GERMAN - MONGOLIAN INSTITUTE FOR RESOURCES AND TECHNOLOGY

The present work was submitted to

the German - Mongolian Institute for Resources and Technology

Autonomous inspection and Data Gathering Robot

Bachelor Thesis

By

Soronzonbold Munkhzul

Study program: Mechatronics engineer

Student ID: B2100495

Supervisor 1: Examiner 1 Ph.D., N.Odbileg

Supervisor 2: Examiner 2 MSc. G.Otgonbayar

Ulaanbaatar/Nalaikh

2025

GERMAN - MONGOLIAN INSTITUTE FOR RESOURCES AND TECHNOLOGY

The present work was submitted to

the German - Mongolian Institute for Resources and Technology

Autonomous inspection and Data Gathering Robot

Bachelor Thesis

By

Soronzonbold Munkhzul

Study program: Mechatronics engineer

Student ID: B2100495

Supervisor 1: Examiner 1 Ph.D., N.Odbileg

Supervisor 2: Examiner 2 MSc. G.Otgonbayar

Ulaanbaatar/Nalaikh

2025

Statutory Declaration

Munkhzul, Soronzonbold	B2100495
Last Name, First Name	Student ID Number

I hereby affirm in lieu of an oath that I provided the submitted bachelor thesis

Autonomous inspection and Data Gathering Robot

I did not use any sources other than those stated. In case that the work is additionally submitted on a data medium, I declare that the written and the electronic form are completely identical. The work was not submitted in the same or similar form to any examination authority.

Ulaanbaatar, Mongolia, May, 2,2025	Соронзонболд
Place, Date	Signature

Acknowledgement

I would like to express my deepest gratitude to Ph.D. N. Odbileg, whose essential guidance, technical advice, and generous support—especially in providing the necessary electronic components—made this project possible. His mentorship and encouragement throughout this journey were invaluable.

It is also a great honor for me to have studied and applied the interdisciplinary knowledge of robotics and mechatronics during this thesis. This experience has greatly enriched my academic and professional development.

I am sincerely thankful to M.Sc. G. Otgonbayar from Oyu Tolgoi LLC for his kind support and expert insights in the field of robotics. Your assistance and professional advice were incredibly helpful and motivating.

Finally, I extend my thanks to the Oyu Tolgoi Young Development Program for supervising and supporting me throughout this project. Your contribution played a vital role in my academic journey.

Abstract

In the modern era, the robotics field is advancing rapidly, with many human labor tasks increasingly being handled by autonomous machines. For robots—especially those operating in dynamic or unknown environments—to function effectively, they must be capable of detecting and analyzing their surroundings.

This project focuses on the design and development of an autonomous inspection robot capable of environmental recognition and mapping with minimal human intervention. Using a combination of electronics and software knowledge gained both academically and independently, the robot integrates sensors, actuators, and ROS2 to perform 2D SLAM (Simultaneous Localization and Mapping). It operates within indoor environments, building maps and navigating autonomously.

Although this project is currently limited to indoor use, it lays the foundation for future development toward more complex environments, such as underground inspection. This thesis demonstrates the practical application of mechatronics skills and contributes to the growing field of intelligent autonomous systems.

Table of Contents

List of tables

List of Figures

List of Diagrams

List of Equations

List of Calculations

1 Introduction

1.1 Background

The world is currently undergoing a major technological transformation under the banner of the Fifth Industrial Revolution (Industry 5.0). Unlike its predecessor, which focused heavily on automation and artificial intelligence, Industry 5.0 emphasizes the collaboration between humans and intelligent machines. In this new paradigm, automation is not simply about replacing human labor but about enhancing human capabilities through smart integration of technology.

As automation and robotics technologies mature, we are witnessing an increasing trend of human roles being replaced or augmented by machines in various sectors. Industries are continuously moving toward autonomous systems to improve efficiency, consistency, and safety. Autonomous robots have become particularly important in environments that are hazardous, repetitive, or require precision beyond human capability. For instance, in the hospitality sector, robotic waiters have been introduced to serve customers, while in manufacturing, assembly line robots have improved production quality and speed.

1.2 Importance of automation in inspection tasks

In hazardous industries such as mining, the deployment of autonomous robots has become a critical necessity. Underground mines present numerous dangers, including unstable ground conditions, exposure to toxic gases, and the risk of accidents. The use of inspection robots in such environments can significantly reduce human exposure to risks, ensuring worker safety while also maintaining continuous operational monitoring.

Beyond workplace safety, the increasing reliance on autonomous systems impacts economic structures, technological innovation, government policy, and public health. Automation enables cost-effective operations, stimulates technological advancement, enforces new safety standards, and contributes to healthier working environments. As a result, the development of

autonomous robots for inspection and data gathering is not only a technological necessity but also an economic and social imperative.

1.3 Objectivities and scopes

The primary objective of this thesis is to design, build, test, and conduct research on an autonomous inspection and data gathering robot, with a strong emphasis on hands-on learning and the exploration of new technical domains throughout the project lifecycle.

The robot is developed to operate in semi-structured environments such as industrial facilities, laboratories, and indoor rooms, where conditions are partially predictable but may include variable obstacles and layouts. The system is equipped to perform real-time navigation and mapping using a 2D LiDAR sensor and supports the integration of additional environmental sensors, including temperature, humidity, and GPS modules, for enhanced data collection capabilities.

To achieve autonomy, the robot implements SLAM (Simultaneous Localization and Mapping) algorithms, allowing it to build a map of its environment and localize itself within that map — all without human intervention. This enables the robot to autonomously explore, avoid obstacles, and adapt to dynamic changes in its surroundings while performing inspection and data logging tasks.

1.3.1 The specific objectives of the project are as follows:

- To develop a low-cost, modular autonomous robot
- To integrate key hardware components
- To implement SLAM-based navigation
- To detect objects using camera
- To achieve reliable mobility and obstacle avoidance
- To perform experimental testing

1.3.2 Scope of the Project

This project is focused on developing a prototype-level autonomous inspection robot, intended for controlled indoor or semi-structured environments. The robot's key functional features include SLAM-based mapping, obstacle avoidance, autonomous path planning, and basic environmental sensing.

Inclusions:

- 3D-printed rover body with stair-climbing support.
- Sensor integration for mapping and potential environmental monitoring
- ROS2-based software stack for navigation and visualization.
- Power management estimation.

Exclusions:

- Complex outdoor terrain adaptation (e.g., mud, rocks, extreme weather).
- Real-time AI-based object detection or scene understanding.
- Long-term wireless data transmission to cloud or edge computing systems.

This well-defined scope ensures that the project remains achievable within the timeframe and resources of a bachelor's thesis, while still delivering meaningful technical value and demonstrating the feasibility of autonomous inspection in hazardous environments.

2 Literature review

Autonomous robots are playing an increasingly important role in environments where human presence is dangerous or inefficient. These include underground mining operations, nuclear plants, disaster response zones, and remote industrial facilities. In such settings, robots can be deployed for inspection, monitoring, and data gathering tasks, reducing the risk to human life while improving the reliability and frequency of observations.

2.1 Role of Autonomous Robots in Hazardous Environments

In particular, the mining robotics sector is experiencing rapid growth. Driven by the need to improve safety and operational precision, automation technologies are being integrated into both surface and underground mining. Robotics solutions enable remote control, real-time data collection, and continuous monitoring in harsh environments. According to global market

estimates, the mining robotics market is expected to grow from USD 1.58 billion in 2025 to USD 3.70 billion by 2034, at a CAGR of 9.91%. Despite the benefits, the widespread adoption of robotic systems in mining is still hindered by the high initial cost of deployment and integration. [1]

While high-end commercial robots offer advanced capabilities, they are often inaccessible for smaller institutions due to their cost. Meanwhile, many low-cost solutions lack the reliability, flexibility, or autonomy needed for meaningful deployment. This project aims to address these limitations by developing a mid-range, modular robot capable of navigating hazardous indoor and semi-structured environments, particularly for inspection and mapping purposes.

2.2 Technologies Enabling Autonomy

The success of autonomous inspection robots relies on technologies that support navigation, mapping, and decision-making. A core component is SLAM, which allows the robot to map its environment while tracking its location. This project uses 2D LiDAR-based SLAM with the GMapping algorithm, chosen for its reliability and simplicity in indoor settings.

To run SLAM and control functions, the system uses ROS2, a modular and scalable robotics framework. ROS2 enables real-time communication between nodes for tasks like sensing, motor control, and path planning, making it ideal for distributed robot systems.

2.3 Open-Source Design Approaches

An important aspect of this project is its reliance on open-source hardware and software platforms. The robot chassis is based on a publicly available 3D-printable stair-climbing rover design from Printables, which provides a proven mechanical foundation for mobility on uneven terrain. This open design allows for rapid prototyping, customizability, and low production cost, all of which are essential in academic and research environments.

Open-source robotic platforms like TurtleBot and Jackal have shown the effectiveness of modularity and community-driven development. By combining an open mechanical platform with ROS2 and standardized electronics, this project leverages the strengths of the open-source ecosystem.

3 System design and Architecture

The robot is developed as a modular platform combining hardware, sensors, and software for autonomous navigation, mapping, and environmental monitoring. Key components include a 3D-printed mobile chassis, multiple actuators, a suite of sensors including LiDAR and camera, and a ROS2-based control system. The architecture is designed to support real-time SLAM, path planning, object detection, and data logging in semi-structured environments.

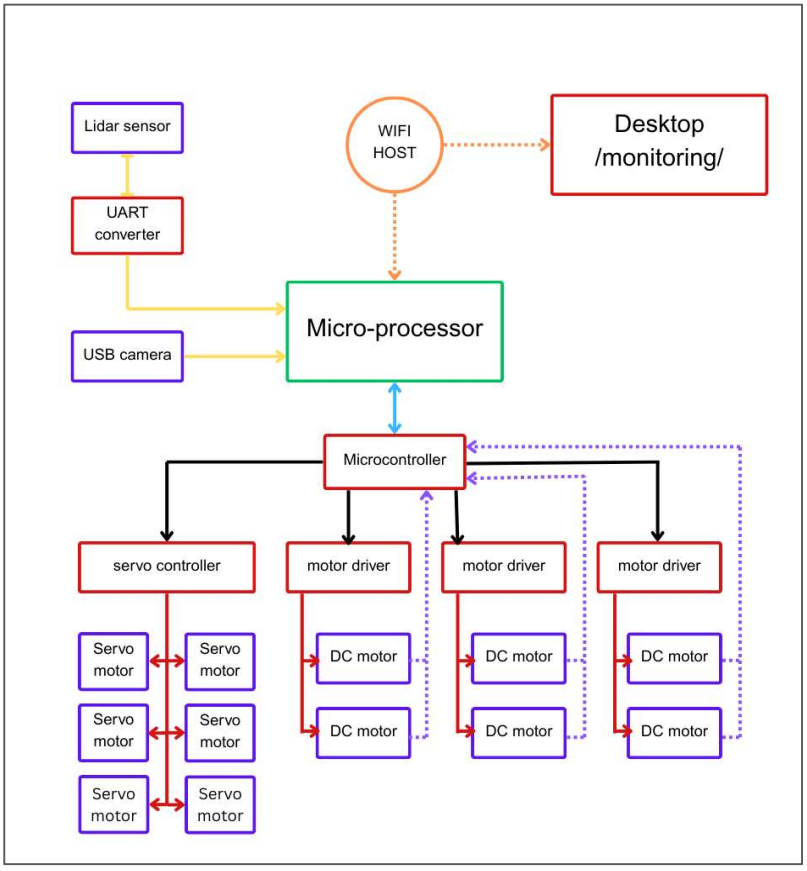


Diagram 1. Overall system architecture

3.1 Hardware layer

The hardware layer of the autonomous inspection and data gathering robot is the physical foundation upon which all sensing, computation, and motion control functionalities are built. This layer is responsible for direct interaction with the environment: gathering environmental data, detecting obstacles, and executing movement commands as determined by the software layer. It

includes the robot's mechanical frame, sensors, actuators, and power management circuits, all integrated into a compact and efficient platform.

