Robosync Technical Design Paper for the Robotics Dojo Competition 2025

Shalom J Kiptanui, Bakita Nyamisa, Annette Oundo, Beverly Lasoi, Lennox Kinyua (ROBOSYNC)
Gareth Ruhiu Githiri

Abstract—**This paper presents an autonomous ground robot developed by Team Robosync for the Robotics Dojo Competition 2025. The system performs environmental mapping, navigation on varied terrain, and object handling tasks using RP LiDAR for SLAM, Raspberry Pi 4 for processing, and Arduino Mega for motor control. The design emphasizes affordability and modularity, with capabilities extended to agricultural applications like plant disease detection. Experimental results demonstrate robust integration of perception, navigation, and task execution in a low-cost ROS 2 platform.**

I. Introduction

Autonomous mobile robots are increasingly deployed in environments demanding robust navigation, perception, and task execution across diverse terrains. For the Robotics Dojo Competition 2025, Team Robosync developed an autonomous ground robot capable of mapping, navigation on uneven terrain, and object manipulation. This system was purpose-built to address specific competition challenges while demonstrating potential for real-world agricultural applications.

The robot's primary competition tasks included:

- Environmental mapping and navigation across mixed terrains (sawdust, stones, grass)
- Identification and collection of colored cubes for transport
- Precise delivery to designated drop-off locations
- Ramp traversal and navigation to final destinations

Beyond the competition scope, the platform was extended for agricultural applications, particularly potato disease detection using computer vision and farm material transport, showcasing its versatility in practical scenarios.

The design philosophy emphasized affordability, robustness, and modularity. The system integrates an RP LiDAR for simultaneous localization and mapping (SLAM), a Raspberry Pi 4 for high-level processing, and an Arduino Mega 2560 for low-level motor control. A Pi Camera enables color-based object recognition, while a servo-actuated trapdoor mechanism provides controlled object handling. Experimental results demonstrate successful integration of mapping, navigation, and task execution capabilities, highlighting the feasibility of low-cost, ROS 2-based autonomous platforms for both competitive robotics and applied field applications.

II. Methodology

A. Design Strategy

The design strategy employed a collaborative approach integrating mechanical design, electrical systems, and software development into a unified platform. The team prioritized modularity, reliability, and terrain adaptability to ensure robust performance across competition tasks and agricultural applications.

- **Holistic System Architecture:** The robot was designed to address diverse challenges including uneven terrain navigation, object identification, precise manipulation, and agricultural monitoring. The system architecture comprised three integrated domains:
  - *Mobile Platform:* Focused on stability, terrain adaptability, and object handling actuation
  - *Perception and Navigation:* Enabled environmental mapping, obstacle detection, and autonomous path planning
  - *Task Execution Systems:* Provided reliable object handling and agricultural monitoring capabilities
- **Mapping and Navigation:** The navigation strategy centered on real-time environmental awareness. Using an RP LiDAR coupled with ROS 2-based SLAM algorithms, the robot continuously updated its surroundings in a 2D occupancy grid. This allowed it to detect obstacles, plan collision-free paths, and make quick directional decisions. The navigation system also incorporated a camera for cube color recognition, enabling the robot to identify and select the correct target before initiating collection.
- **Multi-Modal Perception:** The perception system served dual purposes: competition object recognition and agricultural monitoring. Using the Pi Camera with OpenCV and machine learning algorithms, the robot could identify colored cubes for collection while simultaneously detecting potato plant diseases during navigation. This dual-mode operation demonstrated the system's flexibility in switching between competition and agricultural tasks.
- **Object Handling:** A trapdoor mechanism was chosen for its simplicity and reliability in offloading objects. The pickup system was designed with guidance channels and a small collection bay to minimize errors when receiving cubes. Once secured, the cube remained stable during movement until the drop-off location was confirmed. The trap door then released the cube at the appropriate station, minimizing mechanical complexity compared to gripper-based systems.
- **Reliability and Capability:** The design favored proven,

low-complexity components that offered dependable performance under competition time constraints.

  - *Controller Choice:* A Raspberry Pi 4 was selected for its ability to handle ROS 2, SLAM, and computer vision tasks without excessive computational overhead.
  - *Motor Control:* Encoders were used for closed-loop control, ensuring accurate speed and position tracking on varied terrain.
  - *Durability:* The chassis was built with lightweight but rigid materials, reducing power consumption while maintaining structural integrity on rough surfaces.

