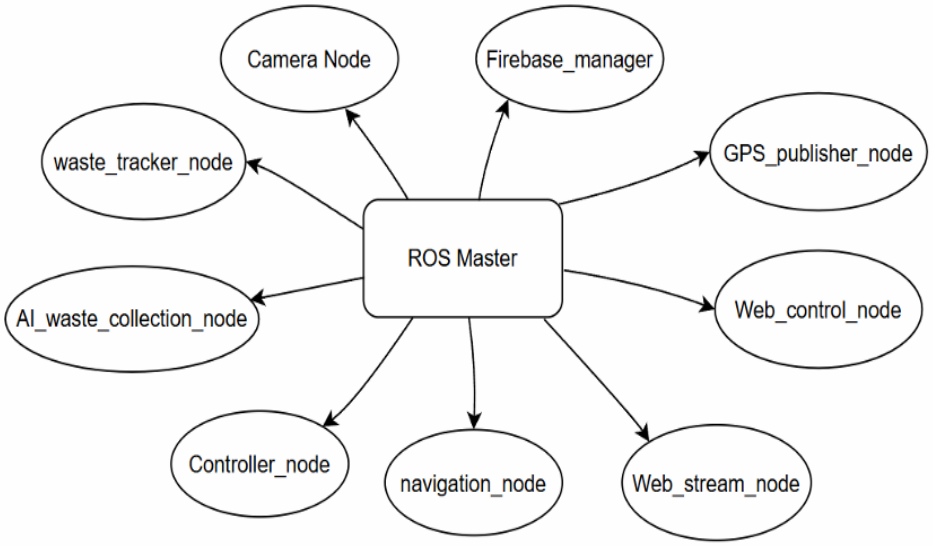

Figure 3.12: ROS Node Architecture

- `Camera_node`**:** Captures real-time video streams from the robot's onboard camera and publishes images to '/camera/`image_raw`'. It supplies image data to other nodes, such as the waste detection and tracking modules.

- `Firebase_manager`**:** This node sets up and manages the connection to the Firebase database. It handles uploading waypoints, downloading mission plans, and keeping the data synchronized between the robot and the cloud.

- `GPS_publisher_node`**:** This node listens to the robot's GPS data and sends those coordinates to Firebase for real-time tracking. Like the Firebase Manager, it communicates directly with the cloud, not through ROS topics.

- `Web_control_node`: Interfaces with the web-based control panel, enabling remote commands and real-time status updates between the robot and the user.
- `Web_stream_node`: Subscribes to '/camera/`image_raw`' and streams video and telemetry data to the web interface, allowing users to monitor the robot's progress and environment remotely.
- `navigation_node`: This node publishes updates about navigation to '/`navigation_status`' and listens for commands on '/`navigation_control`'. It also uses data from Firebase and the GPS publisher to follow the planned path.
- `waste_tracker_node`: This node tracks waste objects over time to help ensure a successful pickup. It listens to the camera stream via '/camera/`image_raw`' and publishes detection results and events on '/`waste_info`', '/`waste_detected`', and '/`waste_detection`'.
- `AI_waste_collection_node`: This node handles the actual collection of trash. When waste is detected, it processes the object's position and controls the robot to pick it up.
- `Controller_node`: Acts as the decision-making unit, managing state transitions (e.g., switching between navigation and waste collection modes). It subscribes to '/`waste_detected`', '/`collection_status`', and '/`navigation_status`'. It publishes to '/`navigation_control`', '/`collection_control`', and '/webcontroler'.

3.3.7 AI-Based Waste Detection and Collection Node Integration

In this project, I implemented the integration of an existing AI-based object detection system with the autonomous cleaning platform using the `Ai_collection_node`. My role focused on connecting the outputs of the pre-trained AI model to the robot's control logic. The `Ai_collection_node` receives detection results and translates them into navigation and collection commands, enabling real-time, autonomous waste tracking and collection.

3.4 Autonomous cleaning program

The autonomous cleaning program is the core of the robot's operation, seamlessly integrating mapping, navigation, detection, and waste collection into a single workflow. Each stage of the process is detailed below, with visual illustrations to support each step.

3.4.1 Defining the Cleaning Zone

To begin, I developed an HTML page displaying an interactive world map (by default, centered on Tunisia). On this interface, I selected the specific area that the robot should clean. The selected region then serves as the static boundary for the robot's autonomous navigation.