3.1.1 Mechanical Structure and Body Frame

The mechanical structure of the autonomous inspection and data gathering robot is purpose-built to overcome challenging terrain, especially staircases, which are common obstacles in underground mining, industrial facilities, and emergency response zones. To address these challenges, the robot utilizes a rocker-bogie suspension system, a configuration originally designed for Mars rovers.

This design ensures that the robot maintains ground contact on all six wheels while traversing uneven surfaces. One side of the suspension "rocks" to adapt to elevation changes, while the opposite side remains stable ("bogie"), distributing the load across multiple points and reducing the chance of tipping. This allows the robot to climb oversteps and obstacles without the need for active balancing mechanisms or complex feedback systems. Open source design [2]

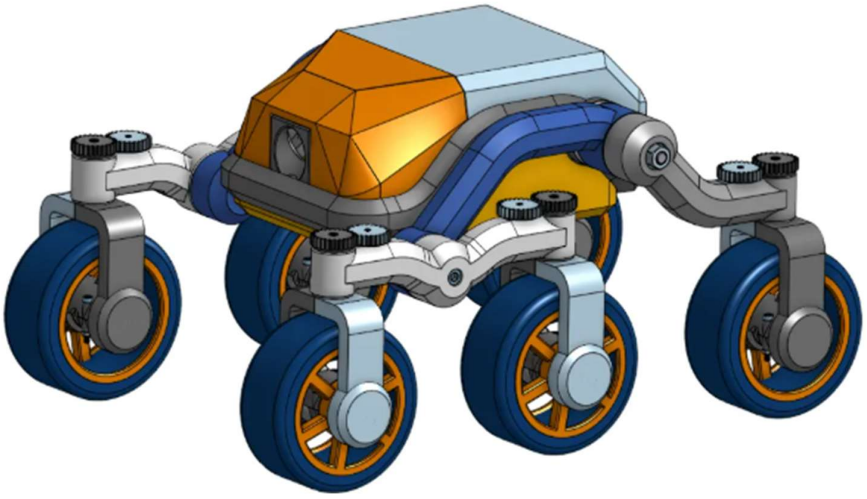


Figure 2. Stair climbing rover design source: [2]

3.1.1.1 3D Printing Tools and Materials

The robot's frame and structural components were fabricated using 3D printing, which provided the benefits of rapid prototyping, lightweight construction, and easy customization.

3D Printer Used: Creality Ender-3 V3 Plus

This printer was selected due to its excellent balance of print speed, build volume, and affordability, making it ideal for medium-scale robotics projects. Key performance features include:

- Build volume: 300 × 300 × 330 mm — capable of printing large parts like the main body frame (up to 230 × 150 mm)
- Print speed: Up to 600 mm/s (recommended ~250 mm/s for quality parts),
- Stability: Dual Z-axis and linear rails improve print accuracy for tall components.
- Auto bed leveling: Simplifies setup and ensures first-layer consistency for larger prints.

Figure 3. Ender3 V3 PLUS source: [3]

Material Used: Standard PLA

PLA was chosen for its low cost, ease of use, and dimensional accuracy. Though not as tough as engineering-grade filaments like PETG or ABS, standard PLA is well-suited for indoor robotics applications. The entire building consumed approximately 4 kilograms of filament excluding support.

3.1.1.2 Slicer Configuration and Print Settings

The slicing process was carried out using OrcaSlicer, which allows precise control over support, infill, and print quality.[10]

- Infill Pattern: Gyroid — chosen for its excellent strength-to-weight characteristics and uniform mechanical properties in all directions. This helps distribute impact loads while stair climbing evenly across the part.
- Infill Density: 40% a relatively high setting to enhance mechanical durability without making the structure excessively heavy.
- Wall Loops: 3 perimeters — for increased part rigidity and improved stress resistance at load-bearing points.

- Support Type: Tree Support — selected because it is lightweight, easy to remove, and minimizes contact with the model, reducing post-processing time and material waste.

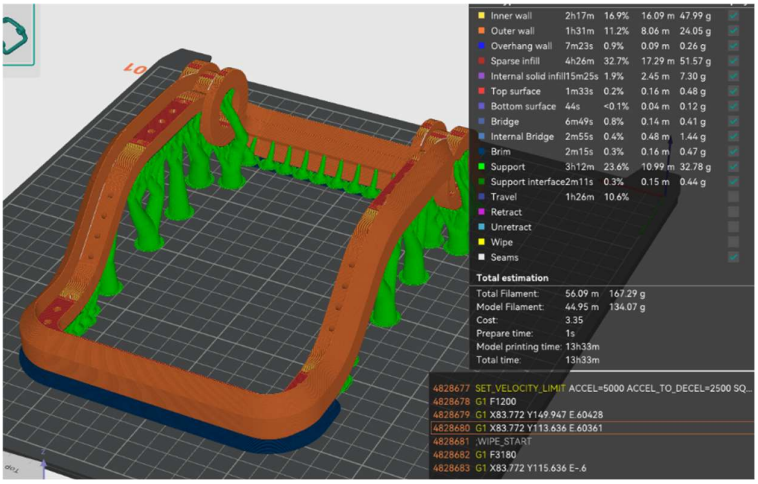


Figure 4. Slicing frame part

The resulting frame is strong, lightweight, and modular, supporting expansion or part replacement while maintaining the ruggedness required for stair navigation and rough indoor environments.[12]

3.1.1.3 Mass analysis

A fundamental aspect of the robot's mechanical and performance design is the estimation of its total mass and the distribution of that mass across critical components. Accurate mass analysis is essential for validating the torque requirements of the motors, assessing the structural strength of the chassis, and predicting energy consumption during operation.

Table 1 summarizes the mass contribution of each major component used in the robot assembly.

	Components	Quantity	Mass per unit kg	Total mass kg
1	JGA25-370 6V DC Motor	6	0.093	0.558
2	3D-Printed PLA Parts	4 | 5	1	4
3	6000mah lipo battery	1	0.425	0.425
4	Raspberry pi 5	1	0.050	0.05
5	Arduino mega	1	0.035	0.035

6	L298n	1	0.025	0.025
7	PCA9625	1	0.015	0.015
8	wire	-	0.1	0.1
9	Power bank /5v 3A output /	1	0.200	0.2
10	bolt	-	0.3	0.3
11	GDW DS041MG 5KG servo motor	6	0.018	0.108
Total				5.876 kg
Safety margin	This quantity is not realistic In the real case margin add x1.2 around for the safety			7.0512 kg

Table 1. Components mass

To ensure realistic expectations for field operation, a 20% safety margin is added to account for factors like extra wiring, structural adhesives, payload variation, and measurement inaccuracies. The final design mass of approximately 7.05 kg is used in all mechanical and torque-related calculations.

3.1.2 Drive and Motion System

The robot's mobility is achieved through a hybrid drive configuration that combines six independently driven DC motors with six servo motors. This design enables both conventional ground navigation and active articulation for stair-climbing and obstacle handling. Each drive component has been selected based on torque requirements, voltage compatibility, and physical integration with the robot's 3D-printed frame.

3.1.2.1 DC motor Configuration

The robot's primary drive system consists of six 130 RPM 6V DC gear motors with integrated encoders (model: GJA25-370). These motors are directly coupled to the wheels, enabling precise and distributed propulsion across the chassis.

Each motor includes a built-in quadrature encoder, allowing real-time feedback on shaft rotation. This feedback is essential for:

- Velocity estimation
- Precise position tracking
- Closed-loop control in ROS2 navigation and path planning

The encoder enables the robot to monitor wheel slippage, correct deviations from intended paths, and execute precise movements — especially useful during stair climbing or in constrained environments.

Performance:

Figure 5. Encoder DC motor
source: [9]

- Motor specifications at 6V:
- Nominal Speed: 130 RPM
- Nominal Torque: 0.8 kg·cm (≈ 0.078 N·m)
- Maximum Torque (Stall): 2.8 kg·cm (≈ 0.274 N·m)
- Nominal Current: 450 mA
- Stall Current: 1.8 A
- Weight: 90 g
- Encoder resolution: (specify if known, e.g., 11 PPR or 1000 CPR)

Torque Verification for Climbing Performance

To ensure the motors can generate enough force to move the robot on an inclined surface, the required torque per motor was estimated using the following expression:[20]

$$\tau_{\text{req}} = \frac{r \cdot m \cdot g \cdot \sin(\theta)}{n}$$

Equation 1. Torque

Where:

- τ_{req}= required torque per motor (N·m)
- r = wheel radius = 0.0515 m (based on tire diameter of 103 mm)
- m = total mass of the robot = 7.05 kg (including safety margin)

- g = gravitational acceleration = 9.81 m/s²
- θ = angle of incline = 30°
- n = number of drive motors = 6

Calculation:

$$\tau_{req} = \frac{0.0515 \cdot 7.05 \cdot 9.81 \cdot 0.5}{6}$$

$$= \frac{1.782}{6}$$

$$\approx \boxed{0.297\ \text{N·m}} \quad \text{per motor}$$

Calculation 1. Required torque each motor

As shown in Calculation 1. This result indicates that each motor must produce approximately 0.297 N·m of torque to propel the robot up a 30° incline under full load.

In the following step, this value is compared with the actual torque capability of the selected DC motor to assess whether the robot's drive system can meet or exceed this requirement under operational conditions.

Motor Capability and Comparison

The selected motors for this robot are GJA25-370 6V 130 RPM gear motors with encoders, each rated for a maximum (stall) torque of 2.8 kg·cm from Figure 5. To convert this into SI units:

$$\tau_{stall} = 2.8\ \text{kg·cm} = \frac{2.8 \cdot 9.81}{100} = \frac{27.468}{100} \approx \boxed{0.274\ \text{N·m}}$$

Calculation 2. Stall Torque

This indicates that each motor can deliver a peak torque of 0.274 N·m, which is slightly below the required torque of 0.297 N·m (as determined in Calculation 1). The shortfall of approximately 0.023 N·m suggests that:

- The motors operate close to their torque limit under stair-climbing conditions.
- Performance is likely adequate due to dynamic factors such as momentum, brief overload tolerance, and assistive servo lifting.

- For safer or long-duration incline navigation, options include reducing payload mass, increasing torque via gear ratio, or using lower RPM motor variants.

