MANCHESTER 1824
The University of Manchester

Final Design Review

A coursework submitted to The University of Manchester for the degree of
Master of Science in Robotics
in the Faculty of Science and Engineering

Year of submission
2025

Author
Yuliang Li (11511336) Zihan Yue (10851707)
Jiadong Hou (11517518) Zixiang He (11542275)

School of Engineering

Contents

1 Introduction

1.1 Problem Statement and Objectives

The primary objective of this project is to design and develop an autonomous robotic system capable of identifying, navigating to, and retrieving specific target objects in a dynamic and unmapped environment. The robot must perform these tasks efficiently and reliably, adhering to safety and operational standards while maintaining user-friendly functionality.

The proposed solution involves a mobile manipulator system built on the **Leo Rover** platform, integrated with the **PincherX 150** robotic arm. The system will be equipped with advanced sensors, including the **RPLIDAR A2M12** for 2D mapping and an **Intel RealSense Depth Camera** for 3D perception, enabling accurate navigation and object recognition. An **Intel NUC** serves as the processing unit, providing computational power for onboard algorithms.

Key objectives include:

- Designing robust navigation algorithms to ensure obstacle avoidance and path optimization.
- Implementing efficient object detection and recognition models capable of distinguishing target objects from the background.
- Developing a manipulation strategy for precise object grasping and handling.
- Optimizing the system's power consumption and ensuring sufficient battery life for extended operations.
- Ensuring data security and system reliability through rigorous testing and implementation of cybersecurity protocols.

The project's ultimate goal is to create an autonomous robot that demonstrates high levels of performance, minimal human intervention, and strict compliance with ethical and safety standards. This document reviews the initial design concepts, strategies for component integration, and the overall feasibility of meeting the outlined objectives.

1.2 Feedback Addressed

- **System description optimization:** The system block diagram is placed at the beginning to provide overall context, the subtitle structure is simplified into a list format for clarity, and it is clarified that the depth camera is solely used for object identification while obstacle avoidance relies on RPLiDAR.
- **Enhanced Block Diagram:** The system block diagram is now placed at the beginning of the section to provide a clear and immediate overview of the system architecture. The diagram has been updated to include data flow details, specifying the types of data exchanged between components and the corresponding ROS2 topics used for communication.

- **Optimization of additional power supply issue:** We purchased a 10m extension cable capable of supporting 120W power output, ensuring compatibility with the Intel NUC. Through experiments, we have successfully verified its feasibility and reliability in practical use.
- **Power Consumption Calculation Optimization:** To improve the accuracy of the power calculation, we have added detailed power estimations for additional sensors and considered battery aging effects that can affect the available capacity over time. The updated estimation ensures a more reliable assessment of the battery's ability to sustain the project's operation.
- **RQT Graph Issues:** Provide a more detailed explanation of the function and principles of each node. For large RQT graphs, include a complete image download link to facilitate a comprehensive understanding of the system architecture.
- **Added Grasping Logic Description:** Based on the RQT diagram, a detailed description of the grasping logic and the data flow involved in the robotic arm's operation has been added.
- **Verification Matrix Description Issues:** Accurately describe verification methods, including detailed experimental procedures and result data. Additionally, provide an analysis of the results and propose potential improvements based on the findings.
- **Manufacturing Process Description Issues:** This document provides a detailed description of production and manufacturing processes. Through research and physical comparisons, we analyzed and compared two mainstream manufacturing techniques: laser cutting and 3D printing. We evaluated these methods in terms of production efficiency, material cost, and durability. Our findings indicate that while acrylic, the commonly used material for laser cutting, produces clean cuts and smooth surfaces, it is relatively brittle and prone to breaking. Additionally, unexpected fractures during the cutting process can impact project progress. In contrast, filament, the material used in 3D printing, better meets our requirements. Therefore, we ultimately decided to use 3D printing as our manufacturing method, with filament as the selected material.
- **Exploration of Depth Camera Placement:** This text describes the final placement of the Intel RealSense D435. Through comparative experiments, we conducted on-site exploration tests by changing the position of the Intel RealSense D435 and evaluated its stability in different locations. After considering factors such as camera field of view, exploration speed, and efficiency, we ultimately decided to install it at the upper end of the robotic arm. Additionally, we found that the two pre-drilled screw holes at the front of the robotic arm provided a convenient mounting position for the camera. Therefore, the Intel RealSense D435 was ultimately placed in the two front screw holes of the robotic arm.
- **3D Model Annotation Issues:** In this document, following the feedback suggestions, serial numbers were added to key components in the three-view diagram of the 3D model, along with corresponding names. This annotation arrangement allows readers to more easily understand the overall composition of the robot, making its structure clearer and more comprehensible.

1.3 Modifications

1.3.1 Optimization of Execution Process

To address the potential delay caused by the storage of the object blocks, our team carefully analyzed the existing workflow and identified opportunities for improvement. After evaluating various approaches, we decided to optimize the execution process by dividing it into two distinct phases: the exploration phase and the return phase.

- **Exploration phase:** The robot first uses a color recognition algorithm to scan the environment and identify the target object based on a specific color. Upon detecting the target color, the robot calculates the direction and adjusts its posture to approach the object. Once near the target, the robot further refines its position and orientation to align with the object and uses image recognition to confirm whether it is the target block. If confirmed, the robot executes the grasping operation.
- **Return phase:** The robot transports the block to the designated storage location, identifies the correct storage box, and adjusts its posture for alignment. The robot then places the block securely into the box, verifies successful placement, and returns to its starting position or proceeds to the next task.

1.3.2 Robotic Arm Overload Issue

We identified an issue where the initial pose of the robotic arm was influenced by the 3D support structure installed by the previous team, causing the first joint motor to overload. This overload led to system failure after the first successful grasp, preventing subsequent grasping operations as the first joint motor would shut down. To resolve this issue, our team decided to remove the previous 3D support structure and adjust the motion sequence to slow down the robotic arm's return to the home pose. This modification reduces the load on the first joint motor, ensuring smoother operation and improving the overall reliability of the grasping process.

1.3.3 Sleep Pose Adjustment

A conflict between the robotic arm's sleep pose and the designed structure was identified, which affected the arm's ability to transition into the sleep state and resulted in motor overload. To address this, the `go_to_sleep_pose()` function was replaced with a custom state that approximates the sleep pose. The new status parameters were set to $x = 0.15$, $y = 0$, and $z = 0.16$, allowing the robotic arm to achieve a stable and consistent resting position without structural interference.

1.3.4 Improvement in Reachability and Negative Coordinates Handling

The robotic arm was unable to set negative coordinates, and there was a lack of clarity regarding the reachable pose range of the arm. To overcome these limitations, we implemented the use of inverse kinematics functions to calculate the required joint angles for a given end-effector position. By controlling the robotic arm using these calculated joint angles, we ensured that the arm could achieve a wider range of poses, including those that may involve negative coordinates.

1.3.5 Z-Axis Overload Issue

The robotic arm encountered an overload issue when reaching negative coordinates along the Z-axis. We plan to implement a physical solution by adding components such as a spring to apply a backward force on the arm. This added force helps to relieve pressure on the joints, preventing them from being overloaded when the arm reaches extreme Z-axis positions.

1.3.6 Optimization of Exploration Targets in the Maze

We observed that in the actual maze setup, due to placement issues, the boards were not perfectly aligned, often leaving small gaps between them. Our explorer node mistakenly identified these gaps as frontiers and set exploration targets accordingly. Two solutions were implemented:

- Ensuring careful alignment of the boards during maze construction to minimize gaps.
- Adding a distance-based criterion when selecting frontiers to avoid choosing excessively small gaps as exploration targets.

1.3.7 LiDAR Pose Correction

Initially, we discovered that the robot's movement direction did not match the planned trajectory. After investigation, we found that the initial orientation of the LiDAR deviated from our intended configuration. The issue was resolved by rotating the LiDAR's pose by 180 degrees in the URDF file.

1.3.8 Wi-Fi Stability Improvement

During the initial stages of testing, we observed that the Wi-Fi signal would occasionally disconnect automatically and experience significant packet loss. To address this issue, we subsequently connected the Leo Rover and the NUC using a physical network cable.

1.3.9 Optimization of Object Detection Strategy

During our tests with YOLO for object detection, we found that it performs best when detecting targets within a range of 0.25m to 0.4m. This means that if a target block is farther away, Leo Rover may not recognize it at all. To address this issue, we are integrating color recognition using OpenCV (cv2) as an additional detection method. When Leo Rover detects the target color from a distance, it will navigate toward it, gradually closing the gap. Once the block enters YOLO's optimal detection range, the system will switch to YOLO for precise identification and initiate the robotic arm's grasping sequence. By combining color recognition with YOLO detection, we can effectively expand the robot's ability to locate target blocks and improve the accuracy and reliability of the grasping process.

1.3.10 Optimization of Intel RealSense D435i Placement

We discovered an issue with the placement of the Intel RealSense D435i. If it is mounted on the lower end of the robotic arm, it overlaps with the LiDAR, creating a blind spot in the visual field and preventing normal operation. To solve this problem, we conducted multiple comparative experiments by placing the camera in different locations and testing its performance in exploration tasks. By comparing detection success rate, detection speed, and overall system performance, we found that mounting the depth camera at the upper end of the robotic arm resulted in optimal performance and high detection efficiency. Additionally, the robotic arm has two pre-drilled mounting holes, making the installation of the Intel RealSense D435i simple and structurally stable.