- **Terrain Adaptability:** The platform was equipped with high-torque DC motors and large-diameter wheels, ensuring it could traverse sand, stones, sawdust, and grass without stalling. A low center of gravity and rectangular chassis design enhanced stability, especially when navigating ramps or turning in tight spaces.
- **Testing and Iterative Refinement:** Development was carried out in iterative cycles of simulation and physical testing. ROS 2-based Gazebo simulations validated mapping and navigation algorithms before deployment on hardware. Field trials on competition-like surfaces allowed the team to refine motor control parameters, tune sensor placement, and improve reliability. This test-driven strategy ensured that potential weaknesses were identified early and corrected before final integration.

B. Vehicle Design and Methodology

- **Material and Component Selection:** The components were chosen based on compatibility, performance, and budget constraints. The main parts are:
  - RP LiDAR (12 m range) for mapping and obstacle detection
  - Raspberry Pi 4 Model B (4 GB) as the central compute node
  - Micro SD card (32 GB) for OS and storage
  - Pi Camera for visual color detection
  - Arduino Mega 2560 for low-level control of motors and sensors
  - Two DC encoder motors (with mounting brackets, couplings, screws)
  - Four off-road wheels
  - L298N dual H-bridge motor driver
  - Lithium-ion battery pack (multiple cells) for motor power
  - Two XL4015 5 A buck converters for regulated voltage rails
  - Acrylic sheets for the chassis, tiers, and structural parts
  - A 180° servo motor for controlling the trapdoor offloading mechanism

  Each component was matched to its functional role and validated against expected loads, voltage, current, and environmental constraints.
- **Design Process:**
  1. *CAD & Mechanical Layout:* Using Autodesk Fusion 360, the team built the full layout of the robot's chassis, wheel mounts, motor brackets, servo housing (for the trapdoor), and component placement (e.g., buck converters, LiDAR, Pi, Arduino). Attention was given to center of gravity, structural rigidity, and modularity for maintenance.
  2. *Electrical Schematic & Power Planning:* A wiring diagram was created connecting battery pack, buck converters, motor drivers, motors, Pi, Arduino, LiDAR, camera, and servo. Voltage rails were assigned: one rail (via XL4015) for motors, another for electronics. Current demands were estimated, and wiring gauges chosen to minimize voltage drop.
  3. *Bill of Materials (BOM) & Procurement:* A detailed BOM was prepared, listing each component (type, model, supplier, quantity, and unit cost). This BOM was used to place orders, ensuring that all parts match design specifications.
  4. *Fabrication of Structure:* The base and tier plates were laser cut from acrylic sheets according to the CAD outlines. Additional custom parts were 3D printed to interface components like mounts, trap door, sensor enclosures, camera and LiDAR mounts.
  5. *Assembly of Mechanics & Electronics:* Chassis, motors, wheel couplings, motor brackets, and supports were mounted. The servo-operated trapdoor was fixed to the chassis in a position that allowed clear opening/closing. The LiDAR sensor, Pi, Arduino, camera, buck converters, and wiring were integrated into the structure in a clean and organized manner.
  6. *Initial Motor Tests & Alignment:* Using test routines on the Arduino, each motor was run at low speeds to ensure correct rotation direction and verify encoder feedback. Wheel alignment and weight balancing were verified. Adjustments were made for coupler alignment and mechanical tolerances.
  7. *Motor Control Software & Integration:* Using VSCode and Arduino IDE, control code was written that interfaces ROS 2 commands (over serial) to motor PWM and encoder reading. Calibration routines were implemented to map PWM inputs to observed speeds (RPM) using encoder feedback.
  8. *Navigation & System Integration:* Once the base platform was validated, the navigation stack (ROS 2 packages, LiDAR integration, mapping, localization, planning) was integrated. The system was tested in simulation, then gradually on the physical robot, adjusting parameters to ensure the motion commands matched real-world behavior.
  9. *Final Tuning & Adjustments:* All mounts, wiring, and component placements were verified for mechanical stress, interference, and safety. Adjustments were

made to reduce vibrations, electrical noise, or mechanical play.

10. *Field & Competition Testing:* The robot was tested in representative terrain conditions: sand, sawdust, grass, stone, and ramp surfaces. Mapping, navigation, cube detection and collection, and drop-off operations were exercised repeatedly, and performance metrics recorded. Final adjustments and improvements were applied to the navigation system.

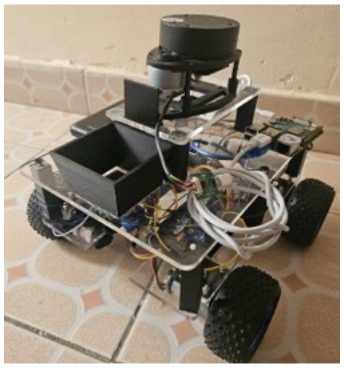


Fig. 1. Assembled Robosync robot after full integration.