Despite the tight margin, this configuration is acceptable for the robot's operational goals, especially within controlled environments where slope angle and load variability are manageable.

Motor driver

To control the six DC motors efficiently, the system uses three L298N dual H-bridge motor drivers. Each L298N module can independently drive two DC motors with full control over speed and direction through Pulse Width Modulation (PWM) and logic-level input signals.

Figure 6.L298N Motor Driver source: [31]

Each L298N module includes:

- IN1/IN2 and IN3/IN4 for direction control of two motors
- ENA/ENB pins for speed control via PWM
- Power input terminals for 6V motor supply (from the LiPo battery)
- Heat sinks and thermal protection for short bursts of high-current demand during tasks like stair climbing

3.1.2.2 Servo Motor Configuration

In addition to DC motor-driven propulsion, the robot is equipped with six GDW DS041MG 5KG servo motors, which enable mechanical articulation for stair climbing and terrain adaptation. These servos are mounted at strategic joints near the wheel arms to allow the robot to lift or tilt sections of its body as needed.

Performance:

- Size: 11.9 × 32.3 × 25.4 mm
- Weight: ~12 g
- Motor Type: Coreless
- Gear Material: High-strength aluminum alloy
- Control Chip: Digital PWM-based controller
- 6v: 3.5kg.cm/0.14sec/60degree

Figure 7. Servo Motor

source: [22]

Servo Performance Verification

Each GDW DS041MG servo motor delivers a stall torque of 3.5 kg·cm (0.343 N·m) at 6.0V. To assess its suitability for stair-climbing support, the required torque to lift one wheel module (approximately 1.18 kg) over a 5 cm lever arm is calculated as:

$$\tau_{\text{stall}} = \frac{3.5 \cdot 9.81}{100} = 0.343 \text{ N·m}$$

Calculation 3. Stall Torque Servo

The estimated load on each wheel module is approximately 1.18 kg, based on even distribution of the robot's total mass (7.05 kg). Assuming the servo needs to rotate a wheel-mounted arm with a radius of 5 cm (0.05 m), the required torque for direct vertical lift can be estimated using:

$$\tau_{\text{required}} = F \cdot r = (1.18 \cdot 9.81) \cdot 0.05 \approx 0.579 \text{ N·m}$$

Calculation 4. Required Servo Torque full capacity

This calculation shows that the available torque (0.343 N·m) is lower than the required torque (0.579 N·m) for full direct lifting. However, the servo motors can still contribute significantly to stair climbing when used in conjunction with the drive motors or mechanical linkages that reduce lifting effort. Eventually it can move enough in the smooth ground.

Servo Driver

The PCA9625 is an I²C-based PWM controller produced by NXP, capable of controlling up to 16 channels independently, making it ideal for robotics applications involving multiple actuators.[11]

Figure 8. PCA9625 servo controller
source: [11]

Performance:

- Control Interface: I²C (2-wire communication)
- Number of Channels: 16 PWM outputs
- Resolution: 12-bit (4096 steps),
- Frequency Range: Programmable from 24 Hz to 1526 Hz
- Power Supply: Operates on 3.3V logic (compatible with 5V-tolerant I/O)

3.1.3 Sensor and Perception System

To achieve autonomous navigation and inspection capability, the robot integrates a compact yet effective sensor that supports environmental awareness, obstacle detection, and real-time feedback. These sensors form the core of the robot's perception layer and feed essential data into the ROS2-based software stack.

LD06 LiDAR Sensor

The primary navigation and mapping sensor is the LD06 2D LiDAR, a lightweight and high-resolution laser rangefinder capable of performing 360-degree scans. It provides real-time distance measurements used for SLAM (Simultaneous Localization and Mapping), enabling the robot to construct an internal map of its surroundings and localize itself accurately.

Specifications:

- Scan Frequency: ~10 Hz
- Angular Resolution: 1°
- Range: 0.05–12 meters
- Interface: UART Serial
- Power Supply: 5V
- Weight: ~55 g

Figure 9.LD06 LIDAR source: [4]

LiDAR publishes scan data to the ROS2 /scan topic, which is consumed by SLAM nodes such as GMapping or Cartographer. This data is also visualized in RViz, where dynamic obstacle feedback and trajectory planning are handled.

Camera Module (Logitech C270)

For visual perception and real-time monitoring, the robot uses a Logitech C270 USB camera, which offers a balance of affordability, reliability, and sufficient resolution for inspection tasks. The camera is directly connected to the Raspberry Pi 5 via USB and integrated into the ROS2 system using standard image transport nodes.

Specifications:

Resolution: 720p (1280×720

Frame Rate: Up to 30 FPS

Interface: USB 2.0

Lens Field of View: ~60°

Figure 10. Logitech c270 source: [7]

The detection pipeline captures frames using OpenCV, which are passed to the YOLOv5 model for inference. Detected objects are annotated with bounding boxes and class labels, and the results are optionally displayed on screen or converted into text outputs.

3.1.4 Power System

A reliable power system is essential for the stable operation of the autonomous inspection and data gathering robot. The design ensures separate power paths for high-current actuators and sensitive microcontrollers and sensors, while also protecting the system from electrical faults or power dips during peak operation.

LiPo Battery power supply

The robot is powered by a Li-power Li-6000mAh 3S lithium-polymer (LiPo) battery, manufactured in Guangdong, China. It provides a nominal voltage of 11.1 V (3 cells × 3.7 V each) and a capacity of 6000 mAh (6.0 Ah), which is suitable for high-drain robotic applications such as mobile platforms and drones.

Key specifications:

Figure 11. Lipo battery 6000mah source: https://shopee.com.my/LIPOWER-Lipo-Battery-6000mAh-60c-7.4v-2S-11.1v-3s-14.8v-4S-XT60-T-Plug-i.127624569.14729960333

- Model: Li-6000mAh
- Rated Voltage: 11.1 V
- Capacity: 6000 mAh
- Cycle life: >500 charge-discharge cycles
- Plug compatibility: XT60 (default),
- Size: 118 × 34 × 28 mm

This battery supports continuous current delivery for motors, servos, and control logic, and is chosen for its high cycle life, compact dimensions, and weight-to-energy ratio.

Fuse Protection

Between the battery and the converter, a fuse module is installed to protect against short circuits or current spikes. This safety feature ensures that:

Figure 12. Fuse source: https://www.ebay.com/itm/164968087331

- The battery is not damaged by overload
- Wires and components do not overheat
- Accidental shorts can be safely interrupted

A 10–15 A fuse is typically sufficient based on system load but can be adjusted depending on actual test performance.

Manual Power Switch

A switch is an electrical component that can “make” or “break” an electrical circuit. Hand actuated switch means that the switch is operated manually using physical contact. A simple mechanical SPST toggle switch is used to manually control power flow from the battery to the system. This enables:

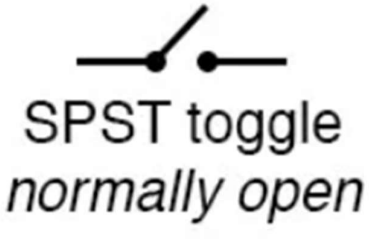


Figure 13. SPST switch source: https://www.allaboutcircuits.com/textbook/reference/chpt-9/switches-hand-actuated/

- Safe startup and shutdown
- Rapid disconnection during testing or emergencies
- Conservation of battery life during idle periods

The switch is placed after the fuse and before the buck converter.

Buck Converter

A buck converter, also known as a step-down converter, is a type of DC-to-DC switched-mode power supply that reduces the input voltage while increasing output current. Unlike linear voltage regulators that dissipate excess energy as heat, buck converters achieve much higher power efficiency, often exceeding 90%. This makes them ideal for robotic systems that require

regulated lower voltages — such as converting the 11.1 V LiPo battery to 6 V for motors and servos

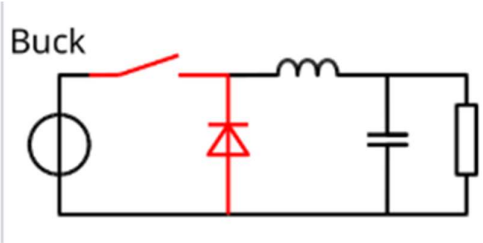


Figure 14. Buck converter scheme
Source:
https://en.wikipedia.org/wiki/Buck_converter

In this project, the SZBK07 300W 20A buck converter is used to step down the 11.1 V from the LiPo battery to a regulated 6 V output. This converter is capable of supplying up to 20 A, easily meeting the combined current requirements of all six DC motors, six servo motors, the Arduino Mega, and associated logic components.

Figure 15. SZBK07 300W 20A buck converter source: https://parts4laptops.eu/en/dc-dc-power-modules/2942-dc-dc-converter-szbk07-step-down-buck-12-36v-20a-300w-cccv-adjustable.html

The SZBK07 features:

- Adjustable constant voltage (CV) and constant current (CC) modes
- An integrated heatsink and cooling fan
- Overcurrent and thermal protection

Its performance and power handling capabilities make it well-suited for the robot's high-torque tasks such as stair climbing and dynamic path correction, while maintaining electrical stability under load.

System Integration

The combination of a high-capacity LiPo battery, robust buck converter, and clear separation between control and actuation power domains results in a stable and modular

electrical system. The power distribution block diagram in Diagram 2 illustrates how energy flows from the battery to each major subsystem.

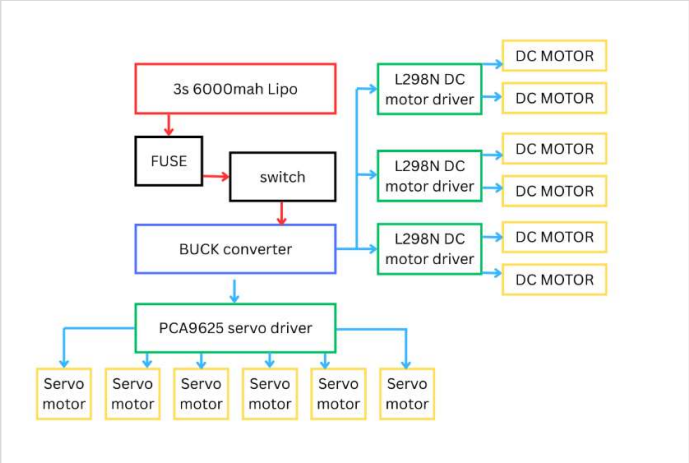


Diagram 2. LiPo battery usage

Diagram Explanation

From Diagram 2:

- The red line represents the 11.1 V output directly from the 3S 6000 mAh LiPo battery.
- This voltage is first passed through a fuse (for protection) and a manual switch (for safe control).
- The blue line indicates stepped-down 6 V output generated by the SZBK07 buck converter, which delivers regulated voltage to all actuators and logic components.

The green boxes (L298N motor drivers and PCA9625 servo driver) receive the 6 V input and distribute it to their respective outputs:

- The L298N modules control the six DC motors, with each module managing two motors.
- The PCA9625 PWM driver independently controls the six servo motors, ensuring coordinated wheel articulation for stair climbing.

This modular design allows independent control and power delivery to high-current devices (DC and servo motors), while keeping the Raspberry Pi and its peripherals powered separately

via a USB power bank. This power isolation prevents voltage dips or noise from affecting the core processing and sensor systems.