1.3.11 Optimization of Support Pillars for the Flat Support Plate

We identified an issue with the number of support pillars used for the flat support plate. An excessive or insufficient number of pillars negatively affected both the aesthetic appeal and performance of the final robot. To determine the optimal configuration, we conducted experiments by varying the placement and quantity of support pillars. The results showed that fewer than five pillars led to structural instability, as each pillar had to bear excessive force. In one experiment, we used a three-pillar setup, which ultimately resulted in pillar fracture. On the other hand, using more than five pillars negatively impacted the LiDAR's efficiency, leading to object recognition failure and task execution issues. Through these experiments, we found that a five-pillar design provided the best balance between stability and aesthetic appeal, and we adopted this solution to resolve the issue.

1.4 Sustainability Checklist

This section follows the BS8622 Guide to Robot Sustainability, analyzing how the Leo Rover project considers sustainability in areas such as materials, software, energy, waste management, emissions, communication, modularity, deployment, maintenance, and reuse.

- **Materials:** Leo Rover is made from a mix of materials, with different parts using different types to balance durability and weight. For example, the robot arm is made of metal to ensure strength, while the body and support parts are made of plastic to keep it lightweight. To be more environmentally friendly, we are testing the use of recycled plastic for some components to reduce waste and pollution.
- **Software:** The object recognition and exploration algorithms are constantly being improved to reduce processing power and energy use. By optimizing how data is processed and cutting out unnecessary calculations, the software helps lower the system's overall power consumption.
- **Energy:** Leo Rover runs on a rechargeable battery, avoiding the waste and pollution caused by disposable batteries.
- **Waste:** During normal operation, Leo Rover does not produce any direct waste. However, over time, battery aging and hardware replacements may create electronic waste. To handle this, we sort and properly dispose of old electronic parts, following the University of Manchester's e-waste recycling policies, ensuring that batteries and circuit boards are sent to specialized recycling facilities.
- **Emissions:** Since Leo Rover is fully electric, it operates with almost zero emissions, meaning no exhaust fumes or fuel-related pollution. While the robot itself doesn't produce emissions, battery production and disposal may have an environmental impact. To reduce this, we prioritize materials that comply with RoHS standards, minimizing harmful substances in manufacturing and disposal.
- **Communication:** Leo Rover's communication system has been optimized to send only essential data, reducing unnecessary data transfers and saving energy. Using ROS2's message-based system and edge computing, we limit long-distance data transmissions, cutting down both processing power and storage needs.
- **Modularity:** The software is designed with a modular ROS2-based system, meaning different parts can start or stop independently without affecting the rest of the system. The hardware is also modular, with many parts made using 3D printing, allowing for quick and easy replacements without needing to discard the whole system.
- **Deployment:** Leo Rover is mainly used indoors, and it is moved and stored manually, meaning no emissions from transportation. If the robot needs to be taken to a different location, team members use public transport whenever possible to minimize the carbon footprint of travel.
- **Maintenance:** We have set up regular diagnostic checks for Leo Rover to prevent failures during experiments. The software includes real-time error alerts, helping the team quickly identify issues and perform manual inspections. The modular design also makes replacing faulty parts much easier, reducing waste from maintenance.
- **Reuse:** Once the project is finished, the team will disassemble the robot and store the parts for future use. Since Leo Rover is built in a modular way, individual components like the robot

arm, sensors, and drive system can be reused in future projects. The software, including object recognition and path planning algorithms, can also be transferred to new projects, reducing the need for redevelopment.

1.5 Cyber Security Considerations

Since this project focuses on basic functionality, the risk of cyber-attacks is relatively low. Communication between the team and Leo Rover is primarily done via SSH, connecting to the Raspberry Pi inside the robot. The team operates the robot using an Intel NUC, which is physically connected to Leo Rover via an Ethernet cable. This wired connection provides a higher level of security, reducing the potential risks associated with wireless communication. Additionally, SSH access is password protected, the password being known only to team members and regularly updated to improve security. These measures help to reduce the likelihood of unauthorized access, ensuring a reasonable level of security for the system in its current application.

2 System

This section provides an overview of the robot's key components and their roles in enabling autonomous operation. Each part has been chosen to ensure the robot can navigate, perceive its environment, and interact with objects effectively. The following descriptions explain the function and importance of each component in the overall system.

Figure 1 shows a block diagram that ensures that all parts of the robot are accounted for and their necessity is justified within the context of the robot's intended functionality.

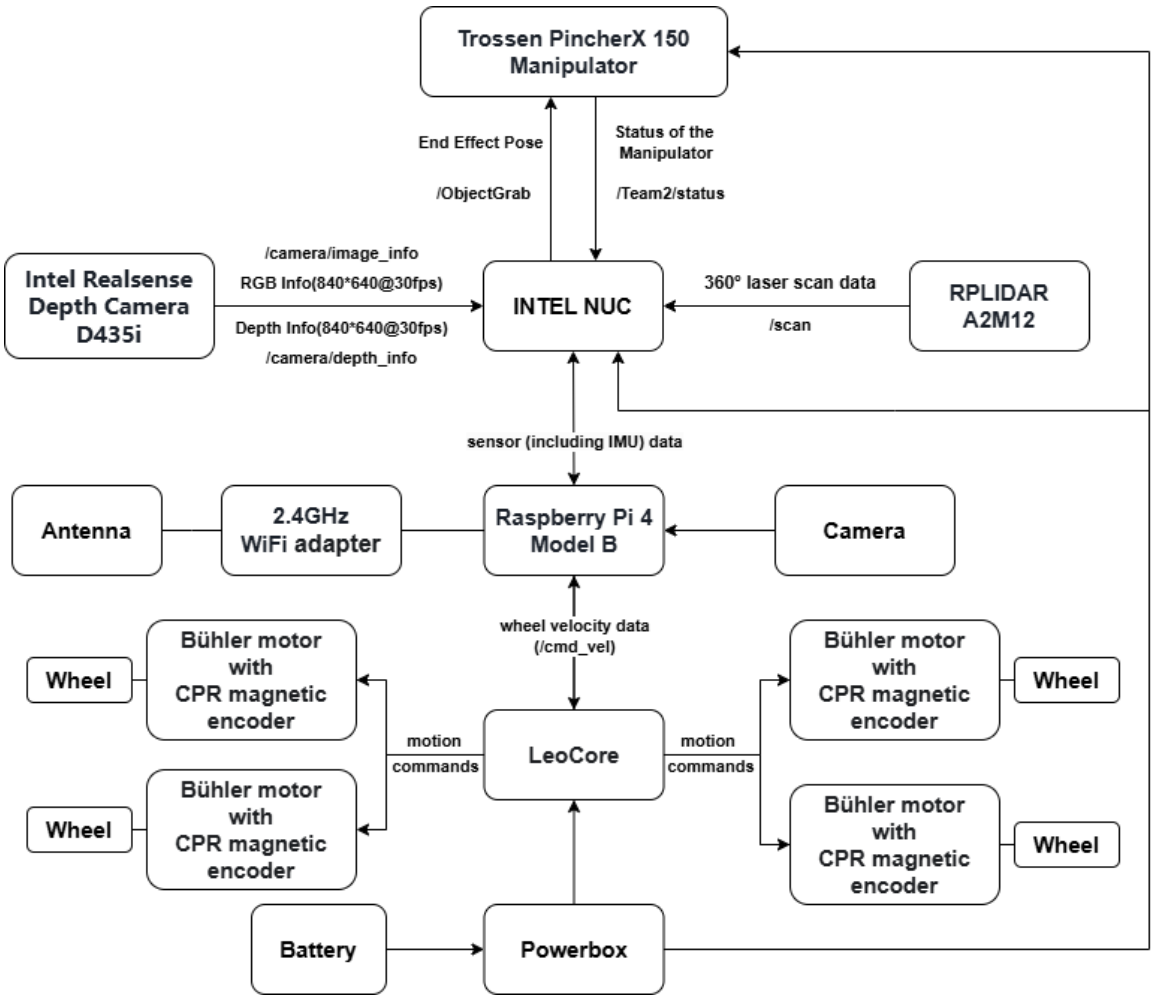

Fig. 1. System block diagram showing all the components in the robot.

- **LEO Core**: The LEO Core is the robot's central control unit, responsible for processing sensor data and controlling the robot's movements. It acts as the brain of the system, coordinating all tasks and ensuring smooth operation.
- **Intel NUC**: The Intel NUC handles the robot's intensive computation tasks. It runs complex algorithms for object recognition, decision-making, and advanced perception, greatly enhancing the robot's capabilities in autonomous operations.
- **Battery**: The robot's rechargeable battery pack powers all the components, providing the necessary energy for autonomous operation without the need for constant recharging or external power sources.
- **Powerbox**: The Powerbox is responsible for efficiently distributing power from the battery to the robot's components. It ensures stable voltage and prevents overload, allowing for reliable and safe operation.
- **RPLIDAR A2M12**: This 2D laser scanner provides precise distance measurements, helping the robot map its environment and detect obstacles. It is essential for navigation and collision avoidance in unknown environments.
- **Intel RealSense Depth Camera**: The depth camera gives the robot the ability to see and understand its surroundings in 3D, crucial for obstacle detection, spatial awareness, and more

accurate navigation.