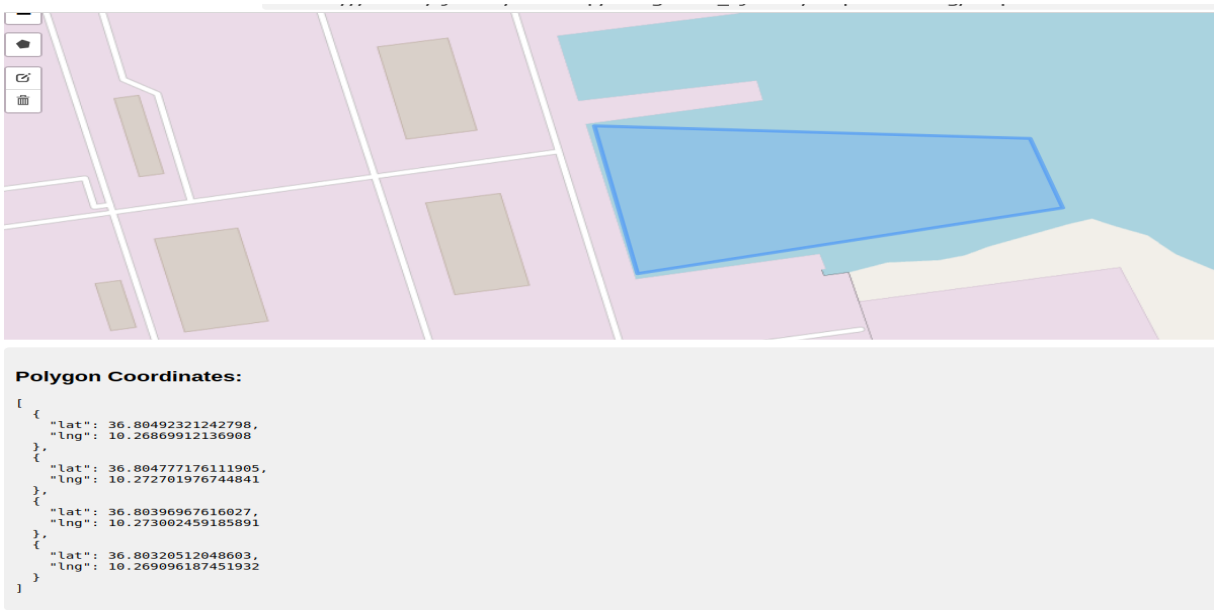

Figure 3.13: Selection of Cleaning Area on Interactive Map

The application generates the latitude and longitude coordinates of the polygon marking this area.

3.4.2 Planning the Navigation Path

Next, I imported these coordinates into a desktop application that converts the (lat, long) points to local (x, y) coordinates. In this app, I chose the robot's starting point for the cleaning mission.

The application supports both manual and automatic waypoint generation, ensuring the robot systematically covers the entire cleaning zone.

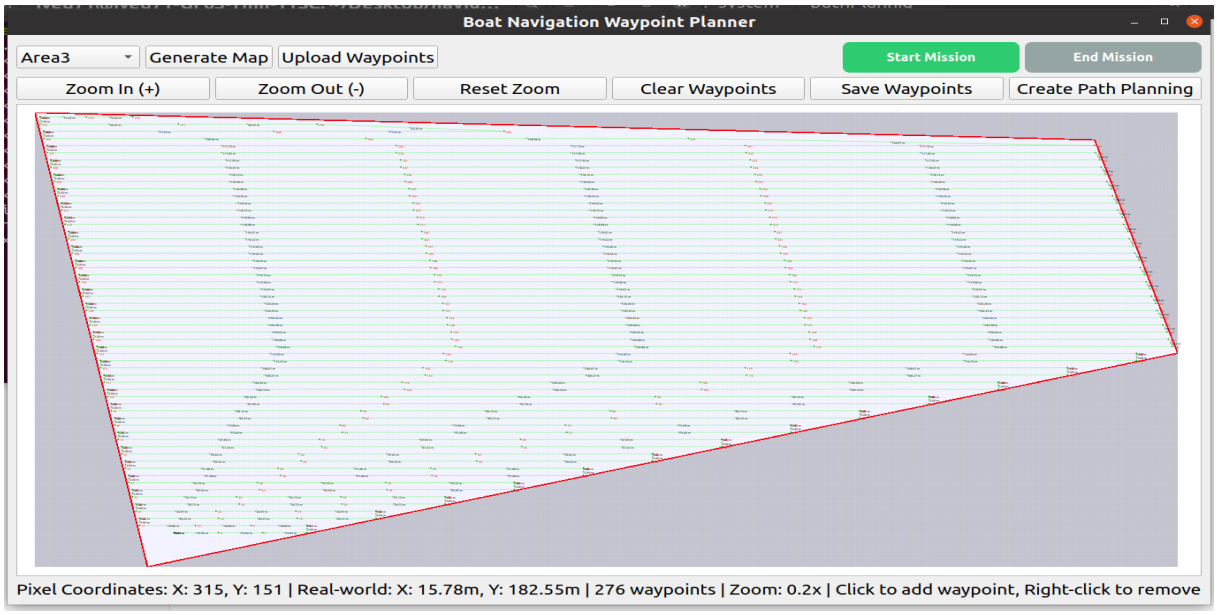

Figure 3.14: Waypoint Generation and Path Planning

3.4.3 Uploading and Synchronizing the Map

Once the navigation map and waypoints are prepared, I uploaded them to Firebase. This ensures synchronization with the main project workspace. The autonomous navigation node then retrieves these waypoints from Firebase and guides the boat through each one in sequence.

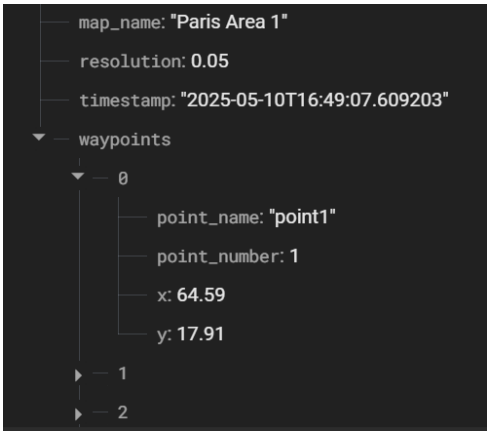

Figure 3.15: Waypoint Generation and Path Planning

The waypoint and map data are structured and stored in Firebase for synchronization with the robot's navigation system.