Power calculations

To evaluate the power requirements of the robot and verify the suitability of the power supply system, each component's voltage, current draw, and quantity were analyzed. The total current and power consumption were calculated for both the 6 V and 5 V systems.

$$P = I * V$$

Equation 2. Power

Component	Voltage	Current (each)	Quantity	Total Current	Power (W)
DC Motors	6V	450 mA (avg)	6	2.7 A	6 × 6V × 0.45 = 16.2 W
Servo Motors	6V	300 mA (avg)	6	1.8 A	6 × 6V × 0.3 = 10.8 W
Arduino Mega	5V	0.15 A	1	0.15 A	0.75 W
Raspberry Pi 5	5V	2.5 A	1	2.5 A	12.5 W
LD06 LiDAR	5V	0.25 A	1	0.25 A	1.25 W
Logitech C270	5V	0.2 A	1	0.2 A	1.0 W
Others (PCA9625, sensors, losses)	5V	~0.3 A	1	0.3 A	1.5 W
Total					44 W

Table 2. Power consumption

From Table 2. The system was designed to handle this load with margin. The SZBK07 converter, rated for up to 300 W, easily supports the 44 W actuator load. These calculations confirm that the chosen power system provides sufficient current and power for reliable operation during continuous movement, stair climbing, and sensor data collection.

3.2 Software Architecture

The software system of the autonomous inspection robot is based on ROS2 Jazzy Jalisco, running on Ubuntu 24.04 across both the control and visualization systems. It is designed to support modular, distributed robotic functions including SLAM, object detection, remote access, and actuator control.[15]

3.2.1 Operating System Configuration

To ensure compatibility, Ubuntu 24.04 is used on both the Raspberry Pi 5 and the laptop, as ROS2 Jazzy is the supported ROS version for this Ubuntu release. Older ROS2 distributions such as Humble are incompatible with 24.04, making version alignment essential for networking and topic sharing.

3.2.1.1 Raspberry pi 5 Micro-processor

The Raspberry Pi 5 (8GB model) is a compact, high-performance single-board computer developed by the Raspberry Pi Foundation. It is widely used in robotics due to its small form factor, GPIO pin access, and ability to run a full Linux OS.

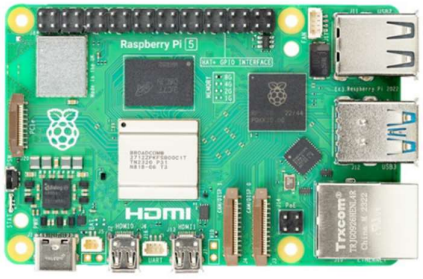


Figure 16. Raspberry Pi 5
Source: [13]

Key hardware features include:

- CPU: 64-bit 2.4 GHz quad-core ARM Cortex-A76 processor
- RAM: 8 GB LPDDR4X-4266 SDRAM
- Connectivity: 2 × micro-HDMI, Gigabit Ethernet, USB 3.0, USB 2.0
- Power: USB-C 5 V/5 A input
- Expansion: 40 GPIO pins, PCIe support

This hardware enables efficient real-time communication with sensors, motor drivers, and the host system, making it well-suited for mobile robotics.

3.2.1.2 Desktop or Laptop

The laptop used for ROS2 development and visualization runs Ubuntu 24.04 Desktop in a native dual-boot configuration. Remote desktop options were tested but found to be unreliable for real-time graphical tasks such as RViz2 visualization. A native installation was therefore selected for full access to the Linux environment and GPU acceleration. The laptop is equipped with an Intel Core i5 10th processor, 16 GB RAM, and a 256 GB SSD, providing enough resources to support real-time SLAM visualization and map saving.

Network Configuration via Mobile Hotspot

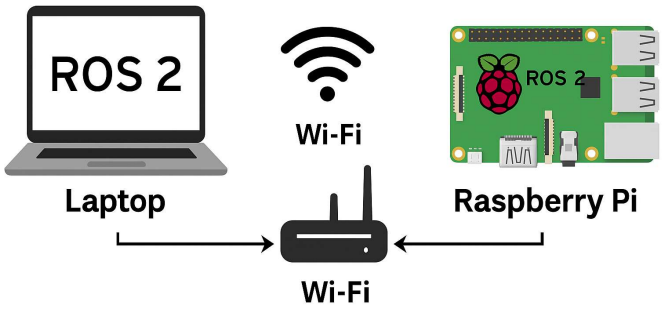


Figure 17. Devices connection

From Figure 17. To establish seamless communication between the Raspberry Pi 5 and the laptop, both devices were connected to the same network using a mobile hotspot. This approach ensured that the devices shared the same subnet, facilitating reliable ROS 2 node discovery and communication.

Setup Process:

1. Mobile Hotspot Activation: A mobile hotspot was enabled on a smartphone, creating a local Wi-Fi network.

2. Device Connection: Both the Raspberry Pi 5 and the laptop were connected to this hotspot, ensuring they were on the same network.
3. SSH Communication: With both devices on the same subnet, Secure Shell (SSH) connections were established, allowing for remote access and control of the Raspberry Pi from the laptop

3.2.2 ROS2 Configuration

Robot Operating System 2 (ROS 2) is a modular and distributed robotics middleware framework that facilitates communication between various components in a robotic system. It provides tools and libraries for building robot applications, enabling developers to design complex systems with relative ease.

From Figure 18. ROS 2 distributions are versioned sets of ROS packages, akin to Linux distributions. Each distribution is released on May 23rd annually, coinciding with World Turtle Day. Distributions released in even-numbered years are designated as Long-Term Support (LTS) versions, supported for five years, while those in odd-numbered years have a standard support duration of 1.5 years.

Distro	Release date	Logo	EOL date
Iron Irwini	May 23rd, 2023		November 2024
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		December 9th, 2022
Foxy Fitzroy	June 5th, 2020		June 20th, 2023
Eloquent Elusor	November 22nd, 2019		November 2020
Dashing Diademata	May 31st, 2019		May 2021
Jazzy Jalisco	May 2024	TBD	May 2029

Figure 18. ROS2 Distribution

Source: https://docs.ros.org/en/foxy/Releases.html

The selection of a specific ROS 2 distribution is crucial, as it determines compatibility with system dependencies and the stability of the development environment. For this project, ROS 2 Jazzy Jalisco, released on May 23, 2024, was chosen. Jazzy Jalisco is an LTS release, ensuring support until May 2029, and is compatible with Ubuntu 24.04, the operating system used on both the Raspberry Pi 5 and the laptop.

3.2.3 Remote Communication and SSH Workflow

Secure Shell (SSH) is a cryptographic network protocol designed for secure communication over unsecured networks. It enables users to remotely access and manage devices, such as servers or embedded systems, by providing encrypted terminal sessions. SSH employs public key cryptography for authentication and encrypt data transmissions to ensure confidentiality and integrity. This makes SSH an essential tool for remote system administration, especially when physical access to the device is impractical or impossible.

The Raspberry Pi 5 operates as a headless device, meaning it functions without a directly connected monitor, keyboard, or mouse. To facilitate development and monitoring, SSH is utilized to establish a secure connection between the Raspberry Pi and the laptop. This setup allows for:

- Remote Code Deployment and Execution: Developers can write and execute code on the Raspberry Pi without physical access.
- Live Terminal Access: Real-time monitoring of ROS 2 topics, bag recording, and debugging is possible through the terminal.
- Resource Efficiency: By avoiding the need for a graphical user interface on the Raspberry Pi, system resources are conserved, enhancing performance.

3.2.4 Visualization and Mapping

Since the Raspberry Pi server lacks a graphical desktop environment, all visualization tasks are handled on the laptop:

- RViz2 is used to display LiDAR scans, build SLAM maps, and track robot position
- The Pi publishes /scan and TF topics over the network
- The laptop subscribes to these and renders in real-time

This distributed visualization model ensures lightweight sensor publishing on the Pi and full graphical processing on the laptop, improving overall system efficiency.

3.2.5 Challenges and Setup Considerations

- Setting up the ROS2 environment was a complex process that required
- Manual installation of ROS2 Jazzy Jalisco from source (if done before official packages)
- Networking setup using static IP or shared Wi-Fi
- Proper sourcing of environment files (setup.bash)
- Matching Python versions and package dependencies
- For beginners, the combination of Linux terminal commands, ROS2's decentralized architecture, and debugging cross-device communication can be especially challenging.

4 Implementation

4.1 Hardware Assembly

The hardware design and assembly of the autonomous inspection and data gathering robot focused on modularity, functionality, and durability. Physical structure, actuation, power delivery, control units, and sensor integration were carefully selected and constructed to support autonomous SLAM-based navigation and stair-climbing capability in semi-structured environments.

4.1.1 Chassis and Mechanical Structure

The robot chassis was based on an open-source 3D-printable stair-climbing rover model from Printables [12]. All structural components were printed using PLA filament with the following specifications:

- 3D Printer: Ender 3 V3 Plus
- Slicer Software: OrcaSlicer
- Infill Density: 40%
- Infill Pattern: Grid (chosen for strength and print speed balance)
- Support Type: Tree support (for easy removal and low material usage)
- Wall Loops: 3
    - Total PLA Used: ~4–5 kg

The robot is assembled mostly with M3 nuts and bolts. The arms are supported by a long M8 threaded rod. The top lid is attached with magnets. The fully assembled robot is about 430x330x220 mm (LxWxH).

Figure 19. Printed robot

Part Name	Quantity	File Name	Notes
Top arm front	1	1x_1xm_top_arm_front.stl	Structural arm component
Top arm back	1	1x_1xm_top_arm_back.stl	Opposing arm half
Arm lower	2	2x_arm_lower.stl	Connects arms to body
End stop	2	2x_end_stop.stl	Limits arm travel
Arm coupler	1	1x_1xm_top_arm_coupler.stl	Connects top arm sections
Frame	1	1x_frame.stl	Main body chassis
Bottom	1	1x_bottom.stl	Base platform
Top back	1	1x_top_back.stl	Rear upper casing
Top front	1	1x_top_front.stl	Front upper casing
Rim	6	6x_rim.stl	Wheel rims
Motor arm 1	6	6x_motor_arm1.stl	Motor support (side 1)
Motor arm 2	6	6x_motor_arm2.stl	Motor support (side 2)
Motor mount	12	12x_motor_mount.stl	Motor-to-chassis adapters
Mount coupler	6	6x_motor_mount_coupler.stl	Extra motor mounting support
Small bevel gear	1	1x_1xm_small_bevel_gear.stl	Transmission gear
Large bevel gear	1	1x_large_bevel_gear.stl	Central drivetrain gear
Differential mount	1	1x_differential_mount.stl	Gear mounting base
Pusher	1	1x_1xm_pusher.stl	Load interaction element
Gear servo	6	6x_gear_servo.stl	Gear interface to servos
Gear	6	6x_gear.stl	General power transmission gear

Table 3. Robot Mechanical printed parts

From the Table 3. list of 3D-printed components reflects the modular and customizable nature of the robot's mechanical design. By utilizing open-source models and customizing them to suit the functional requirements of autonomous navigation and stair climbing, the mechanical structure balances robustness, weight, and printability. Each printed part was chosen to contribute to the overall system's functionality—whether in structural support, actuation, or drivetrain. The modular assembly allows for ease of maintenance and reconfiguration, supporting future improvements or experimentation.