- **Trossen PincherX 150 Manipulator**: The robotic arm enables the robot to interact physically with objects, including grasping, manipulating, and placing them. This adds an important layer of functionality for tasks like object retrieval and manipulation.
- **Raspberry Pi 4 Model B**: The Raspberry Pi handles less intensive tasks such as image processing and communication. It supports high-level algorithms for object recognition, task planning, and ensures seamless interaction with sensors.
- **Camera**: The camera captures visual data from the environment, providing essential input for tasks like object recognition, scene analysis, and navigation.
- **2.4GHz WiFi Adapter**: The WiFi adapter enables wireless communication between the robot and external systems, allowing remote control, monitoring, and data transfer in real time.

3 Mechanical Design

3.1 Mechanical Design Overview

In this design, the main design feature is that the LEO robot is connected to the robotic arm via a support platform. The support system consists of five columns and a support plane.A set of mechanical arms for gripping objects is installed in the center of the platform. Such a design ensures that the robot and the robot arm are reasonably separated to ensure that the robot arm is not subject to bumps brought by special situations of the robot and ensures that the whole system has high stability and safety.

3.2 3D CAD Design Model

The 3D CAD model is shown in Figure 2, illustrating the overall mechanical structure of the design.

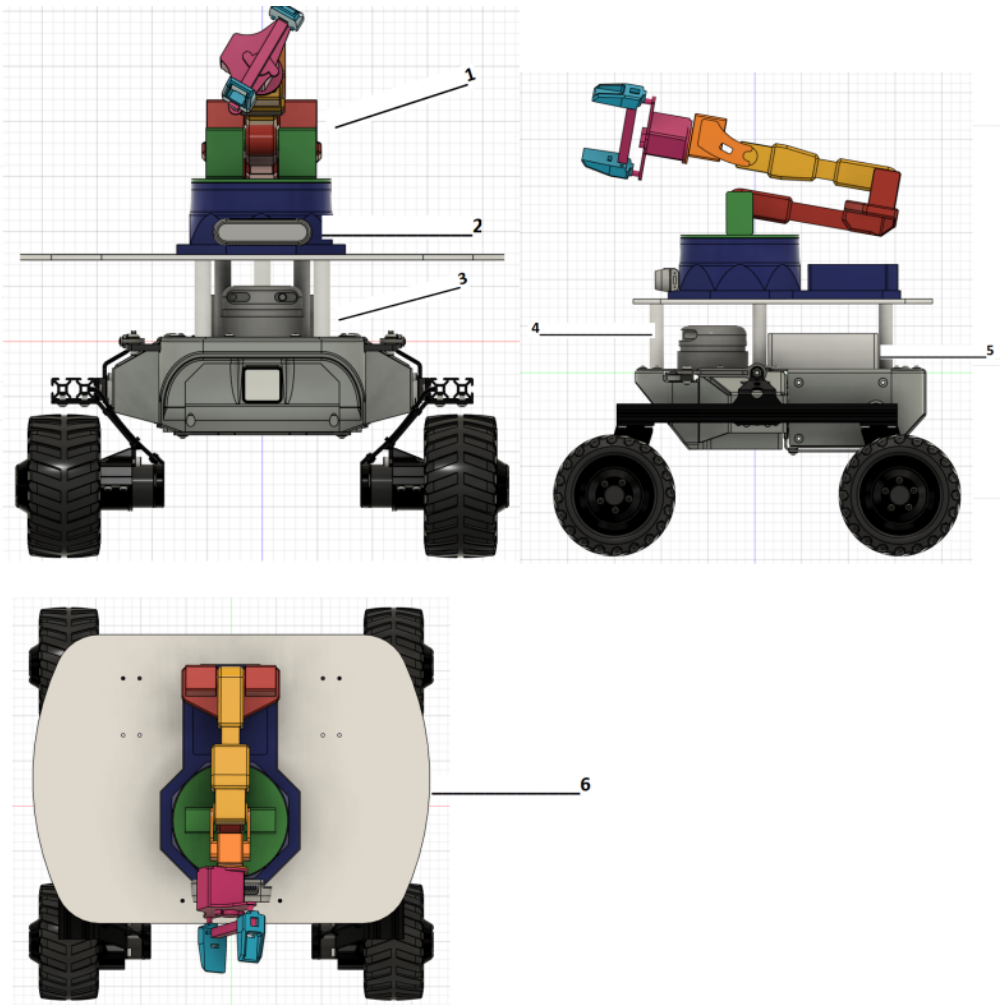

Fig. 2. 3D CAD model design - Three Orthographic Views.

- **1:PincherX** - The robotic arm responsible for grasping and manipulating objects.
- **2:Intel RealSense D435i** - A depth camera used for 3D object recognition or obstacle detection.
- **3:LiDAR** - A laser sensor used for environment perception and distance measurement.
- **4:Support Pillars** - Structural columns connecting the support plate to the base, providing stability.
- **5:NUC** - A mini-computer responsible for processing sensor data and controlling the system.
- **6:Support Plate** - The platform that connects various components, ensuring overall system stability.

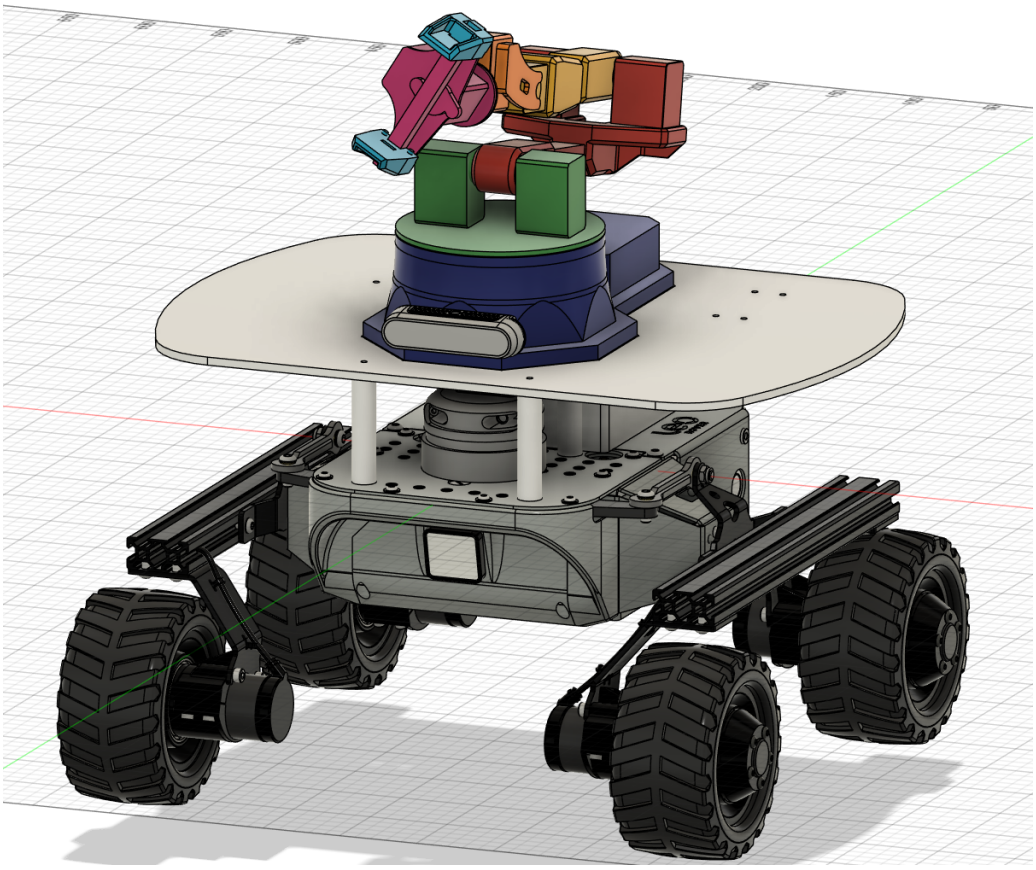

Fig. 3. 3D CAD model design - Isometric View.

3.3 Design Considerations

The main factors of this design are:

- The dimensions of the plane and its impact on stability.
- The justification for the five-pillar support system and alternative designs (e.g., three or four pillars).
- How the robot arm is connected to the robot while minimizing vibrations and blind spots for the LiDAR.
- The material required for each component, considering load capacity and manufacturability.
- The manufacturing process, including the feasibility of laser cutting and 3D printing for different parts.

3.4 Trade-offs or Challenges

3.4.1 Trade-offs and Challenges Leading to the Five-Pillar Support Structure

Trade-offs: How to design the connection between robots, robot arms, and other components is a key part of this design. The connection system should not be too complex, as it will result in enough space to place the required functions. However, it should also not be too simple, which may lead to support failure. Through eight comparative experiments, we found that:

1. .A design with more than five pillars would lead to excessive density, obstructing the LiDAR's field of view and causing misjudgments.
2. A design with fewer than five pillars would result in insufficient structural stability and an inability to provide adequate support.
3. The five-pillar configuration is also the most suitable for the LEO robot's pre-existing screw hole positions, ensuring ease of assembly.

Challenge: One of the main challenges faced in this design is minimizing the vibration transmitted from the chassis to sensitive platform components. To address this, the layout of the pillars was symmetrically designed, isolating components from the robot body through the pillars to reduce vibration effects and improve operational stability. The final design effect is shown in Figure 4, and this symmetrical five-pillar support structure successfully addresses these trade-offs and challenges.