3.4.4 Autonomous Navigation and Waste Detection

As the boat follows its planned route, the onboard navigation node continuously manages movement commands. If the waste detection camera node spots a floating object, the controller node pauses navigation to prioritize collection.

3.4.5 Waste Tracking and Collection

When waste is detected, control shifts to the waste collection node. This node tracks the detected object, plotting a path to align the waste with the centre of the boat's collection area for reliable retrieval. The robot actively adjusts its heading during this phase to ensure successful collection.

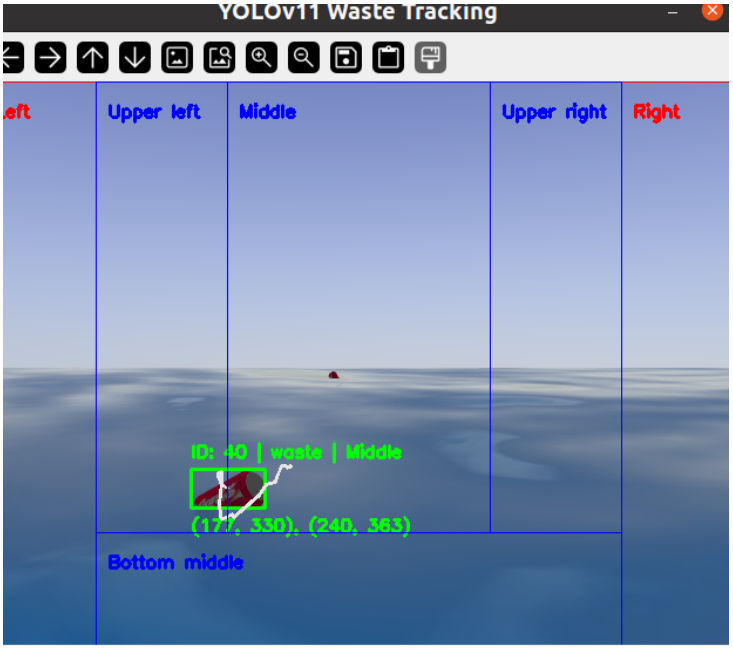

Figure 3.16: Waste Detection and Tracking Interface

3.5 Hardware Integration and Testing

3.5.1 Introduction

The success of any autonomous system not only depends on the quality of its hardware and software but also on the precision and reliability of its electrical integration. In the Ocean Cleaner project, electronic cabling forms the backbone of communication and power distribution across all critical components, from sensors and actuators to processing units and power systems. In addition, rigorous testing of both the hardware and software was conducted to validate the system's performance and reliability. This includes unit testing for individual components, integration testing to evaluate how different modules interact, and system-level testing in simulated and real-world environments to ensure the Ocean Cleaner performs its mission of autonomous waste collection effectively and safely.

3.5.2 Raspberry Pi Environment Setup

This section outlines the essential steps to prepare the Raspberry Pi for robotics development. It covers downloading and installing Ubuntu 20.04, performing initial setup, and installing the ROS 1 Noetic framework.

3.5.2.1 OS Installation

The Raspberry Pi Imager software was utilized to configure the system's SD card efficiently. This tool streamlines the process of selecting a compatible operating system and ensures a reliable installation on the device. For this project, Ubuntu Server 20.04.5 LTS (64-bit) was chosen due to its stability and robust support for robotics applications. The user-friendly interface of Raspberry Pi Imager allows for quick selection and writing of the OS image, minimizing setup time and reducing the likelihood of errors [12].

Figure 3.17: Raspberry Pi Imager - Operating System and Storage Selection

3.5.2.2 System Initialization

Following the initial boot sequence of the Raspberry Pi, the system was prepared for development by ensuring that all software packages were current. This was accomplished by executing the following command to update the package index and upgrade all existing packages:

```
sudo apt update && sudo apt upgrade -y
```

To establish a comprehensive graphical user interface conducive to development and general usage, the complete Ubuntu desktop environment was installed using the command below:

```
sudo apt install ubuntu-desktop -y
```

Once the installation was complete, the Raspberry Pi was rebooted to apply all changes and activate the newly installed desktop interface. This initialization step ensured a fully functional and up-to-date software environment, providing a stable foundation for subsequent configuration and deployment of the robotic system.

3.5.2.3 Remote Control and Monitoring Setup

Efficient remote control and monitoring of the Raspberry Pi during the development and testing phases was achieved using **RealVNC Viewer**, selected as the remote desktop solution for this project. This tool offers secure and reliable access to the Pi's graphical interface over a network, enabling full system interaction without the need for a physically connected monitor, keyboard, or mouse.

The VNC Server feature was activated using the Raspberry Pi configuration tool by executing the following command:

```
sudo raspi-config
```

Then, under `Interface Options > VNC`, the VNC server was enabled.

Once enabled, **RealVNC Viewer** was launched on a host computer, and a connection was established by entering the Raspberry Pi's IP address.