4.1.2 Drive system

The robot's locomotion is powered by six GJA25-370 6V DC gear motors, each rated at 130 RPM with integrated quadrature encoders. Each pair of motors is controlled via a dedicated L298N dual H-bridge motor driver, allowing for independent directional control and speed modulation using PWM signals. These drivers are connected to an Arduino Mega 2560, which acts as the motor controller and interfaces with the Raspberry Pi 5 via serial communication.

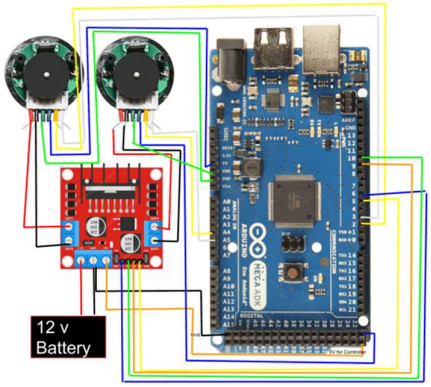


Figure 20. DC motor driver connection

Source: https://forum.arduino.cc/t/motors-drive-properly-while-using-library-with-l298n-driver-but-no-encoder-values-are-picked-up-arduino-mega/1054564

From Figure 20. That illusrates the Encoder Dc motor connection between arduino mega bridging L298N Motor driver.

Importance of Encoder Feedback

Encoders are critical for measuring the actual rotational movement of the DC motors. Unlike open-loop control, where the motor speed and position are estimated based only on the input PWM signal, encoders provide real-time feedback, enabling closed-loop control. This is particularly important for:

- Speed regulation: Ensuring each motor rotates at the desired rate, even under load.
- Position tracking: Supporting path correction and odometry when integrated with SLAM systems.
- Slippage detection: Identifying discrepancies between intended and actual movement.
- Differential control: Matching wheel speeds across opposite sides for straight movement.

From Figure 20. connect an incremental quadrature encoder (like the ones in your GJA25-370 motors) to two digital input pins on the Arduino, it provides two types of critical data.

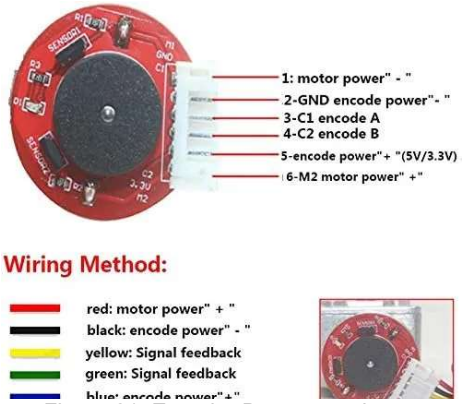


Figure 21. Encoder Dc motor scheme

Source: https://mostelectronic.com/shop/motors-drives-2/dc-motors-motors-drives-2/dc-gear-motor-with-magnetic-linear-encoder-jgy-370-15kg-97rpm-12v-motor-bracket/?add-to-cart=9928

1. Rotation Count (Position)

- Each rising or falling edge of the signal on either channel (A or B) represents a change in shaft position.
- By counting these pulses, you can determine how many degrees or revolutions the motor shaft has turned.

2. Direction of Rotation

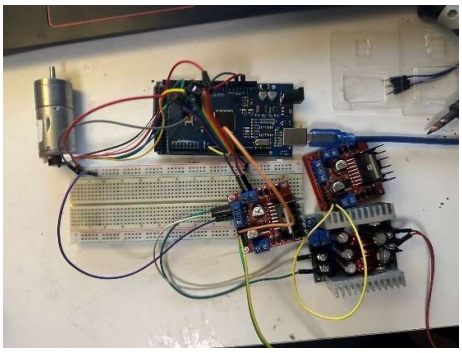
- The two encoder signals (Channel A and Channel B) are 90° out of phase.
  - By comparing which signal leads the other:
  - If A leads B → the shaft is rotating in one direction (e.g., clockwise).
  - If B leads A → the shaft is rotating in the opposite direction (e.g., counterclockwise

DC Motor and Encoder Testing

To verify encoder functionality, a quadrature encoder was connected to digital pins 2 and 3 of an Arduino Mega. The interrupt-driven approach was used to count pulses and determine direction. The wheel radius was 5.1 cm, and the encoder provided 990 pulses per revolution after gear reduction. The distance traveled was calculated using the formula:

$$Distance = N \times 2 * pi * r/ppr$$

Equation 3. Wheel distance



- N = Number of encoder pulses
- r = Radius of the wheel (cm)
- PPR= Pulses per revolution (after gearbox) =11*90= 990

Figure 22. Encoder motor testing

$$\text{Distance per pulse} = \frac{\text{Circumference}}{\text{Pulses per revolution}} = \frac{32.04\ \text{cm}}{990} \approx 0.0324\ \text{cm/pulse}$$

Calculation 5. Distance per pulse

Figure 23. shows the Code: 1 Encoder testing code is result. That can define the traveled distance for the well. This test successfully tested.

Output Serial Monitor ×

Not connected. Select a board and a port to connect automatically.

```
09:07:12.308 -> Encoder Count: -7 | Distance: -0.23 cm
09:07:12.535 -> Encoder Count: -48 | Distance: -1.55 cm
09:07:12.713 -> Encoder Count: -77 | Distance: -2.49 cm
09:07:12.923 -> Encoder Count: -107 | Distance: -3.46 cm
09:07:13.114 -> Encoder Count: -154 | Distance: -4.98 cm
09:07:13.328 -> Encoder Count: -180 | Distance: -5.83 cm
09:07:13.538 -> Encoder Count: -206 | Distance: -6.67 cm
09:07:13.748 -> Encoder Count: -266 | Distance: -8.61 cm
09:07:13.934 -> Encoder Count: -343 | Distance: -11.10 cm
09:07:14.136 -> Encoder Count: -413 | Distance: -13.37 cm
09:07:14.339 -> Encoder Count: -413 | Distance: -13.37 cm
09:07:14.559 -> Encoder Count: -413 | Distance: -13.37 cm
09:07:14.743 -> Encoder Count: -414 | Distance: -13.40 cm
09:07:14.953 -> Encoder Count: -452 | Distance: -14.63 cm
09:07:15.173 -> Encoder Count: -523 | Distance: -16.93 cm
```

Figure 23. Encoder test result

4.1.3 Raspberry Pi and Arduino Communication

In this system, the Raspberry Pi 5 serves as the central processor, responsible for high-level logic, mapping, and navigation. The Arduino Mega 2560, on the other hand, handles low-level motor and servo control. To allow these two components to operate in coordination, a serial communication interface is used.[16]

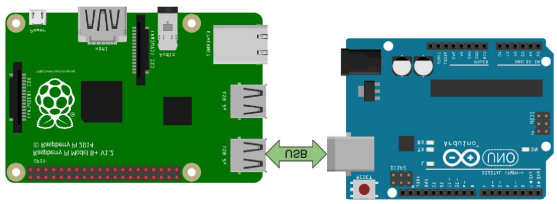


Figure 24. Serial Communication

source: [16]

To establish communication between the Raspberry Pi and Arduino, the Python script utilizes the serial.Serial() function from the PySerial library. This function requires several key parameters:

- Port name: This is the device file associated with the Arduino when connected via USB, typically /dev/ttyACM0 or /dev/ttyUSB0 on Linux systems. The actual path can be verified using the ls /dev/tty* command after plugging in the Arduino.
- Baud rate: This must match the value configured in the Arduino sketch to ensure accurate data transmission. In this project, a baud rate of 9600 is used on both the Raspberry Pi and Arduino. A mismatch here would result in corrupted or unreadable data.
- Timeout: The timeout parameter defines how long the read() function waits for data before giving up. A timeout of 1 second is set to prevent the program from hanging indefinitely if no data is received. If the complete message is not available within the timeout period, the function will return whatever bytes have been received up to that point.

4.1.4 Sensor | LD06 lidar

The robot's ability to perform real-time mapping and obstacle detection is enabled by the integration of the LD06 2D LiDAR sensor, a compact, high-precision scanning module. The LD06 provides 360° distance measurements using a rotating laser system, making it ideal for SLAM-based navigation in indoor and semi-structured environments.

for specific interfaces:

Number	Signal Name	Type	Description	Min	Typical	Max
1	Tx	Output	Radar data output	0V	3.3V	3.5 V
2	PWM	Input	Motor control signal	0V	-	3.3 V
3	GND	Power Supply	Power negative	-	0V	-
4	P5V	Power Supply	Power positive	4.5V	5V	5.5 V

Figure 25. LD 06 performance

source: [4]

UART-to-USB Communication Setup

Figure 26.USB-UART converter
source: [24]

The LD06 communicates over a UART interface, but the Raspberry Pi 5 does not provide a native UART port that is plug-and-play for the sensor.

To solve this, a USB-to-UART converter based on the CP2102 chip was used. This module converts TTL UART signals into a USB virtual serial port that can be read by the Raspberry Pi.

Converter Pin	Connected To (LD06)	Description
5V (VCC)	P5V	Power supply to LiDAR (5V logic)
GND	GND	Common ground
TX	RX of Raspberry Pi via USB	Data sent from LiDAR

Table 4. UART converter and Lidar connection

From Table 4. That shows wiring connection between lidar and converter. When plugged into the Raspberry Pi, the converter appears as a serial device such as /dev/ttyUSB0. The LD06 sensor outputs distance and angle data as structured packets, typically at 230400 baud rate. This data is then processed by a ROS2 node to publish Lasers can messages for SLAM and visualization.

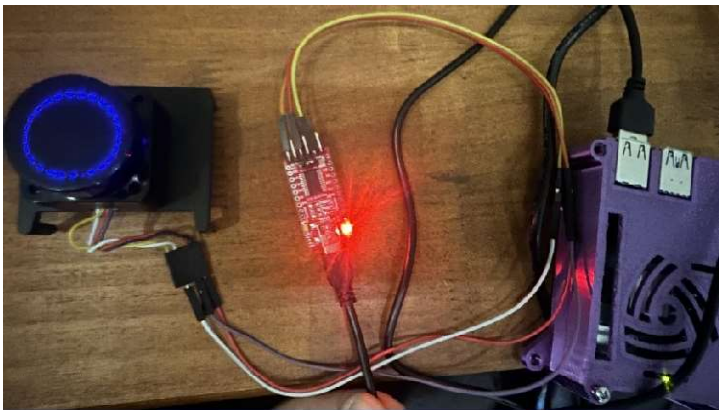


Figure 27. Actual connection lidar

From Figure 27. Lidar is powered by raspberry pi because the motor light is on. That shows this connection works well.

4.2 Software Stack and Configuration

The autonomous inspection robot operates using a modular software stack based on Ubuntu 24.04 and ROS2 Jazzy Jalisco. This combination was chosen to support multi-device communication, distributed processing, and real-time visualization using open-source tools. The software architecture divides tasks across two main computational nodes:

- Raspberry Pi 5 (server edition): Handles sensor data acquisition, actuator commands, and ROS2 publisher nodes.
- Laptop (desktop edition): Used for visualization, simulation (RViz2, Gazebo), and high-level algorithm development.