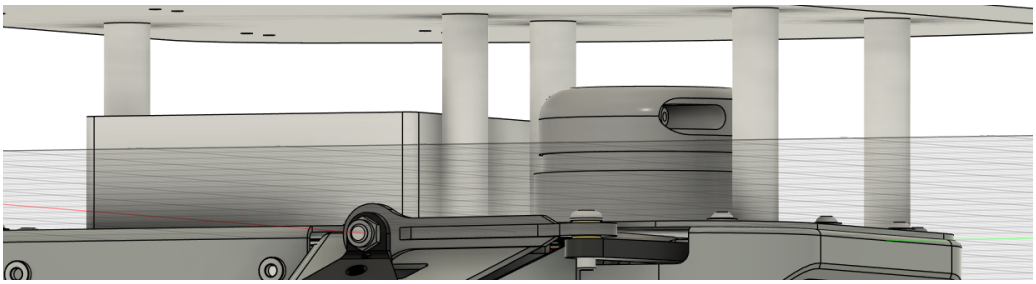

Fig. 4. Five-pillar support structure.

3.4.2 Trade-offs and Challenges Leading to the Support Platform Structure

Trade-offs: We conducted multiple experiments to compare whether a storage box should be included in the support platform. If a storage box was included, all objects needed to be placed inside the box and returned at once. Without a storage box, objects had to be picked up and returned immediately, before fetching the next one.

By comparing the completion time and success rate under both designs, we ultimately decided to remove the storage box. The task required picking up three objects in total. Adding a storage box introduced additional uncertainty, as it required precise unloading upon return. In contrast, although

removing the storage box slightly increased completion time, it significantly improved success rate and operational smoothness.

Challenge: Another key challenge in the support platform design was ensuring stability while maintaining a clean and efficient layout. By removing the storage box, the design became more reliable in task execution. However, we retained the original motor mounting holes, as they allow for future functional expansions of the robot.

Conclusion: As shown in Figure 5 and Figure 6, these two represent our final design options, with Figure 5 being our ultimate choice. The final support platform structure successfully integrates safety, stability, and adaptability. The symmetrical design ensures operational efficiency and reliability, while the preserved mounting holes provide potential for further development.

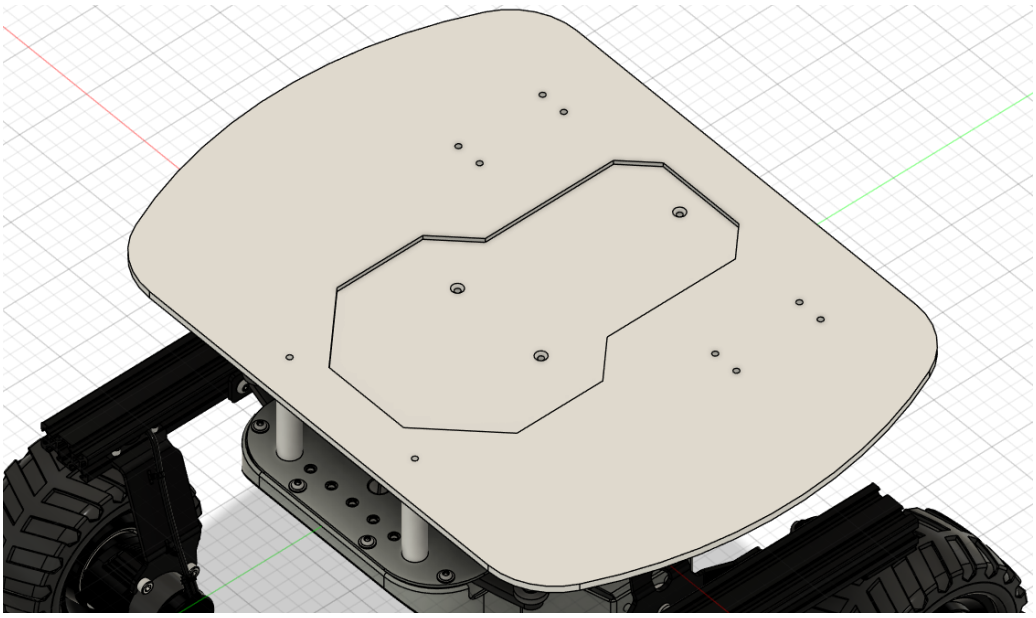

Fig. 5. Support platform structure - Final Design (Option 1).

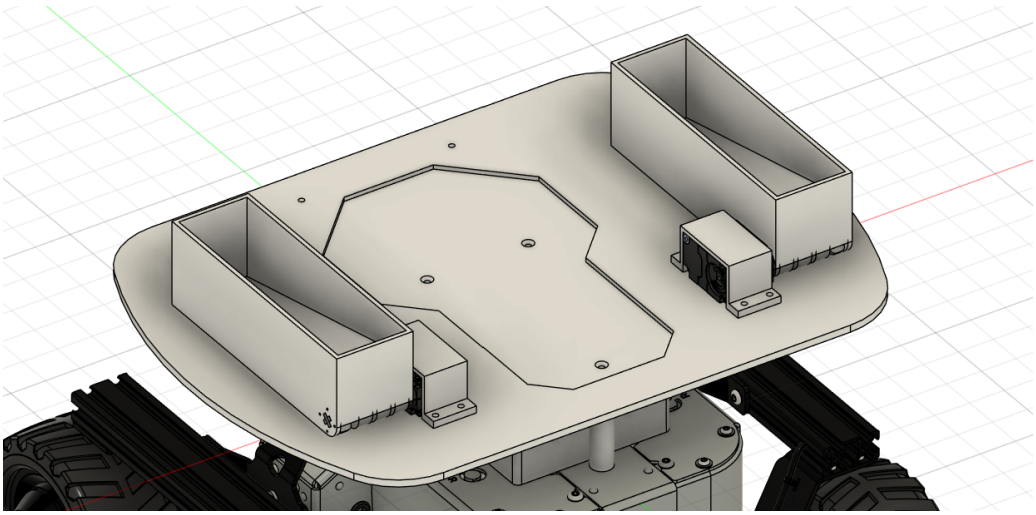

Fig. 6. Support platform structure - Alternative Design (Option 2).

3.4.3 Placement Design of Intel RealSense D435

Through five experiments, we compared multiple placement options, including mounting the Intel RealSense D435 on the robotic arm and beneath the robot body. After evaluation, we ultimately decided to place it at the first two mounting holes on the robotic arm. This decision was based on three key factors:

Easier installation :The placement does not require additional components for connection, making the setup simpler and more efficient. Time and cost efficiency:Eliminating the need for extra parts reduces assembly time and costs. Avoiding interference with LiDAR:Mounting the camera underneath the robot would overlap with the LiDAR's position, negatively affecting detection accuracy. The final design is shown in Figure 7, where the Intel RealSense D435 is the component we discussed and analyzed.

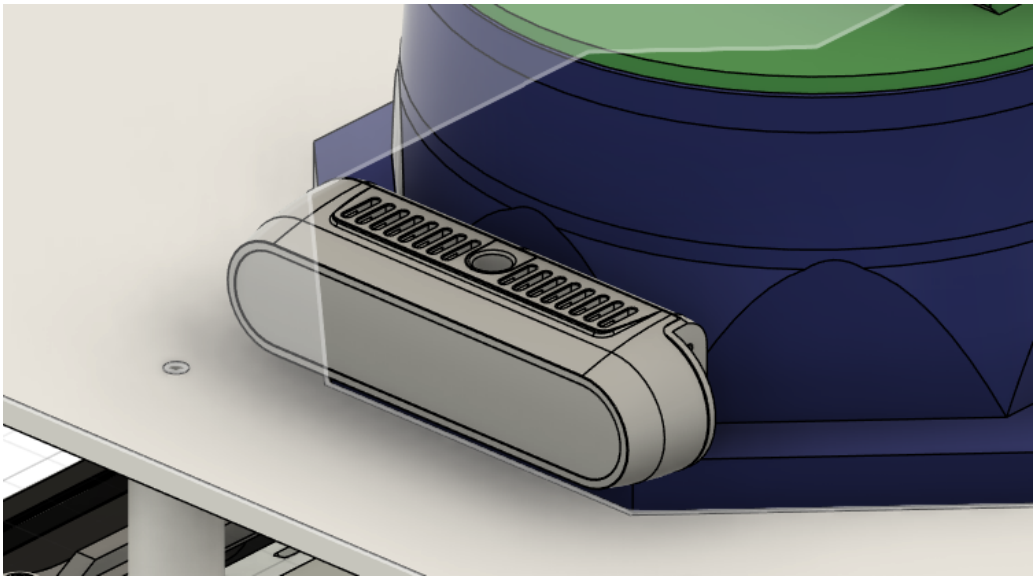

Fig. 7. Placement design of Intel RealSense D435.

3.5 Discussion on Manufacturability and Suitability of the Sled Design

Through a comparative analysis of laser cutting and 3D printing, we evaluated their feasibility for producing the support platform. Laser cutting using acrylic material results in a smoother and more precise cut, but acrylic is brittle, making it prone to manufacturing failures. Additionally, the longer production cycle of laser cutting could negatively impact the project timeline. In contrast, 3D printing produces a lighter final product, and the material filament offers a better cost-performance ratio while being less prone to breakage. This makes it more aligned with our expectations in terms of reliability and efficiency. After thorough evaluation, we ultimately chose 3D printing as the manufacturing method due to its higher efficiency, lower material fragility, and better cost-effectiveness.