3.5.2.4 ROS Noetic Installation

With the system environment fully prepared, the next step involved installing ROS Noetic Ninjemys, the final long-term support (LTS) version of ROS 1, compatible with Ubuntu 20.04. ROS Noetic was chosen for its stability, extensive community support, and compatibility with the computational resources available on the Raspberry Pi. The installation process followed the official guidelines provided by the ROS community [2], ensuring a reliable and standardized setup. The installation process began by configuring the system to use the official ROS software repository, ensuring access to the most recent and stable ROS packages. Next, the full desktop version of ROS Noetic was installed to provide access to a complete development and simulation environment:

```
sudo apt install ros-noetic-desktop-full -y
```

After installation, the ROS environment was initialized and configured by updating the system's shell profile to source the ROS setup script. Essential development tools and

dependencies were then installed and initialized using the 'rosdep' utility to ensure proper package management. The successful installation of ROS Noetic was verified by running the 'roscore' command. As shown in Figure below, the ROS master and core services started without errors, confirming a functional ROS environment.

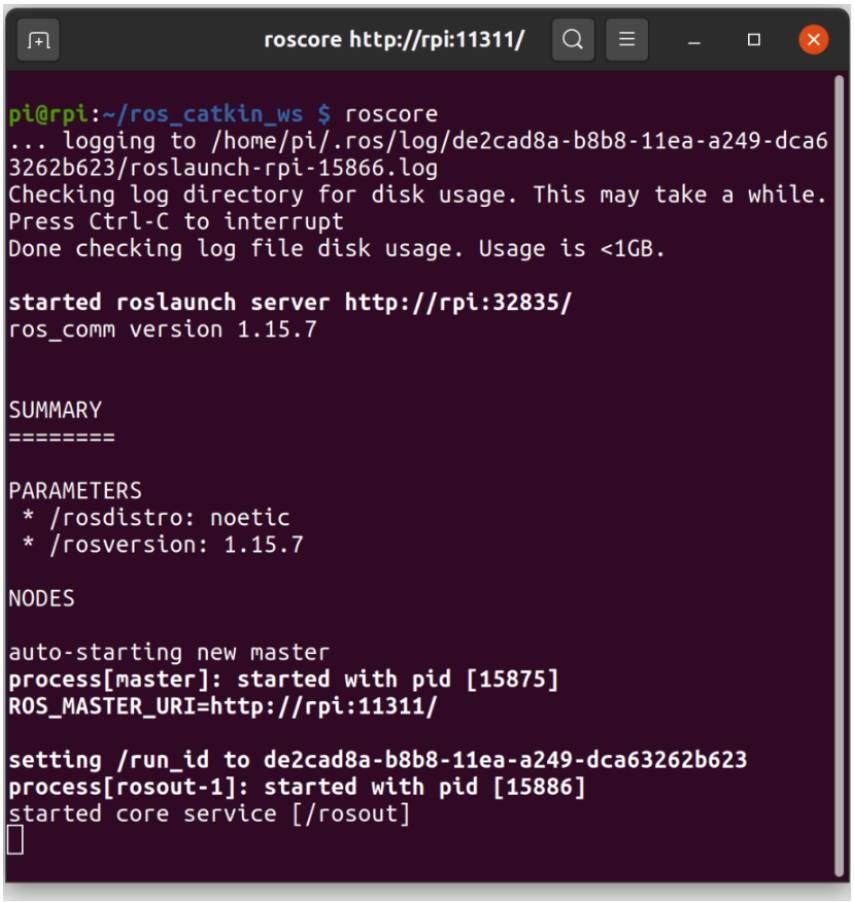

Figure 3.18: ROS Noetic core successfully running on Raspberry Pi

3.5.3 Hardware Component Integration

With the Raspberry Pi and ROS environment fully prepared, the next phase centered on the physical integration and configuration of essential hardware components. This section provides an overview of the cabling, initial setup, and verification steps performed for each device.

3.5.4 LIDAR Sensor Connection and Configuration

The RPLIDAR A1 was integrated into the system to provide real-time laser scanning capabilities. Emphasis was placed on the hardware connection and proper system-level permissions to ensure reliable communication before proceeding with software integration.

3.5.5 Hardware Connection and USB Permissions

The RPLIDAR sensor was physically connected to the Raspberry Pi using a *USB-to-microUSB* cable.

To verify that the LIDAR device was recognized by the operating system and to set the correct permissions, the following commands were executed:

```
ls -la /dev | grep ttyUSB
sudo chmod 666 /dev/ttyUSB0
```

This process ensured that the ROS nodes would have the necessary access to the serial port for real-time data acquisition.

3.5.5.1 ROS Driver Installation and Visualization

Once the hardware connection was confirmed, the ‘`rplidar_ros`‘ package was installed on ROS Noetic. The serial port was configured according to the detected device (typically ‘/dev/ttyUSB0‘), and the LIDAR node was launched. Real-time laser scan data was successfully visualized in RViz, confirming the successful integration of hardware and software (see Figure 3.19).

This approach ensured robust hardware integration as a foundation for further development and mapping tasks.