Together, they operate as a unified ROS2 network, communicating over a shared local Wi-Fi hotspot. This structure allows for resource optimization while maintaining real-time robotic capabilities.

4.2.1 Operating System and ROS2 Setup

To ensure compatibility and synchronous operation between devices, both the Raspberry Pi 5 and the laptop were configured with Ubuntu 24.04, which is the first version officially supporting ROS2 Jazzy Jalisco.[17]

Raspberry Pi 5 Setup

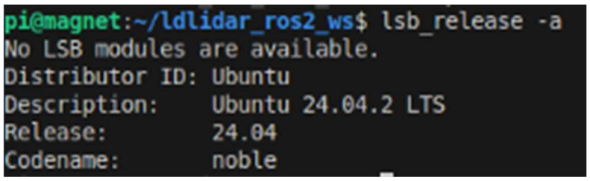


Figure 28. Raspberry pi operation system check

Chosen for its minimal resource usage, ideal for real-time publishing tasks.

Headless operation (no GUI) with SSH access enabled.

Key configuration steps:

- Ubuntu image flashed using Raspberry Pi Imager.
- SSH was enabled by default and used for all headless setup tasks.
- ROS2 Jazzy Jalisco installed with ros-base package:
    - Bash sudo apt install ros-jazzy-ros-base
- Environment setup via:
    - Bash echo "source /opt/ros/jazzy/setup.bash" >> ~/.bashrc
    - source ~/.bashrc

This configuration ensured efficient publishing of LiDAR and motor feedback topics from the Raspberry Pi.

Laptop Setup

The laptop (used for visualization, simulation, and SLAM processing) runs From Figure 26. Ubuntu 24.04 Desktop, installed via dual-boot alongside Windows 11. Remote desktop alternatives such as VNC and RDP were evaluated but found inadequate for real-time 3D visualization in RViz2 and Gazebo. A native installation was ultimately chosen for performance and stability.

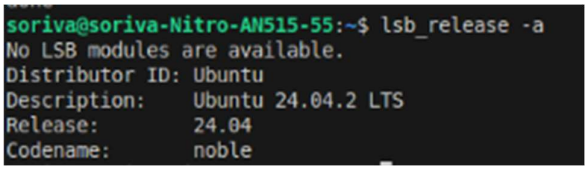


Figure 29. Laptop Operation system check

OS: Ubuntu 24.04 Desktop (Dual Boot)

Full GUI environment used for visualization and SLAM simulation (via rviz2 and gazebo)

This setup provides access to ROS2 development tools, visualization software (RViz2), and simulation environments like Gazebo.

That configuration only works if both devices have the same ROS2 distribution installed—in this case, ROS2 Jazzy Jalisco, which is fully compatible with Ubuntu 24.04. Mixing different ROS2 versions across devices can cause message mismatches, dependency conflicts, or DDS discovery failures.

4.2.2 ROS2 Network configuration

In order to enable multi-device ROS2 operation, it was essential to establish a local network between the Raspberry Pi 5 and the laptop. This configuration allowed distributed ROS2 nodes to discover each other and share topics in real time. [17]

Wi-Fi Hotspot-Based Local Network

Instead of using a router or complex LAN setup, a smartphone hotspot was configured, and both the Raspberry Pi 5 and the laptop were connected to it. This allowed them to operate under the same IP subnet, which is a prerequisite for ROS2 nodes to discover each other using the DDS (Data Distribution Service) protocol.

This approach provided:

- Network independence
- On-the-go compatibility during field tests
- Avoidance of firewall restrictions

Environment Variable Configuration

ROS2 requires specific environment variables to be configured in order to allow communication between machines:

- Bash:
    - export ROS_DOMAIN_ID=0
    - export ROS_LOCALHOST_ONLY=0

These lines were added to the. bashrc file on both devices to ensure they are automatically applied in every terminal session:

- Bash:
    - echo "export ROS_DOMAIN_ID=0" >> ~/. bashrc
    - echo "export ROS_LOCALHOST_ONLY=0" >> ~/. bashrc
    - source ~/. bashrc

ROS_DOMAIN_ID=0: Ensures both devices belong to the same ROS domain.

ROS_LOCALHOST_ONLY=0: Enables communication across IP addresses instead of restricting to localhost.

Once these settings are applied, ROS2 nodes on both the Raspberry Pi and laptop can communicate over the same network.

Verifying IP and Network Status

In Figure 27, the command sudo arp-scan --localnet was executed to scan the local Wi-Fi network for all connected devices. This command identifies each device's IP address and MAC address.

- The laptop running the scan has an IP of 172.20.10.4.
- The devices detected on the network include:
- 172.20.10.1 – the default gateway (typically the smartphone hotspot).
- 172.20.10.3 – likely the Raspberry Pi (based on MAC address or process of elimination).

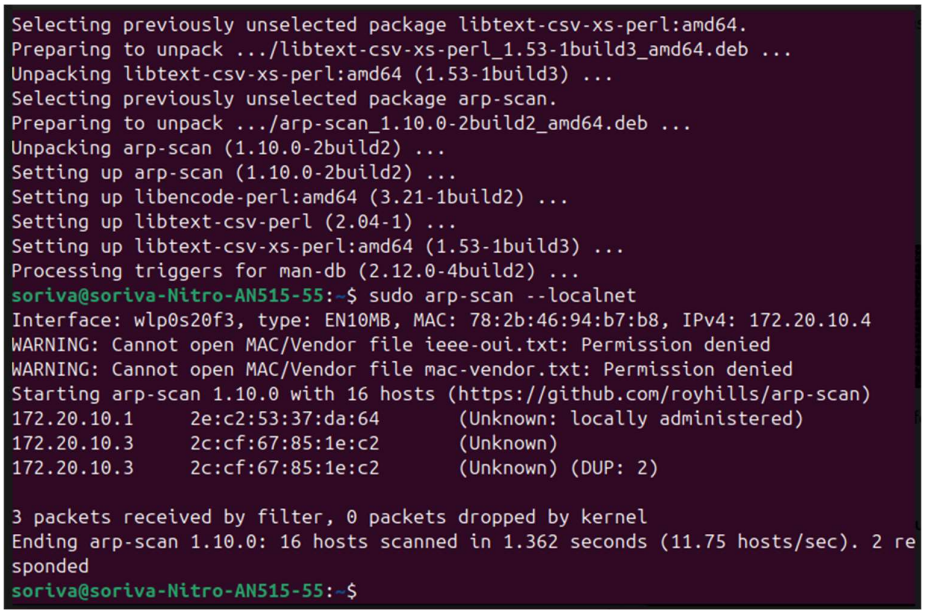


Figure 30. IP network scan

The results confirm that both the Raspberry Pi and the laptop are connected under the same subnet (172.20.10.x), which is essential for ROS2 discovery protocols to function properly over Wi-Fi. This method is especially useful when a display is not attached to the Raspberry Pi, as it helps identify the device's IP for SSH access and ROS2 multi-device configuration.

4.3 Lidar sensor integration

This section outlines the integration and functional testing of the LD06 2D LiDAR sensor, which serves as the primary perception component for environmental mapping and obstacle detection. The sensor plays a central role in enabling SLAM (allowing the robot to build maps and localize itself in semi-structured environments.

4.3.1 ROS2 Integration and Launch

From Figure 28. The ROS2 driver used is ldlidar_stl_ros2, which supports the LD06 sensor out of the box. After installing the package and dependencies, a custom launch file was created to initialize the LiDAR and begin publishing data to the /scan topic:

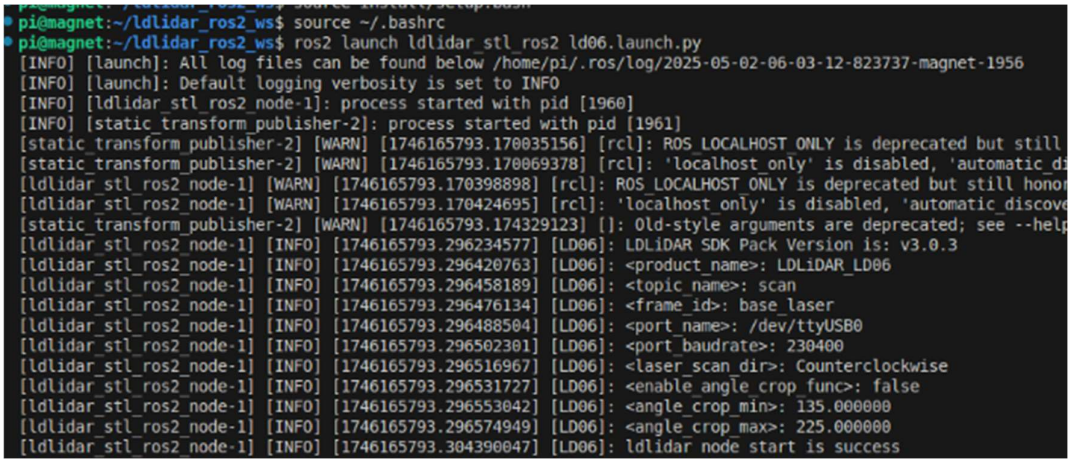


Figure 31. Raspberry pi publishing /scan

Figure 31 shows the process of initializing the ROS2 workspace and launching the LiDAR node. After sourcing the appropriate environment (both the local workspace and /opt/ros/jazzy/setup.bash), the ros2 topic list command confirms that the /scan topic is active. This verifies that the LIDAR is successfully publishing laser scan messages over ROS2.

4.3.2 Data Visualization and Testing

To visualize the laser data and validate scan quality, RViz2 is used. RViz2 is a powerful 3D visualization tool in ROS2 that supports a wide range of sensor displays. After launching rviz2, the /scan topic is added with the LaserScan display type. As seen in Figure 32, the scan data appears as colored points in the 3D scene, accurately depicting surrounding obstacles.

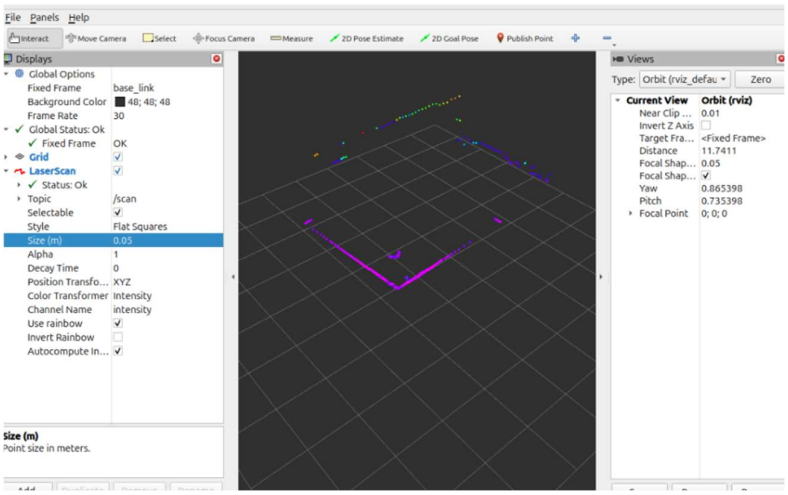


Figure 32. RVIZ scan data simulation

This setup confirms that the LD06 sensor is correctly installed and publishing scan data usable for future SLAM implementation and autonomous navigation. The visual output also helps in verifying sensor alignment, scan density, and detection range.