4 Electrical Design

4.1 Electrical Design

Figure 8 shows the power connection section of the Leo Robot.

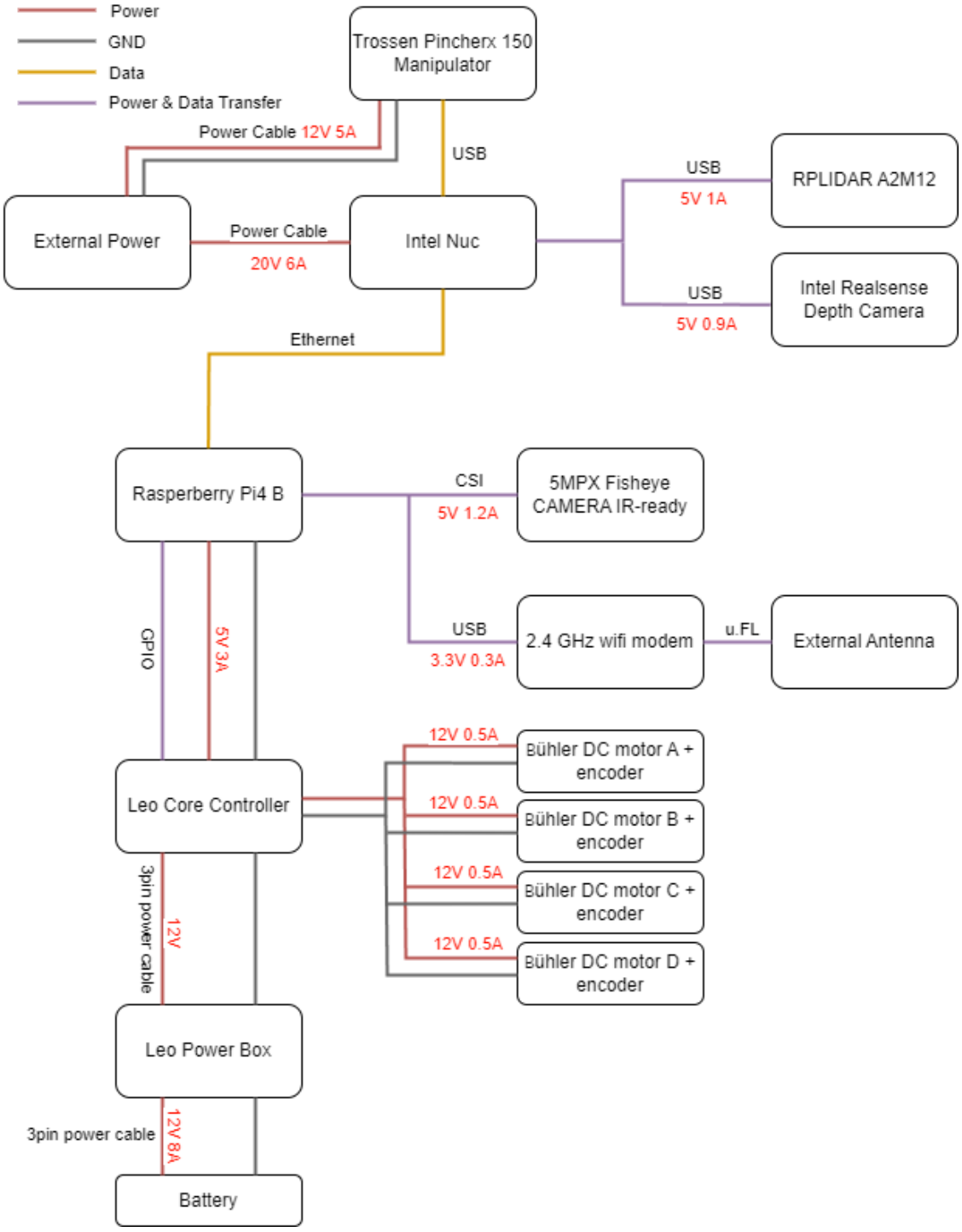

Fig. 8. Electrical connection.

Provides information on the electrical connections between various components, including details on voltage, current, and peripheral types. Since the robot requires an Intel NUC as its computing

platform, it demands a power input of 20V and 6A. However, the built-in battery of the robot provides the maximum output of 12V and 8A [1], which cannot meet this requirement. To address this, we purchased a 10m extension cable capable of supporting 120W charging to provide power for the Intel NUC, along with its connected RPLIDAR and Intel RealSense depth camera. Additionally, this external power source helps support the robotic arm, as the instantaneous maximum power demand of the manipulator and other components exceeds the battery's maximum output capacity.

4.2 Power Consumption Calculation

According to the Leo Robot's documentation, the primary power source for the Leo Robot is an internal battery. The battery operates at a voltage of 11.1V with a total capacity of 5800mAh. Using the formula below, we can calculate the battery capacity:

$$\text{Battery Capacity} = \frac{\text{BatteryCapacity(mAh)} \times \text{BatteryVoltage(V)}}{1000}$$

$$\text{Battery Capacity} = \frac{5800 \times 11.1}{1000} = 63.48\text{Wh}$$

The components consuming power from the built-in battery include:

- Raspberry Pi 4 B (5V, 3A) [2]
- 4 x Bühler DC Motors with encoders (12V, 0.5A each) [3]
- 5MPX Fisheye Camera (5V, 1.2A)
- 2.4GHz Wi-Fi modem (3.3V, 0.3A)

Since each sensor on the Leo Rover must operate in real-time to provide sufficient data for the robot to perceive its surrounding environment, all sensors are required to remain constantly active. However, the primary sensors, RPLIDAR A2M12 and RealSense Depth Camera, are connected to the Intel NUC, which is powered by an external power source. As a result, these components do not draw power from the built-in battery. Therefore, the components relying on the built-in battery for power supply are limited to: Raspberry Pi 4 B , 4 × Bühler DC Motors with encoders , 5MPX Fisheye Camera and 2.4GHz Wi-Fi modem:

$$\text{PowerConsumption} = 15\text{W} + 24\text{W} + 6\text{W} + 0.99\text{W}$$

$$\text{PowerConsumption} = 45.99\text{W}$$

Next, the duration for which the built-in battery can support all components can be calculated:

$$\text{Time} = \frac{\text{Battery Capacity (Wh)}}{\text{Power Consumption (W)}}$$

However, after conducting multiple charge and discharge tests, we determined that the current battery capacity is 90 percent of its maximum capacity:

$$\text{Usable Battery Capacity} = 63.48Wh \times 0.9 = 57.13Wh$$

Therefor the usable time for Leo Rover will be:

$$\text{Time} = \frac{57.13\text{Wh}}{45.99\text{W}} \approx 67\text{min}$$

The final demonstration is scheduled to last 20 minutes. Under these calculated conditions, we have an additional 47 minutes available for further adjustments. This shows that the power connection setup is expected to meet the power support requirements for the final test.

5 Software Design

In this project, we have divided the software part into three sub-parts which are 'Mapping and Navigation', 'Objection Detection' and 'Grasping'. We are developing these three parts in parallel and testing their functions separately before integrating the whole system.