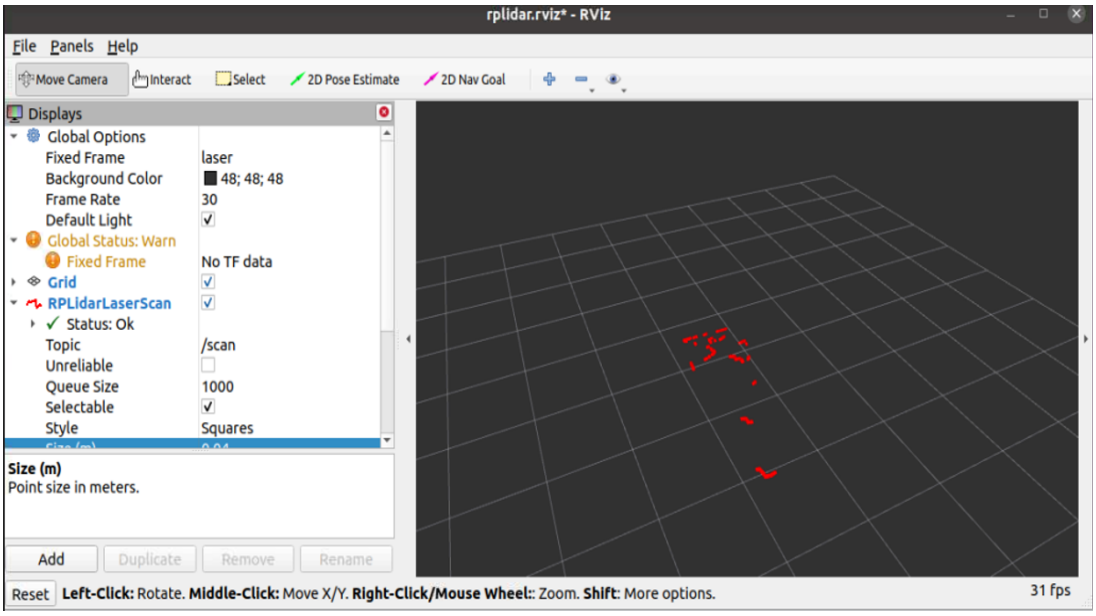

Figure 3.19: RViz LIDAR scan visualization

3.5.6 IMU Module Connection and Configuration

- **IMU Hardware Configuration and Coupling**

  Configuration of the 9-axis IMU communication on the Raspberry Pi 4 is accomplished by enabling the I2C interface. The I2C option is activated via the Raspberry Pi configuration tool, allowing the system to recognize and interact with I2C peripherals. The IMU's SDA and SCL lines are connected to pins 3 and 5 on the Raspberry Pi 4 GPIO header, with power and ground properly linked. Wiring adheres to the I2C protocol, ensuring reliable communication for sensor data acquisition (see Figure 3.20).

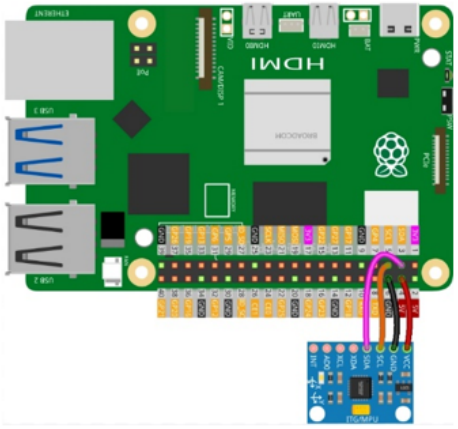

Figure 3.20: IMU–Raspberry Pi Wiring

- **ROS Driver Installation and Data Visualization** A custom ROS node was developed to interface with the 9-axis IMU via the I2C port on the Raspberry Pi 4. This node continuously reads raw sensor data from the IMU and publishes it as standardized ROS messages, making orientation and acceleration information available to the entire ROS ecosystem. IMU data is visualized in RViz for real-time monitoring, as illustrated in Figure 3.21.