4.4 SLAM

Simultaneous Localization and Mapping (SLAM) is a critical capability for autonomous mobile robots. It enables a robot to construct a map of an unknown environment while simultaneously tracking its position within that map—without relying on GPS or predefined landmarks. For this project, SLAM allows the inspection robot to explore, navigate, and perform data-gathering tasks in semi-structured environments such as laboratories and industrial interiors.

4.4.1 How SLAM Works

SLAM combines two complex tasks:

- Mapping: Building an accurate spatial representation of the environment.
- Localization: Estimating the robot's position and orientation in real time.

These processes are interdependent—accurate mapping requires good localization, and vice versa. SLAM systems operate using two core components:

- Front-end (Sensor Signal Processing): Converts raw sensor input (e.g., LiDAR scans) into a geometric representation of the surroundings. This step is highly dependent on the type of sensors used.

- Back-end (Pose Graph Optimization): Optimizes the robot's estimated path over time using mathematical methods like nonlinear optimization.

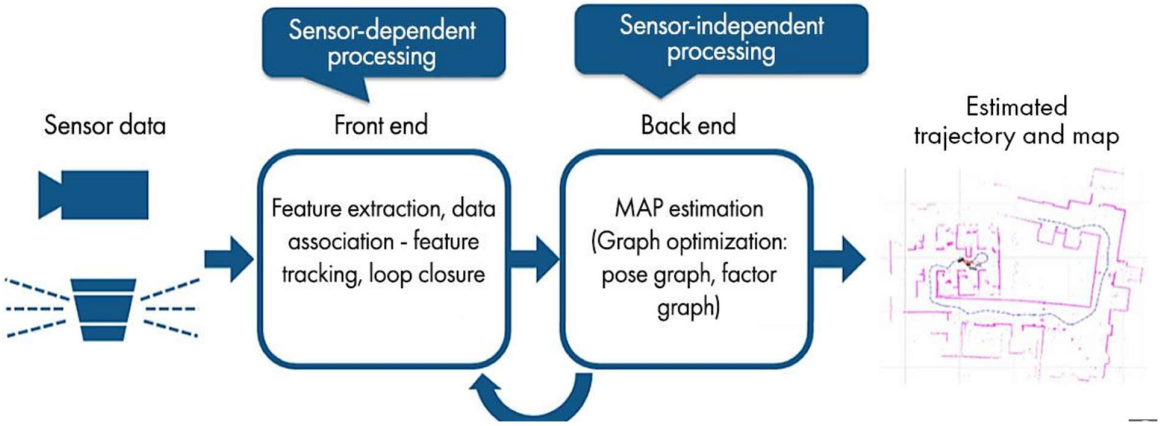


Figure 33. How SLAM works

source: https://www.mathworks.com/discovery/slam.html

By continuously integrating sensor data and motion estimates, SLAM systems can incrementally update maps while tracking the robot's trajectory.

4.4.2 SLAM Toolbox in ROS2 Jazzy

In this project, SLAM is implemented using the widely adopted slam_toolbox package in ROS2 Jazzy. It is a powerful and mature solution for 2D SLAM, originally developed by Steve Macenski, and designed for both production robotics and academic research. slam_toolbox provides a robust combination of real-time scan matching, pose-graph optimization, map serialization, and lifelong mapping capabilities, making it well-suited for autonomous mobile robots.

slam_toolbox includes the following capabilities:

- Online and offline mapping: Real-time scan-based SLAM with support for asynchronous and synchronous modes.
- Pose-graph optimization: Using solvers like Ceres, with support for loop closure, manual graph editing, and plugin-based optimizers.
- Localization mode: Efficient "lidar odometry" using a rolling buffer of recent scans. This acts as a modern replacement to AMCL.
- Lifelong mapping: Continual mapping and refinement of environments through pose-graph serialization and reloading sessions.

- Map merging and graph manipulation: Multiple serialized pose-graphs can be combined into a unified map via RViz interactive tools.
- Lossless map storage: Saves full pose graphs instead of only image files for high-fidelity remapping.

SLAM Architecture

The slam_toolbox node operates as follows:

Data Input:

- Subscribes to /scan (from the LD06 LiDAR)
- Optionally uses /odom (wheel encoder data)

Pose Estimation:

- Each laser scan is processed into a PosedScan object, combining LiDAR data and odometry to estimate robot pose.

Pose Graph Formation:

- These PosedScan nodes are added to a pose graph.
- Scan matching (correlative scan matcher) is used to refine odometry.

Loop Closure Detection:

- When re-visiting an area, a loop closure constraint is added.
- The graph is then globally optimized to correct drift.

Map Generation:

- A 2D occupancy grid map is built using all scan nodes after pose optimization.

4.4.3 SLAM on RVIZ

RViz (ROS Visualization) is a powerful visualization tool for ROS that enables developers to view data from sensors, robot models, maps, and transforms in real-time. In this project, RViz was used to visualize SLAM mapping output generated by the slam_toolbox package using the LD06 LiDAR sensor.

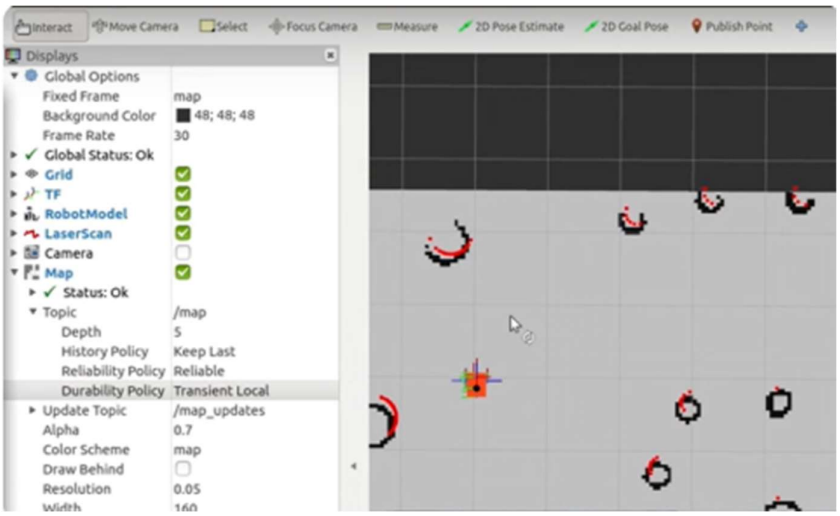


Figure 34.SLAM RVIZ

Map Construction

- The map shown in RViz is generated from the accumulated laser scan data.
- The gray area represents free space, black pixels indicate obstacles, and red arcs highlight real-time scan overlays.
- As the robot moves, each scan is added to the map, and loop closure detection refines the map via pose-graph optimization.

RViz provided an effective and intuitive way to monitor the progress of SLAM in real time. It helped in verifying the integrity of scan data, map quality, and transform alignment during testing and debugging. The combined use of LaserScan and Map topics ensured complete visualization of both local sensor feedback and global mapping performance.

4.5 DC motor Motion coding / testing

```
volatile long encoderCount = 0;
float distancePerPulse = 32.04 / 990.0;  // cm
long lastCount = 0;
void setup() {
  pinMode(2, INPUT_PULLUP); // Encoder A
  pinMode(3, INPUT_PULLUP); // Encoder B
  attachInterrupt(digitalPinToInterrupt(2), updateEncoder, CHANGE);
  Serial.begin(9600);
  Serial.println("Encoder Test Started");
}
void loop() {
  long currentCount = encoderCount;
  long delta = currentCount - lastCount;
  float distance = currentCount * distancePerPulse;
  Serial.print("Encoder Count: ");
  Serial.print(currentCount);
  Serial.print(" | Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  lastCount = currentCount;
  delay(200);
}
void updateEncoder() {
  int a = digitalRead(2);
  int b = digitalRead(3);
  if (a == b) {
    encoderCount++;
  } else {
    encoderCount--;}}

```

Code: 1 Encoder testing code

5 Results and Discussion

5.1 Mechanical Results and Observations

The robot's mechanical structure was successfully assembled using custom 3D-printed PLA components. As shown in Figure 35 and Figure 36 , all articulated joints, motor mounts, and chassis structures were fitted together without the need for extensive post-processing. The modular design ensured that each component was independently accessible for testing and maintenance.

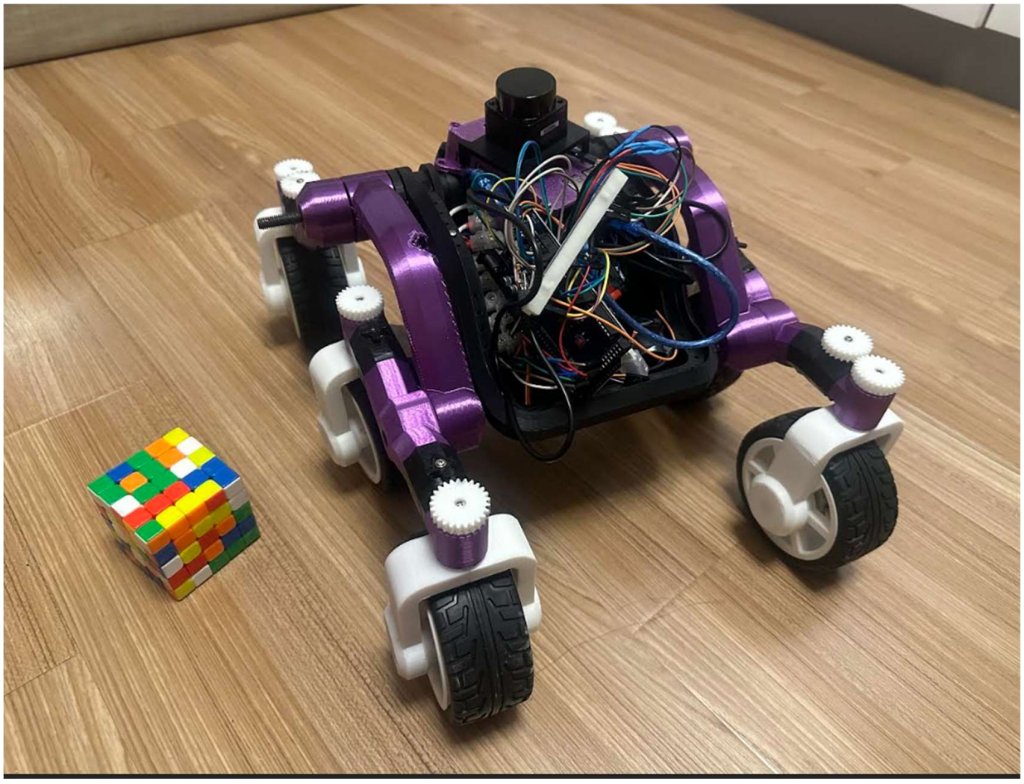


Figure 35. Final project robot

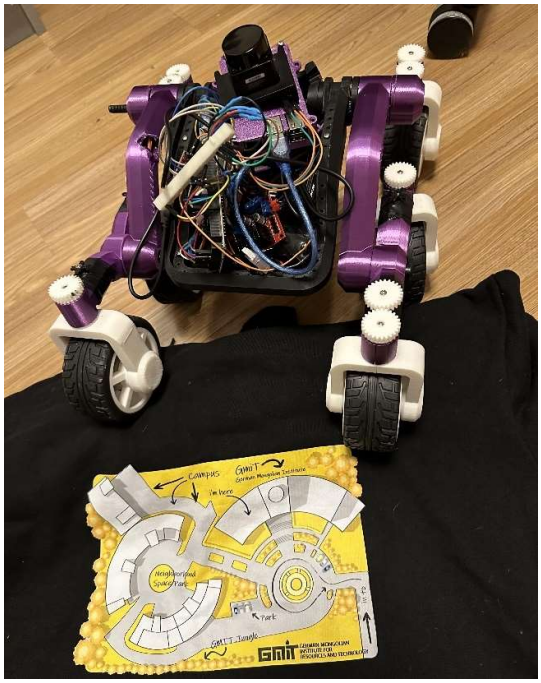


Figure 36. Final Robot v2

However, several practical issues emerged during physical integration:

Component Fitment and Space Constraints: Although initial CAD planning considered most component sizes, actual integration revealed that certain electronic components did not fit well within the printed frame. The tight internal spacing made routing wires difficult and limited airflow around key heat-sensitive modules.