5.1 Mapping and Navigation

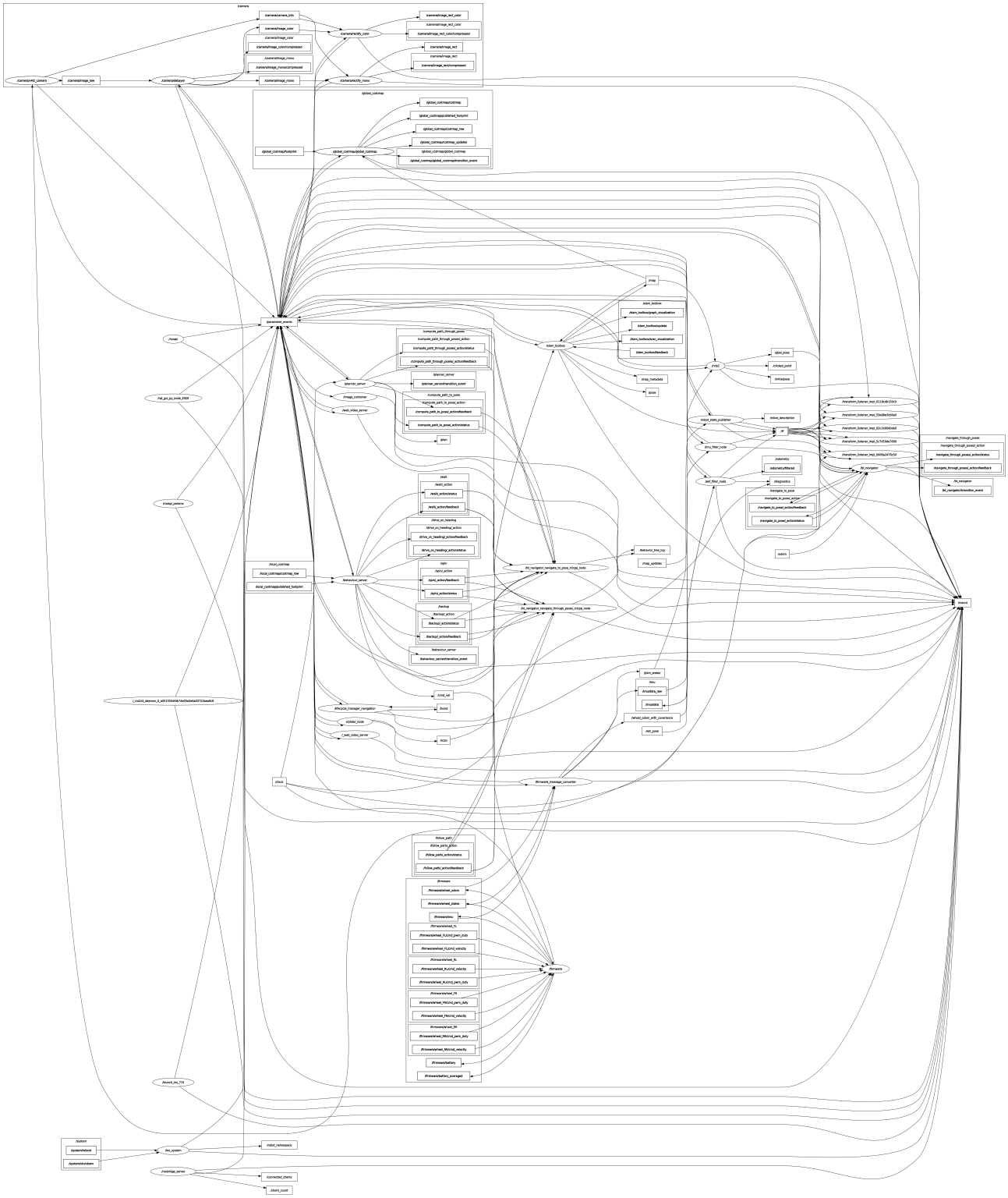

Fig. 9. RQT Graph of Mapping and Navigation.[1]

Figure 9 shows the primary ROS nodes, their published and subscribed topics, as well as various TF frames. Below is a detailed explanation of how these two parts (SLAM and exploration) function

[1] If that's not clear, we have the full image for you. `https://github.com/Lyrance/Robotics_Team2/blob/main/images/nav.png`

and interact.

5.1.1 SLAM

1. **URDF and TF Publication:** The `robot_state_publisher` reads the robot's URDF description and publishes the corresponding transforms (`tf_static`/ `tf`). This ensures a consistent spatial relationship between components, such as `base_link` and `scan`, which is essential for sensor fusion and localization.
2. **IMU Filtering and State Estimation:** The `imu_filter_madgwick` processes raw IMU data using the Madgwick filter to generate a stable orientation estimate, reducing noise. The filtered IMU data is then fed into the `ekf_node` of the `robot_localization` package, which fuses it with odometry data (if available) using an Extended Kalman Filter (EKF). This improves the robot's pose estimation and provides the `odom` $\rightarrow$ `base_link` transform.
3. **SLAM with slam_toolbox:** The `slam_toolbox` is responsible for online 2D SLAM. It subscribes to the laser scan topic (`/scan`), published by the RPLidar sensor, and optionally uses odometry or TF data to estimate and continuously refine the occupancy grid map. The package publishes the generated map (`/map`) and the transformation from `map` to `odom`, enabling global localization.
4. **Navigation:** With the live map and refined pose estimates, the navigation stack (Nav2) can plan paths and issue velocity commands (`/cmd_vel`) for autonomous movement. The robot can dynamically avoid obstacles and reach its designated goal in an unknown or partially mapped environment.

5.1.2 Exploration

In addition to basic mapping and navigation, our system supports automated exploration to autonomously discover and map unknown areas. The exploration node leverages real-time map data and navigation capabilities to systematically traverse the environment and expand the known map.

1. **Frontier Detection:** The node analyzes the live occupancy grid to locate *frontiers*, which are boundaries between known and unknown cells. These regions serve as exploration targets.
2. **Goal Selection:** Based on frontier properties (distance and size), the node computes a valid goal pose in the vicinity of a selected frontier. It also checks for navigable space to ensure the goal is feasible.
3. **Navigation Command:** The exploration node sends the goal pose to the navigation stack (Nav2) via an action, prompting the robot to travel to that location. Throughout the process, it monitors feedback or status updates from the navigator.
4. **Map Update and Repeat:** Once the robot reaches the goal or if navigation fails, the map is updated. The node then reevaluates the occupancy grid for any remaining frontiers. If new frontiers exist, it repeats the process; otherwise, exploration concludes.

5.2 Objection Detection

In this project, we utilized YOLOv8, a state-of-the-art object detection algorithm known for its accuracy and efficiency. YOLOv8 builds on the advancements of its predecessors in the YOLO series, offering improved speed and precision while maintaining a compact model size, making it suitable for real-time applications.

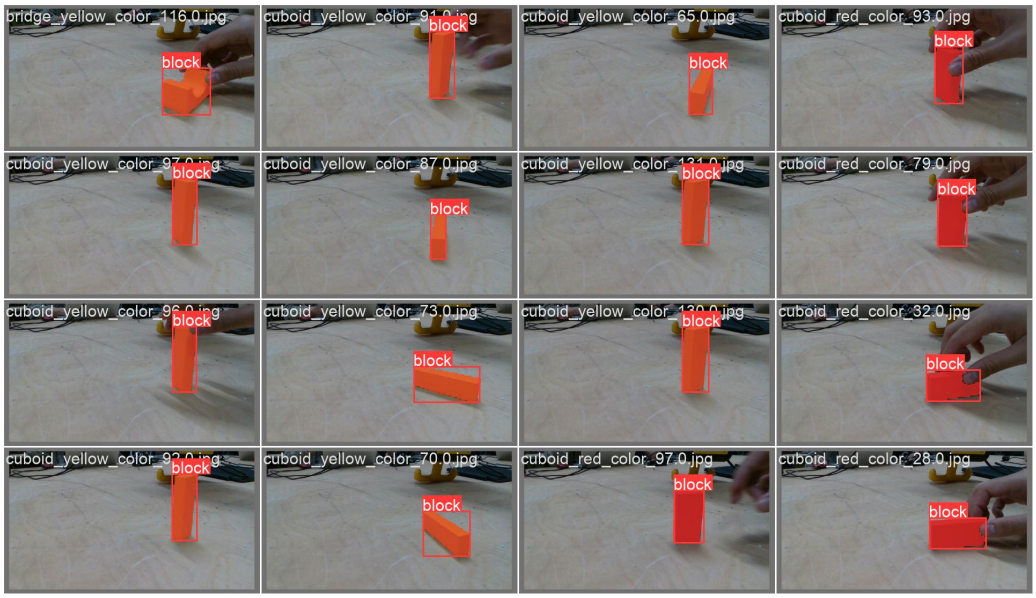

Fig. 10. Dataset with Labels.

To train the model, we collected a dataset of 2,447 images using the Intel Realsense D435i camera.(Figure 10) The dataset focuses on the specific shapes and forms of the blocks assigned to our team, capturing various configurations and angles to ensure comprehensive coverage. Each image was meticulously labeled to prepare it for training with YOLOv8.

The training process involved three iterations, with adjustments made based on the results of each round. After the final iteration, the model achieved a high level of performance. Specifically, it demonstrated a 90% detection accuracy for blocks within a 60 cm range, showcasing its effectiveness for close-range object recognition.

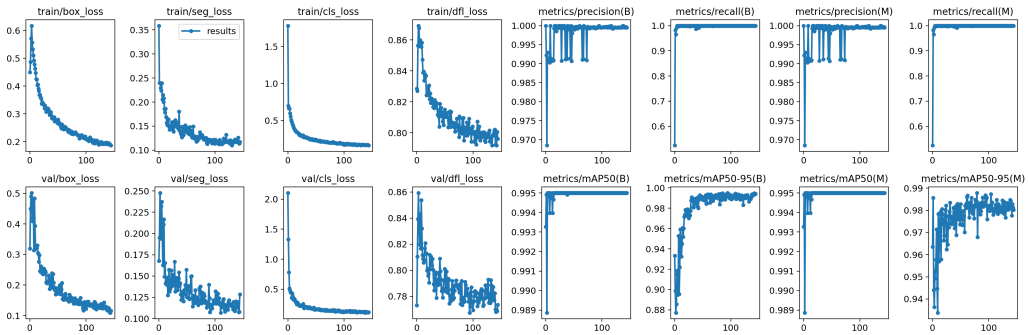

Fig. 11. Results of Model training.

Figure 11 shows the variation in model training and validation losses, as well as performance met-

rics for different training calendars.