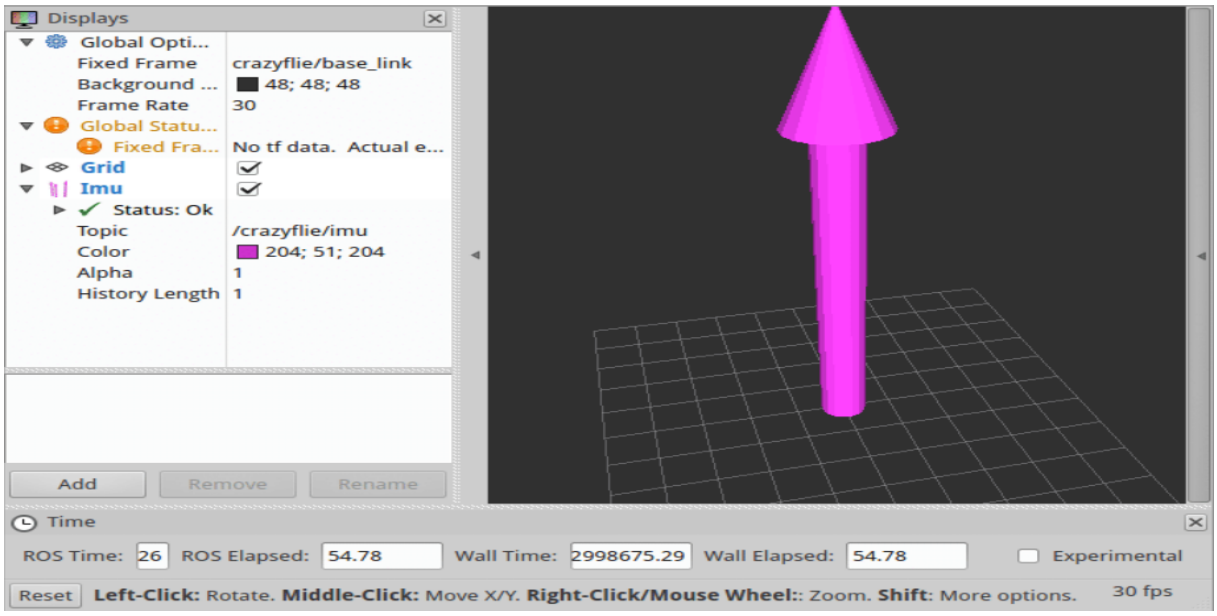

Figure 3.21: IMU data visualization in RViz.

3.5.7 GPS Sensor Connection and Configuration

Integration of the SIM808 GPS module was essential for real-time geolocation and remote monitoring of the Ocean Cleaner. This section details the configuration of UART communication, the hardware wiring, and data validation.

3.5.7.1 UART Configuration on Raspberry Pi

UART communication was enabled by activating the serial interface and disabling the serial login shell using raspi-config. This configuration step was necessary to avoid interface conflicts and ensure seamless data exchange with the Arduino Uno intermediary.

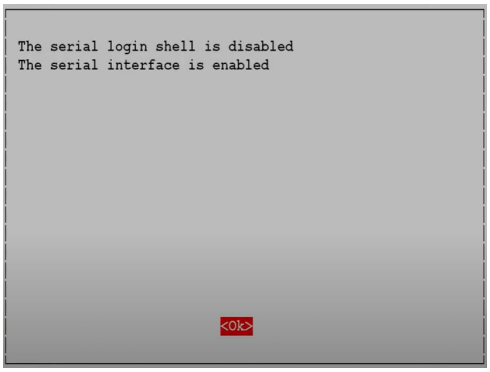

Figure 3.22: UART configuration in raspi-config.

3.5.7.2 Hardware Wiring

Due to the lack of a reliable SIM808 GPS library in the Raspberry Pi community, the Arduino Uno was used to interface with the SIM808 module via UART (Arduino pins 7 and 8). The Arduino was then connected to the Raspberry Pi 4 through UART (Arduino pins 0 and 1 to Pi 4 pins 8 and 10), with all devices sharing a common ground. Power was distributed from a single power bank, which supplied both the SIM808 module and the Raspberry Pi 4; the Raspberry Pi, in turn, provided 5V power to the Arduino Uno.

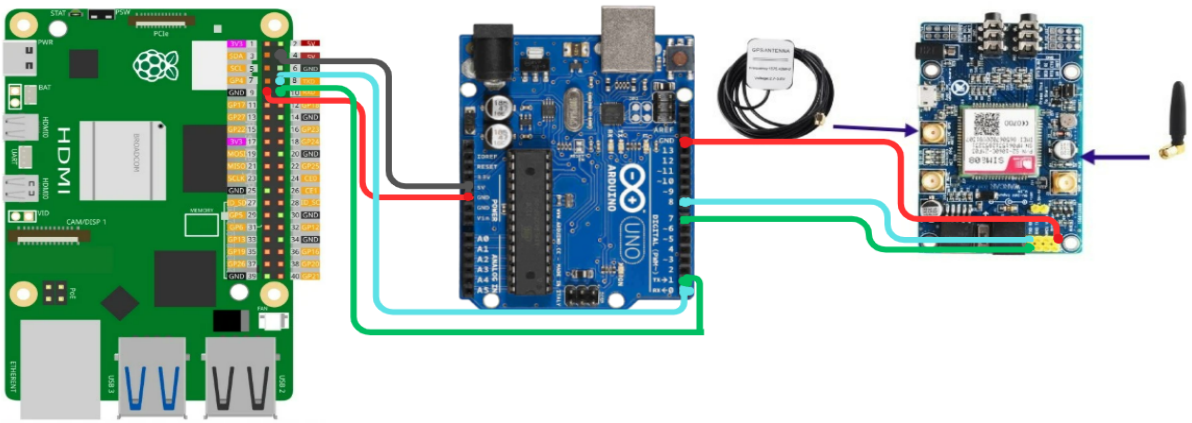

Figure 3.23: Wiring diagram for UART communication.

3.5.7.3 Data Acquisition and Validation

A Python script utilizing the pyserial library was developed to verify UART communication on the Raspberry Pi, targeting the /dev/ttyAMA0 port. The script successfully received GPS coordinates from the Arduino Uno, confirming reliable data transmission and integration.

```
pi@raspberrypi:~$ python3 testuart.py
Listening to Arduino UART on /dev/ttyAMA0...
Received: GPS: 36.849136, 10.274944
Received: GPS: 36.849132, 10.274959
Received: GPS: 36.849121, 10.274966
Received: GPS: 36.849105, 10.274969
Received: GPS: 36.849098, 10.274972
Received: GPS: 36.849098, 10.274969
Received: GPS: 36.849105, 10.274966
Received: GPS: 36.849113, 10.274964
Received: GPS: 36.849109, 10.274962
Received: GPS: 36.849113, 10.274960
Received: GPS: 36.849113, 10.274959
Received: GPS: 36.849113, 10.274957
Received: GPS: 36.849113, 10.274953
Received: GPS: 36.849109, 10.274950
Received: GPS: 36.849109, 10.274952
```

Figure 3.24: GPS data on the Raspberry Pi terminal.