Weight Deviation: The final assembly weighed more than the estimated 7.05 kg. This discrepancy primarily came from underestimating the combined mass of wires, bolts, battery, and connectors. As a result, the originally selected 6V DC motors struggled to generate sufficient torque on outdoor terrain. While they performed adequately on smooth indoor floors, their performance dropped significantly when tested on rougher ground, highlighting the need for stronger motors or upgrading to 12V operation for future iterations.

Cable Management Issues: As seen in the figure, cable routing remained a major challenge. Due to limited space and complexity, cables often became tangled or dislodged, especially when

the robot moved suddenly. This led to intermittent disconnections at the Arduino and breadboard level. Although attempts were made to shorten cables, this made routing even more difficult and fragile. A long-term solution may include designing and fabricating custom PCBs or using wire sleeves and clamps.

Thermal Concerns: During extended testing, the Raspberry Pi 5 and buck converter exhibited noticeable heat buildup. Although the robot is not designed for long continuous use, the thermal load was significant enough to warrant concern. Adding heatsinks or active cooling (fans) is recommended for stability in future designs.

Servo Failure Due to Overload: The intended servo-based articulation system for stair climbing could not function properly. The servos were underpowered relative to the robot's weight, and ROS2–Arduino integration proved more complex than expected. The mechanical load frequently exceeded the servo torque capacity, causing motion to fail or jitter under stress.

5.2 Software Performance

On the software side, the integration between Arduino and Raspberry Pi 5 using serial communication worked reliably for basic motor control. This connection enabled the Pi to act as the main processing unit while offloading low-level motor commands to the Arduino, providing a functional bridge between hardware and ROS2-based software control.

However, the SLAM performance was limited when deployed on the physical robot. Due to the robot's relatively large chassis and dynamic motion behavior, the environment scanning was insufficient to maintain consistent localization. The motion control algorithms were not fully optimized for the physical characteristics of the robot, which led to drift and inconsistencies in the SLAM map.

Further, the setup and configuration of the ROS2 environment presented a steep learning curve. Many libraries and dependencies—such as those for slam_toolbox and serial communication—had undergone recent updates, and official documentation often lagged behind. This made it difficult to find reliable, up-to-date references. Even small differences in ROS2 versions or missing dependencies caused significant time loss during development.

Although Nav2 (Navigation 2) was considered for advanced navigation, it could not be successfully integrated due to the limitations of the current motion control code and lack of a robust odometry source. ROS2 Nav2 requires precise velocity control, feedback loops, and consistent tf transforms—all of which proved too complex within the available timeframe.

5.3 Personal Reflection and Project Scope

The thesis project was a challenging but immensely valuable experience. Many of the required concepts—such as SLAM, robot communication, and real-time control—extended far beyond the curriculum offered at the undergraduate level. As such, a large portion of the learning was self-directed.

Over the course of development, significant time was spent troubleshooting, researching undocumented bugs, testing various ROS2 configurations, and learning to interface complex hardware components. The project required dedication not only to implementation but to understanding how each subsystem operates individually and as a whole.

Despite the incomplete performance in some areas, the project laid a solid foundation for future development. Key improvements moving forward include:

- Implementing a custom PCB or connector system for cable reliability
- Refining the control code with PID or velocity control support
- Upgrading to more powerful motors or optimizing weight distribution
- Completing Nav2 integration with reliable SLAM-based localization

The experience has underscored the complexity of real-world robotics and highlighted areas for continued learning. This project was more than just a final assignment—it was a first step toward professional robotics development.

6 Conclusion

This project has been a vital and hands-on journey in advancing my skills in mechatronics engineering, combining both hardware and software aspects. Throughout the development of this autonomous inspection robot, I encountered numerous technical and practical challenges that pushed me to research, prototype, and often fail—yet ultimately grow significantly in the process.

The core objective of the robot—to detect its surrounding environment using a LiDAR sensor, transmit that data to a Raspberry Pi, and visualize the results remotely on a laptop—was successfully achieved. The system is capable of basic autonomous movement and environmental mapping, avoiding obstacles based on real-time sensor input.

However, the project also revealed several areas that require further improvement. Issues such as delayed response, unstable motion control, and wiring complexity impacted performance. These challenges are valuable learning opportunities that inform how future iterations can be more robust, efficient, and reliable.

This thesis represents not only a technical achievement but also a vision of what is to come. As robotics continues to grow rapidly, the demand for skilled engineers to design, maintain, and improve intelligent machines will increase. I aspire to be part of this future—one of the engineers contributing to meaningful innovations that shape the next generation of autonomous systems.

References

[1] Grand View Research. (n.d.). Mining automation market size, share & trends analysis report. Retrieved May 1, 2025, from https://www.grandviewresearch.com/industry-analysis/mining-automation-market

[2] Creative Commons. (n.d.). *Attribution 4.0 International (CC BY 4.0)*. https://creativecommons.org/licenses/by/4.0/

[3] Creality. (n.d.). *Creality Ender-3 V3 Plus*. https://www.creality.com/products/creality-ender-3-v3-plus

[4] Gupta, M. (2023, August 9). *Testing cheap Lidar LD06*. Medium. https://medium.com/@cyber_explorer/testing-cheap-lidar-ld06-275e682acce

[5] Hellomudit. (2021, May 4). *Understanding SSH workflow*. Medium. https://medium.com/@hellomudit/understanding-ssh-workflow-66a0e8d4bf65

[6] HyperPhysics. (n.d.). *Torque: Rotational analog of force*. http://hyperphysics.phy-astr.gsu.edu/hbase/torq2.html

[7] Logitech. (n.d.). *Logitech C270 HD Webcam*. https://www.logitech.com/en-us/shop/p/c270-hd-webcam

[8] MathWorks. (n.d.). *What is SLAM? Simultaneous Localization and Mapping*. https://www.mathworks.com/discovery/slam.html

[9] MOST Electronic. (n.d.). *DC Gear Motor with Magnetic Linear Encoder JGY-370 15KG 97RPM 12V Motor Bracket*. https://mostelectronic.com/shop/motors-drives-2/dc-motors-motors-drives-2/dc-gear-motor-with-magnetic-linear-encoder-jgy-370-15kg-97rpm-12v-motor-bracket/?add-to-cart=9928

[10] Orca Slicer. (n.d.). *Orca Slicer*. https://orca-slicer.com/

[11] PCA9685. (n.d.). *PCA9685 16-Channel 12-Bit PWM Servo Driver*. AZ-Delivery. https://www.az-delivery.de/en/products/pca9685-servotreiber

[12] Printables. (n.d.). *Stair-Climbing Rover*. https://www.printables.com/model/194299-stair-climbing-rover

[13] Raspberry Pi Foundation. (n.d.). *Raspberry Pi 5*. https://www.raspberrypi.com/products/raspberry-pi-5/

[14] Raspberry Pi Foundation. (n.d.). *Remote access documentation*. https://www.raspberrypi.com/documentation/computers/remote-access.html

[15] Raspberry Pi Foundation. (n.d.). *How to install Ubuntu on your Raspberry Pi*. https://ubuntu.com/tutorials/how-to-install-ubuntu-on-your-raspberry-pi#1-overview

[16] Robotics Backend. (n.d.). *Raspberry Pi Arduino Serial Communication*. https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/

[17] ROS.org. (n.d.). *Installation Guide - Jazzy*. https://docs.ros.org/en/jazzy/Installation.html

[18] Stereolabs. (n.d.). *RViz2 Integration with ZED SDK and ROS2*. https://www.stereolabs.com/docs/ros2/rviz2

[19] The PRIF. (2021). *Handbook Battery Energy Storage System*. https://www.theprif.org/sites/theprif.org/files/documents/handbook-battery-energy-storage-system.pdf

[20] Thomas, L. F. (2017). *Electronics Fundamentals: Circuits, Devices and Applications* (8th ed.). Pearson.

[21] Voltaat. (n.d.). *Geared Brushed DC Motor with Encoder 12V 130RPM*. https://www.voltaat.com/products/geared-brushed-dc-motor-with-encoder-12v-130rpm

[22] Amazon. (n.d.). *DS041MG Torque Digital Servo*. https://www.amazon.sg/DS041MG-Torque-Digital-Helicopter-Fix-Wing/dp/B0BH4RXPB2

[23] Wikipedia. (n.d.). *Buck converter*. https://en.wikipedia.org/wiki/Buck_converter

[24] ElectroPeak. (n.d.). *CP2102 USB to TTL Serial Converter*. https://electropeak.com/cp2102-ld-led-usb-ttl-serial

[25] Articulated Robotics. (n.d.). *Power Theory for Mobile Robots*. https://articulatedrobotics.xyz/tutorials/mobile-robot/hardware/power-theory

[26] Craig, J. J. (2005). Introduction to Robotics: Mechanics and Control (3rd ed.). Pearson Education.

[27] Macenski, S., Jambrecic I., “SLAM Toolbox: SLAM for the dynamic world”, Journal of Open Source Software, 6(61), 2783, 2021.

[28] Glover, J. D., Sarma, M. S., & Overbye, T. J. (2017). Power system analysis and design (6th ed.). Cengage Learning.

[29] Grainger, J. J., & Stevenson, W. D. (1994). Power system analysis. McGraw-Hill Education.

[30] El-Hawary, M. E. (2003). Principles of electric machines and power electronics (2nd ed.). Wiley-IEEE Press

[31] Voltaat. (n.d.). 2Amp 7V–30V L298N Motor Driver – Stepper Driver (2 Channels). Voltaat. Retrieved May 2, 2025, from https://www.voltaat.com/products/2amp-7v-30v-l298n-motor-driver-stepper-driver-2-channels