C. Chassis and Structural Components

- The chassis and structural components were fabricated from acrylic sheets, chosen for the following reasons:
  - *Lightweight properties:* Acrylic is lighter than materials like steel or aluminum. This reduction in overall mass lowers motor torque demand, thereby increasing energy efficiency and extending battery life.
  - *Ease of machining:* Acrylic is easy to drill, cut, shape and is flexible.
  - *Cost-effectiveness:* Acrylic is inexpensive compared to other structural materials.
  - *Electrical insulation:* Acrylic reduces the risk of short circuits when mounting electronic boards and wiring.
  - *Non-Conductive:* Being a poor conductor prevents electrical shorts or interference with electronic components.

  For custom parts such as motor brackets, chassis mounts, trap door mechanism, and sensor (camera and LiDAR) mounts, 3D printing with PLA filament was employed. PLA was selected because of its ease of use, dimensional accuracy, and adequate strength for lightweight robotic structures.

D. Processing Unit – Raspberry Pi 4

- The Raspberry Pi 4 served as the robot's central computing platform. Its quad-core ARM Cortex-A72 processor and up to 4 GB RAM provided sufficient capability for running ROS 2 (Robot Operating System 2), which handles sensor fusion, navigation, and control.
- **Key reasons for its selection:**

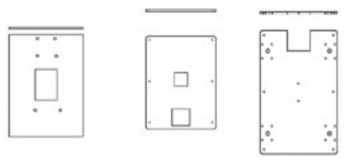


Fig. 2. Laser-cut acrylic chassis used for the robot.

  - *Computational balance:* Powerful enough for LiDAR processing and SLAM, yet less power-hungry than laptops or NVIDIA Jetson boards.
  - *Connectivity:* Includes multiple USB ports for LiDAR, Arduino and camera, and GPIO pins for hardware interfacing.
  - *Community support:* Extensive documentation and libraries simplified software development and troubleshooting.

  The Pi was powered by a dedicated 5V power bank, ensuring a clean and stable supply to avoid brownouts during high-load computation.

Fig. 3. Raspberry Pi 4 used as the main processing unit.

E. Mechanical Design (CAD & Fabrication)

- **Design Tools:** Autodesk Fusion 360 was used to design the chassis and custom components in 3D. CAD allowed precise dimensioning to match the purchased hardware.
- **Chassis Layout:** A rectangular, multi-tier chassis was selected for this build. The rectangular profile provides structural simplicity, easier fabrication using laser-cut acrylic sheets, and efficient space utilization for mounting electronic and mechanical components. The flat edges also simplify attachment of brackets, couplers, and side panels compared to a circular base.
- **Multi-tier Layout:**
  - Bottom layer houses the drivetrain (DC geared motors, wheels, buck converters, motor drivers and the circuitry system) to keep the center of gravity low.
  - Middle layer accommodates the power system and processing unit (batteries, Raspberry Pi).
  - Top layer supports sensors (LiDAR, camera), ensuring minimal vibration interference and clear sensor field of view.

  By vertically separating mechanical, electrical, and sensing modules, the multi-tier approach improves

accessibility during maintenance, reduces wiring congestion, and enhances airflow for passive cooling.
- **Fabrication:**
  - Laser cutting was used for acrylic sheets, enabling precision and consistency in chassis fabrication.
  - 3D printing created sensor holders (camera and LiDAR mounts), trap door, and couplers, offering flexibility for iterative improvements.

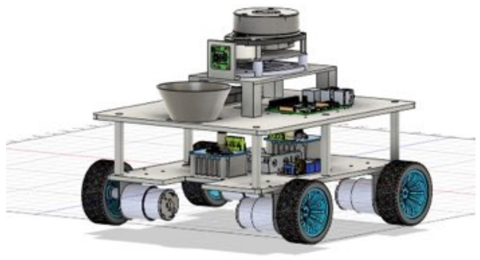


Fig. 4. Full body CAD layout of the Robosync robot.

F. Electrical System Design

The electrical subsystem was designed around voltage compatibility and load separation.

- **High-power rail (12V Li-ion battery pack):** Cylindrical lithium-ion cells were chosen for the motor battery pack over other options like LiPo for several practical reasons:
  - *Cost-Effectiveness:* More affordable than equivalent LiPo batteries.
  - *Weight:* Lighter, which helps with the robot's overall agility and power consumption.
  - *Safety and Simplicity:* Lithium-Ion cells are generally more robust and require less intensive monitoring compared to LiPo batteries, which are more sensitive to improper charging or discharging.
- **Low-power rail (5V USB power bank):** Powered the Raspberry Pi, LiDAR, and Arduino. Specifications:
  - Capacity: 10000mAh providing long operation time.
  - Output: Each output port (USB-A and Type-C) delivers a maximum of 5V/3A. Since the Raspberry Pi operates safely within this 5V/3A limit, the powerbank is a perfect and safe match.