3.5.8 Camera Connection and Configuration

The integration of a camera module was essential for providing vision-based perception to the Ocean Cleaner system. This section outlines the hardware connection, configuration, and testing of the camera on Raspberry Pi 4.

3.5.8.1 Hardware Connection

The camera module was physically connected to the Raspberry Pi 4 using the dedicated CSI (Camera Serial Interface) port. Care was taken to ensure correct orientation of the ribbon cable and a secure connection to prevent signal loss. Figure 3.25 illustrates the proper connection of the camera module to Raspberry Pi 4.

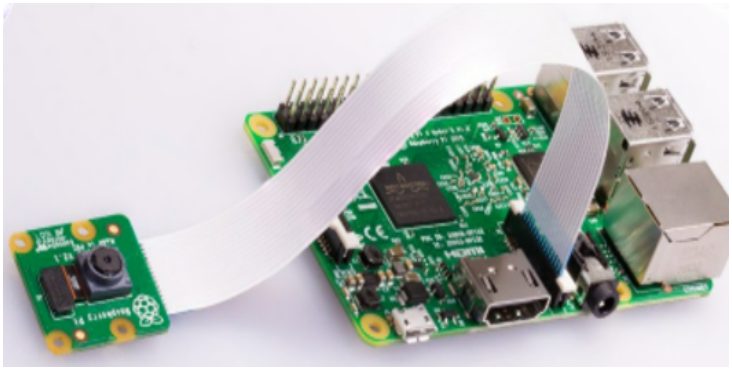

Figure 3.25: Camera module connected to the Raspberry Pi 4 via CSI port.

3.5.8.2 Camera Module Configuration on Ubuntu 20

During the setup on Ubuntu 20, the default `libcamera` suite was found to be incompatible with the operating system. To resolve this issue, alternative packages `ffmpeg` and `v4l-utils` were installed using the following command:

```
sudo apt install ffmpeg v4l-utils
```

After installation, camera functionality was successfully verified by streaming live video from the camera module using:

```
ffplay /dev/video0
```

This command opened a new window displaying the real-time camera feed, confirming successful integration of the camera module.

3.5.9 Motor Control System Integration

The motor control system for the Ocean Cleaner is based on a modular approach that separates high-level commands and low-level actuation.

3.5.9.1 Motor Control System

The Raspberry Pi 4 is responsible for overall system control and transmits motor control data to the Arduino Uno via UART. The Arduino Uno then interprets these signals and generates PWM to the Electronic Speed Controllers (ESCs) of each brushless three-phase motor. This architecture enables the Raspberry Pi to focus on navigation and decision-making, while the Arduino handles real-time actuation, ensuring responsive and robust differential steering.

3.5.9.2 Hardware Wiring and Power Distribution

The ESCs are directly powered by separate LiPo batteries (Zeee 4S 14.8V 5200mAh) to meet the high current requirements of the APISQUEEN U2 MINI thrusters. The Arduino Uno

receives its power from the 5V output of the Raspberry Pi 4. All modules share a common ground to maintain electrical stability. The UART interface connects Raspberry Pi pins 8 (TX) and 10 (RX) to Arduino pins 0 (RX) and 1 (TX), respectively, facilitating reliable serial communication. PWM signals for motor control are output from Arduino pins 9 and 10 to the signal wires of each ESC.

The complete wiring configuration, integrating the Raspberry Pi 4, Arduino Uno, two ESCs, two brushless motors, and dual LiPo batteries, is illustrated in Figure 3.26.

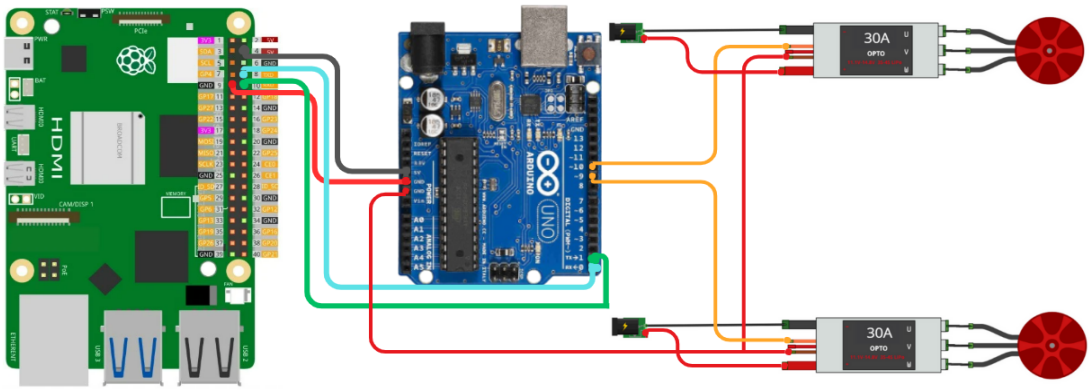

Figure 3.26: Motor Control System Wiring Diagram.