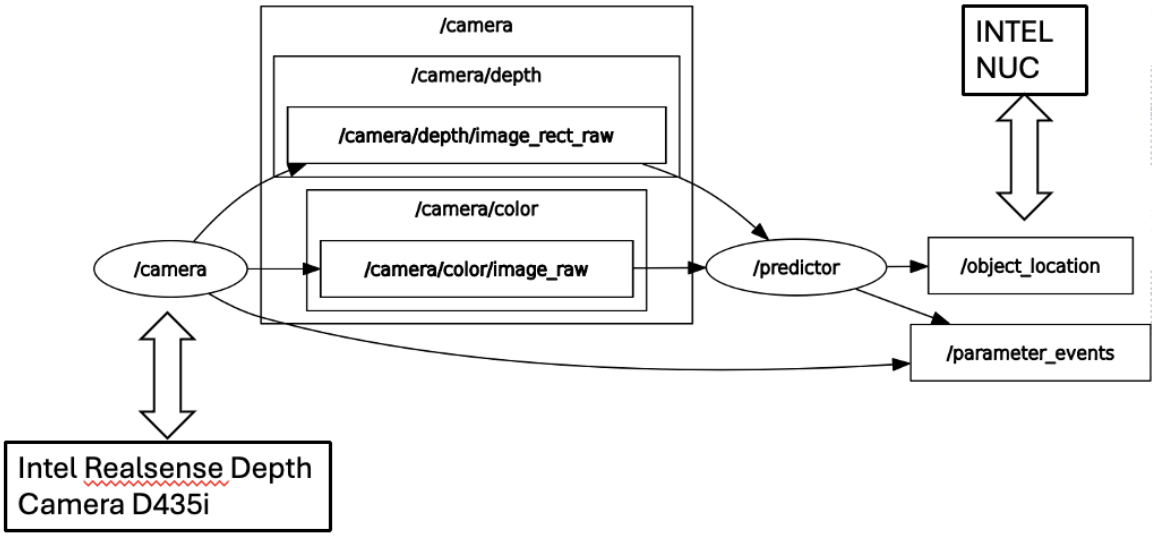

Fig. 12. RQT Graph of Object Detection.

Figure 12 shows the 'camera' node streams depth and color information at a fixed frequency by publishing depth data to the '/camera/depth/image_rect_raw' topic and color images to the '/camera/color/image_raw' topic from Intel Realsense Depth Camera D435i. Our custom 'predictor' node subscribes to these topics, leveraging the color image data to infer the target object's x, y coordinates and utilizing the depth information to calculate the distance to the object's center. Finally, the computed coordinates of the target object are published to the '/object_location' topic for INTEL NUC using.

5.3 Grasping

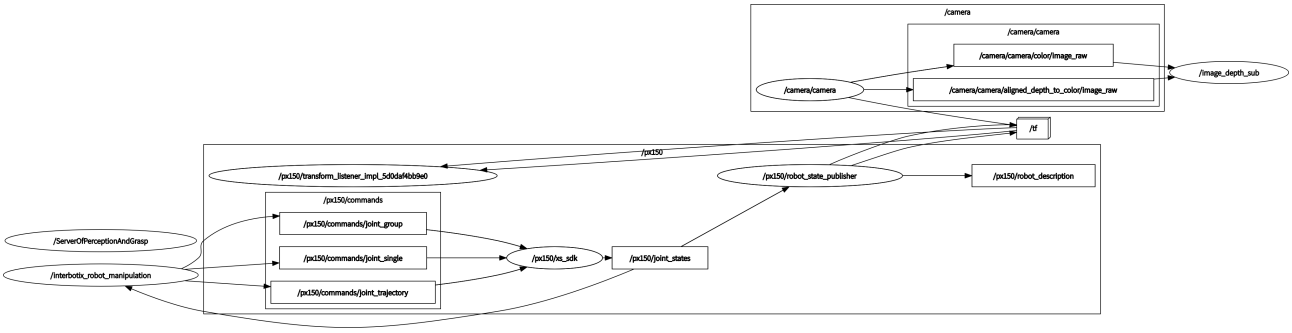

Fig. 13. RQT Graph of Grasping.[2]

The integration of the Trossen PincherX 150 manipulator into the robotic system was achieved using the 'interbotix_ros_manipulators' package. Figure 13 shows this package offers robust tools

[2]If that's not clear, we have the full image for you. `https://github.com/Lyrance/Robotics_Team2/blob/main/images/arm.png`

and interfaces for precise control of the manipulator. A key component of this setup involves the '/px150/joint_states' topic from Trossen PincherX 150 Manipulator, which publishes '/sensor_msgs/JointState' messages containing critical information such as joint names, positions, velocities, and efforts. The system's custom 'Arm_Control' node processes these joint states by interpreting data from the 'object_pose_to_camera' topic, which provides '/sensor_msgs/geometry_msgs' messages. Using this data, the node calculates the necessary joint configurations to position the end effector accurately, enabling it to grasp objects and deposit them into the retrieval mechanism.

Based on the Figure 13, the robotic arm achieves grasping through coordinated communication between nodes: First, the '/camera'-related nodes process depth images and publish the target pose to the '/object_pose_to_camera' topic; then, the '/px150/transform_listener' node converts the pose data into the robotic arm's base coordinate system, while the '/px150/robot_state_publisher' continuously broadcasts joint states. The core control node '/ServerOfPerceptionAndGrasp' integrates perception information and real-time states, issuing joint trajectory commands via '/px150/commands/joint_group' to drive the robotic arm's motion and fine-tuning the end-effector pose using '/px150/commands/joint_single'. Finally, the '/px150/commands/gripper' topic triggers the gripper's closing action to complete the grasping process. All nodes are integrated through the '/interbotix_user_manipulation' module, forming a "perception-decision-execution" closed loop.

5.4 Team's Git Repository

https://github.com/Lyrance/Robotics_Team2

6 Analysis

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
1	The robot shall autonomously navigate an unmapped environment.	The robot achieves autonomous navigation success in at least 90% of trials, stopping or rerouting within 0.5 seconds upon detecting obstacles within 0.30 m, with detection accuracy exceeding 98%. It also reliably returns to within ±0.10 m of its initial position, ensuring precise task completion.	Team2	We conducted the tests in a safe environment. A total of five different mazes were used, with each maze being tested more than ten times from different starting points. The objective was to observe whether the robot could fully explore the map while avoiding obstacles during the exploration process.	In the later stages of testing, our robot was able to perfectly explore the maze and avoid obstacles, achieving a success rate of over 90%. However, two issues remain concerning. First, the robot's edges occasionally scraped against obstacles, which was significantly improved by increasing the predefined safety distance. Second, the robot sometimes lingered around small gaps in the maze walls, which was mitigated by incorporating a distance-based criterion when selecting frontiers.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
2	Robot detects target blocks consistently under standard lighting with 90% accuracy.	The robot shall detect colored blocks in the environment with 90% or higher accuracy to ensure the manipulator can effectively perform its grasping tasks.	Li Yuliang	Multiple tests were conducted under standard indoor lighting conditions. Various colored target blocks were placed within the robot's working area, and image data was collected through the camera. The YOLOv8 model was used for object detection. The detected results were compared with the ground truth labels to verify that the recognition accuracy met the 90% requirement.	Our YOLOv8 model has undergone three evolutions. We have conducted extensive testing in indoor lighting environments and the results show that the current model has over 90% recognition of blocks up to 0.5m.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
3	Accurate coordinate data logged for the block's position.	Upon detection, the robot shall identify the center coordinates of the block within ±10% of the block's dimension because a percentage-based margin scales with the object's size, improving grasp alignment.	He Zixiang	Tests were conducted using target blocks of different sizes and positions in the experimental environment. The robot calculated the block's center coordinates through image processing and sensor data. These calculated coordinates were compared with manually measured ground truth values to confirm that the error was within ±10% of the block's dimensions.	Thanks to the accuracy of the model, we succeeded in getting the exact center coordinates every time.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
4	The manipulator shall securely grasp objects.	The manipulator achieves a 95% grasp success rate, completing at least 19 out of 20 test trials without dropping or misalignment. If the initial grasp fails, it adjusts and reattempts within 1 second, allowing up to 3 retries for successful task completion which could ensure enough time for the next step.	Team2	Grasping tests were conducted in both simulation and real-world environments using objects of various shapes and materials. Each grasp attempt was recorded, including cases where the robot automatically re-attempted failed grasps. The success rate was calculated to ensure it reached 95% or higher.	In over 50 grasping experiments, the robotic arm demonstrated near-perfect performance when grasping cubic blocks. However, some concerns remain. For instance, when handling objects of different shapes, such as cylinders, the robotic arm occasionally fails to maintain a firm grip, leading to unintended drops.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
5	The manipulator shall place the block in the storage basket accurately.	Block placement remains secure in the basket throughout operations in at least 9 out of 10 test trials, ensuring a success rate of 90% to demonstrate the robot's ability to securely store objects. If placement fails, the robot will automatically adjust its position and retry up to a maximum of 3 attempts.	Yue Zihan	Through live demonstrations and video monitoring, the robot's ability to place target blocks in the storage basket was observed. Each attempt was recorded to assess the accuracy and stability of the placement, ensuring at least 9 out of 10 successful attempts.	After extensive testing and careful consideration, we decided to remove the storage basket, leading to corresponding changes in the workflow. The specific reasons for this decision can be found in the previous sections. Therefore, this item is no longer applicable.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
6	The robot shall place the blocks in a designated area upon task completion.	Measure the error between the robot's placement of blocks and the predetermined target location to ensure the error is within ±0.05 m in at least 9 out of 10 test trials.	Li Yuliang	Demonstration tests were conducted after task completion, where the robot placed blocks in a predefined target area. Precision measuring tools were used to record the actual placement position and compare it to the target location, ensuring the error was within ±0.05 m in at least 9 out of 10 trials.	Through testing, the robotic arm achieved a 95% success rate in grasping objects and placing them into the box within the test environment. The few failure cases observed were due to either the block hitting the edge of the box and bouncing out or the grasping distance being too far, causing motor overload in one of the arm's joints, preventing it from returning to the home pose.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
7	Grasping consistently occurs within the specified time frame.	The manipulator shall grasp objects within 10 seconds once aligned in at least 9 out of 10 test trials to ensure successful performance during the final demonstration within the specified time.	Yue Zihan	During testing, the robot's grasping time was measured using a timer, starting from the moment the robot aligned with the object. Each attempt was recorded to verify that the grasping operation was completed within 10 seconds in at least 9 out of 10 trials.	We implemented a grasping time limit—if the grasping process exceeds this limit, it is considered a failure. In such cases, the robotic arm returns to the home pose and waits for a new command to retry the grasping operation.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
8	Successful alignment within tolerance range logged before each grasp.	Manipulator alignment precision shall be within ±0.010 m to ensure a secure grasp despite minor sensor or control errors.	Li Yuliang	Before each grasp attempt, precision sensors recorded the manipulator's alignment data relative to the target block. These measurements were compared with the ideal alignment position to ensure the deviation remained within ±0.010 m.	Since the robotic arm has a limited grasping range, precise alignment is crucial for successful operation. Based on current testing, the alignment process is usable but not yet perfect.
9	Storage space accommodates specified block dimensions without displacement.	Storage basket shall hold blocks up to a size of 0.040 m per side.	Hou Jiadong	We designed the basket with an internal width of 0.049 m to fulfill this requirement.	Requirement met before. However, we decided to remove the storage basket, leading to corresponding changes in the workflow. The specific reasons for this decision can be found in the previous sections. Therefore, this item is no longer applicable.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
10	Placement accuracy shall be within ±10% of the designated location.	Measure the error between the placement position and the predetermined target location to ensure the error is within ±10% of the target area's dimension from its intended placement point in at least 18 out of 20 test trials to ensure scalability and reduce misplacement.	He Zixiang	During grasping and placement tests, precision measurement tools were used to determine the deviation between the placed block's position and the intended target location. The error percentage was calculated to ensure it remained within ±10% of the target area's dimensions in at least 18 out of 20 trials.	Through testing, the placement accuracy has been significantly improved and successfully meets the required standard.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
11	Block release occurs within a time frame upon reaching the target location.	The block placement process shall be complete within 30 seconds from alignment in at least 9 out of 10 test trials to ensure successful performance during the final demonstration within the specified time.	Team2	During demonstration tests, a timer was used to measure the time taken from alignment with the target location to the completion of the block release. Each attempt was recorded to ensure the process was completed within 30 seconds in at least 9 out of 10 trials.	After reaching the target position, the control node sends a release command and starts a timer. If a success message from the robotic arm is not received within the specified time or an error is detected, the operation is considered a failure. During testing, nearly all release operations were successfully executed.
12	System reattempts up to limit, then records issue if unsuccessful.	The robot shall retry up to 3 times upon each failed grasp attempt, achieving this in at least 9 out of 10 failure scenarios to ensure consistent recovery efforts before logging an error.	He Zixiang	During testing, we monitored the number of failures and the number of retries. After the test was completed, we reviewed the error logs for further analysis.	We added a failure retry limit in the code, so if the number of retries exceeds the limit, the current action is abandoned. Additionally, system information and debugging messages are logged to facilitate future analysis.