3.5.10 Power System Integration

The power system for the Ocean Cleaner was wired to ensure reliable operation, with all components except the motor cables housed in a protective box to prevent water exposure. The Raspberry Pi 4 and SIM808 module were powered via a TECTIN 20000mAh 66W power bank, connected through a USB to Type-C cable for the Pi 4 and a USB to DC cable for the SIM808, with cables secured inside the box. The two APISQUEEN U2 MINI thrusters, each paired with an ESC, were powered by separate Zeee 4S 14.8V 5200mAh LiPo batteries, with positive and negative terminals wired to the ESCs, while only the motor cables extended outside, designed to withstand water contact. This setup ensures stable power distribution, minimizes overheating risks, and supports extended mission durations for autonomous waste collection.

3.6 Conclusion

This chapter successfully developed and validated the Autonomous Cleaner Boat's Minimum Viable Product (MVP) for waste collection in marinas, rivers, and lakes through robust electrical integration and comprehensive virtual simulation. The power system, utilizing a TECTIN 20000mAh power bank for the Raspberry Pi 4 and SIM808 module, and Zeee 4S 14.8V 5200mAh LiPo batteries for APISQUEEN U2 MINI thrusters, was meticulously wired with protective housing to ensure reliable operation and water resistance, supporting extended mission durations. Concurrently, a virtual simulation environment built on ROS1 Noetic Ninjemys, leveraging packages like `asv_wave_sim`, `move_base`, amcl, and `slam_gmapping`, provided a realistic platform to test navigation, vision-based debris detection, and trajectory planning under simulated maritime conditions. The modular ROS architecture and Firebase integration enhanced data flow and mission adaptability. Through unit, integration, and system-level testing in both virtual and real-world settings, this work confirmed the MVP's reliability and performance, establishing a solid foundation for autonomous waste collection and paving the way for optimized real-world deployment in marine environments.

GENERAL CONCLUSION

This Final Year Project (PFE) successfully developed and validated a Minimum Viable Product (MVP) for an Autonomous Cleaner Boat, designed to address floating debris pollution in marinas, rivers, and lakes, in collaboration with Graines d'Entrepreneurs. Chapter 1 established the environmental urgency of waste pollution and introduced Graines d'Entrepreneurs' role in fostering youth innovation, while comparing solutions like Seabin and WasteShark to justify the need for an autonomous system. Chapter 2 detailed the MVP's design, integrating a catamaran with Raspberry Pi 4, sensors (LiDAR, GPS, cameras), ROS Noetic software, and ABS hulls for stability and efficiency. Chapter 3 demonstrated the system's development through a ROS1 Noetic-based simulation environment, robust electrical integration with UART communication and LiPo battery power, and rigorous testing across unit, integration, and system levels in virtual and real-world settings. The MVP proved its feasibility, achieving reliable navigation, vision-based debris detection, and waste collection under dynamic aquatic conditions. Challenges, such as managing variable water currents and ensuring long-term durability, were identified, offering valuable lessons for refinement. This project contributes to sustainable marine waste management and Tunisia's entrepreneurial ecosystem, showcasing the potential of autonomous surface vehicles. Future iterations could enhance debris detection accuracy, integrate multi-boat coordination, and expand deployment across diverse waterways, paving the way for scalable environmental solutions.

BIBLIOGRAPHY

[1] H. Durrant-Whyte and T. Bailey, “Simultaneous Localization and Mapping: Part I,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[2] Seabin Project, “How It Works,” 2024. [Online]. Available: https://seabinproject.com/how-it-works/

[3] RanMarine Technology, “WasteShark: Autonomous Surface Vessel for Water Debris Collection,” 2024. [Online]. Available: https://www.ranmarine.io/wasteshark/

[Online]. Available: https://doi.org/10.1109/MRA.2006.1638022

[4] Iterative waterfall methodology. *software-engineering-iterative-waterfall-model/. May 2024.* https://www.geeksforgeeks.org/

[5] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA Workshop on Open Source Software*, 2009. [Online]. Available: https://www.ros.org

[6] M. A. Hafez, J. G. Manley, and M. R. Benjamin, “ASV Wave Simulation: A Gazebo-based Package for Testing Autonomous Surface Vehicles in Ocean Conditions,” *GitHub Repository*, 2019. [Online]. Available: https://github.com/osrf/asv_wave_sim

[7] J. Carlton, *Marine Propellers and Propulsion*, 4th ed., Butterworth-Heinemann, 2018.

[8] M. T. Tarafder *et al.*, “Comparative Analysis of Stability Characteristics of Catamaran and Monohull Vessels,” *International Journal of Naval Architecture and Ocean Engineering*, vol. 8, no. 6, pp. 470–479, 2016. [Online]. Available: https://doi.org/10.1016/j.ijnaoe.2016.06.004

[9] "ABS for Marine Applications," *Professional BoatBuilder Magazine*, no. 132, 2011.

[10] International Organization for Standardization, *ISO 12217-1:2015 – Small Craft, Stability and Buoyancy Assessment and Categorization – Part 1: Non-sailing Boats of Hull Length Greater Than or Equal to 6 m.*

[11] A. F. Molland, S. R. Turnock, and D. A. Hudson, *Ship Resistance and Propulsion*, Cambridge University Press, 2011.

[12] Raspberry Pi Foundation, "Raspberry Pi Imager," 2024. [Online]. Available: https://www.raspberrypi.com/software/