ID	Requirement	Verification Success Criteria	Responsible Party	Verification Method	Current Results
13	State updates at required frequency logged and verified.	Robot shall report current state every 0.5 seconds for timely operator awareness without excessive communication overhead.	Team2	The runtime interface integrates the status information of various systems, including states and errors. Additionally, the update frequency of status information is monitored to ensure timely and accurate system feedback.	Our projects are able to report details to operators in real time as they run, and errors can be detected in a timely manner.
14	Error messages sent and logged within specified time.	Error notifications shall occur within 2 seconds of detection to ensure timely responses and maintain operational efficiency.	Hou Jiadong	After an unexpected behavior occurs, we check the system status and error logs to diagnose and analyze the issue.	Our projects are able to report details to operators in real time as they run, and errors can be detected in a timely manner.

7 Project Plan

Figure 14 shows the planning of various tasks in the second semester and presents it in the form of a Gantt chart.

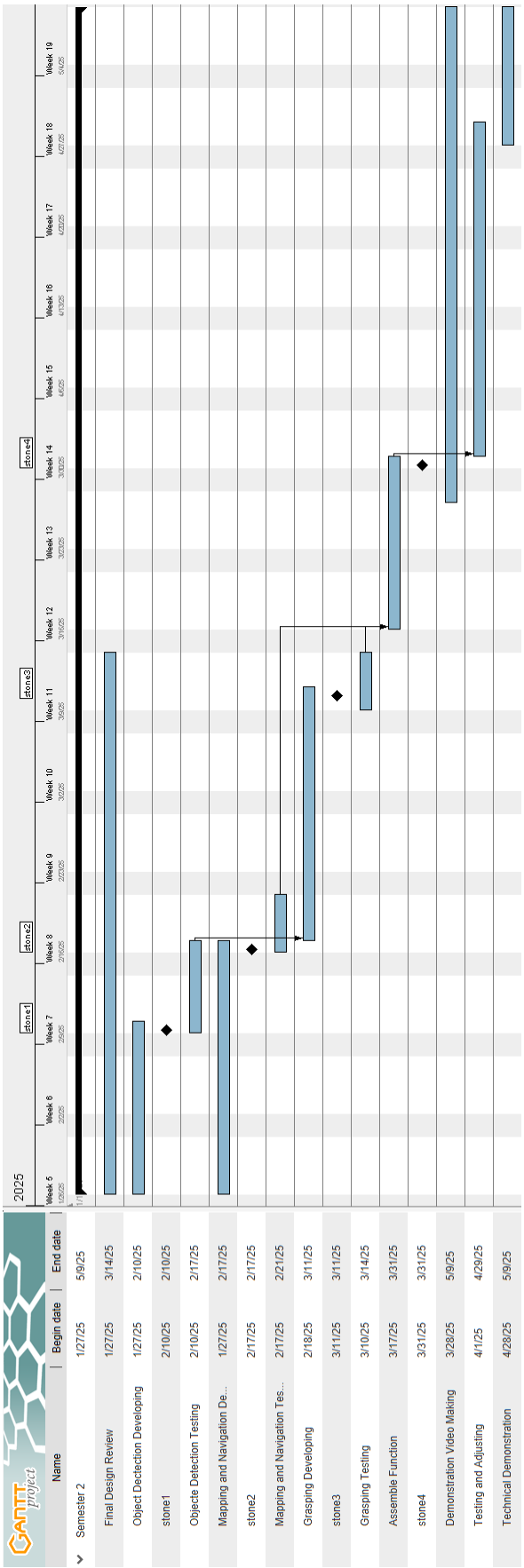

Fig. 14. Gantt chart. [3]

[3] If that's not clear, we have the full image for you. `https://github.com/Lyrance/Robotics_Team2/blob/main/images/gantt_chart.png`

Figure 15 shows the resource allocation in the second semester.

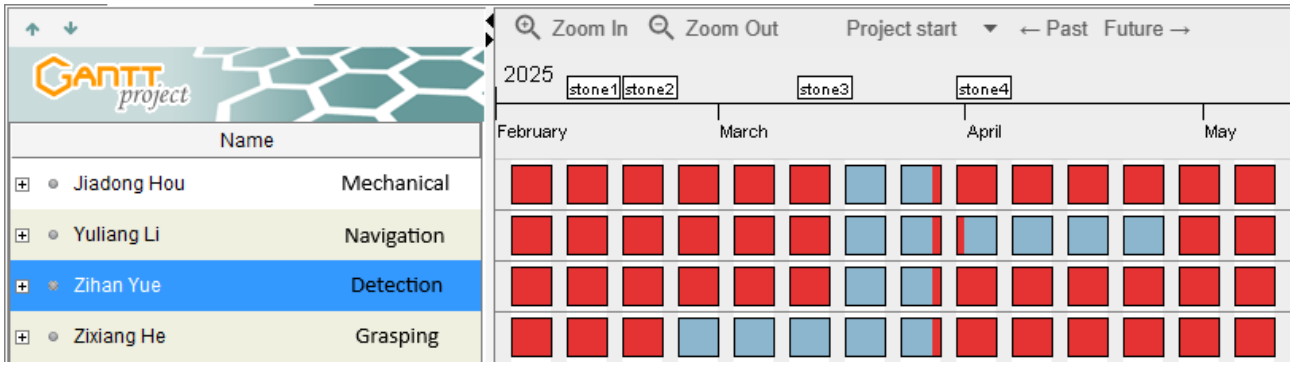

Fig. 15. Resource allocation.

References

[1] FictionLab, "Leo Rover Specification - Connection Interfaces," *Leo Rover Documentation*, [Online]. Available: `https://docs.fictionlab.pl/leo-rover/documentation/specification#connection-interfaces`. [Accessed: Nov. 25, 2024] (cited on p. 18).

[2] Raspberry Pi Foundation, "Raspberry Pi 4 Model B Datasheet," [Online]. Available: `https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf`. [Accessed: Nov. 26, 2024] (cited on p. 18).

[3] Buehler Motor, "Motors - Industrial Products," [Online]. Available: `https://www.buehlermotor.com/products-and-markets/industrial/industrial-products/motors.html`. [Accessed: Nov. 26, 2024] (cited on p. 18).