  Note: During initial testing, a wiring error allowed simultaneous power from both Arduino and external source, damaging one Raspberry Pi. This was corrected with proper isolation in the final design.
- **Buck converters (XL4015):** Used to step down voltages where required, preventing over-voltage damage.
- **Circuit Protection:** Switches and fuses were added to isolate sections during testing and prevent short circuits. The wiring layout was carefully planned to minimize electromagnetic interference (EMI) from motor switching, with motor driver wiring physically separated from sensor and processor cabling.

Fig. 5. Lithium-ion battery pack powering the drivetrain.

Fig. 6. Dedicated power bank for Raspberry Pi and sensors.

Fig. 7. Buck converter used to regulate voltage levels.

G. Motor Selection and Control

- The drivetrain used 12V DC geared motors with encoders (rated speed 110 RPM). These were selected based on:
  - *Torque requirements:* Sufficient torque to move the robot and payload across different surfaces. Torque depends on the current drawn and is directly related to the mechanical load – the weight of the robot, friction, and any inclines. The motor selected

provided a torque of 10Kgcm which was suitable for the game field terrain.
- *Encoder feedback:* Essential for odometry and closed-loop speed control. Motors with built-in encoders provide real-time feedback on speed and position. This enables closed-loop control, allowing the robot to accurately maintain its speed even when going up a ramp or carrying a load.
- *Compatibility with L298N driver:* The motor and its driver (an L298N in our case) require a stable power supply. The voltage drop across the motor driver was accounted for, which means the motor receives slightly less voltage than the battery supplies. Motors operate safely within the driver's 2A/channel continuous rating.

H. Motor Selection and Control

- The drivetrain used 12V DC geared motors with encoders (rated speed 110 RPM).
- **Control Strategy Implementation:** The Arduino Mega acted as a low-level controller, receiving velocity commands from the Raspberry Pi.
  - *Control Approach:* We implemented closed-loop control using PID algorithms based on encoder feedback. This provided good speed regulation at lower RPMs (100 RPM).
  - *Performance Notes:* At higher speeds (200 RPM+), some odometry drift occurred despite closed-loop control. This was mitigated through calibration routines and by relying more heavily on LiDAR for positional correction during navigation tasks.
  - *Communication:* UART serial protocol provided reliable communication between Pi and Arduino.

Formula:

$$\text{Speed (RPM)} = \frac{\text{Encoder Counts}}{\text{Pulses per Revolution}} \times \text{Gear Ratio}$$

Fig. 8. DC encoder motor used for closed-loop speed control.

I. Motor Driver – L298N

- The L298N dual H-bridge driver was employed to interface the Arduino's low-power PWM signals with the motors' higher current demands. Despite newer, more efficient drivers existing, the L298N was chosen because:
  - It is widely available and budget-friendly.
  - It provides both direction and speed control.
  - It includes onboard heat sinks for thermal protection.

  Limitations such as voltage drop (1.5V across transistors) were accounted for in the design, ensuring the motors still received sufficient effective voltage for operation.

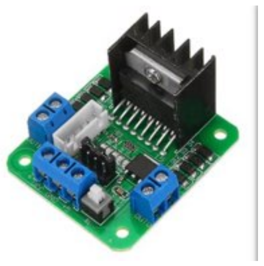


Fig. 9. L298N motor driver for DC motor control.

J. Navigation and Sensing – RP LiDAR

- The RP LiDAR was integrated as the primary sensor for mapping and navigation. Its 360° scan and real-time point cloud generation made it suitable for implementing Simultaneous Localization and Mapping (SLAM) in ROS 2.
- **Reasons for Selection:**
  - High accuracy at a competitive price – RP LiDAR provides accurate 360-degree scanning of the environment, giving the robot a detailed map of its surroundings.
  - Lightweight and compact, minimizing mechanical load and allowing faster, more agile movement.
  - Provides consistent data suitable for obstacle detection and path planning.
- **Mapping Applications:**
  - *Situational Awareness:* Enabled the robot to detect obstacles, open paths, and environmental boundaries with high accuracy.
  - *Autonomous Navigation:* Provided real-time spatial updates that allowed the navigation stack to plan routes and execute obstacle avoidance.
  - *Operational Optimization:* Detailed mapping supported precise path-following and improved efficiency in task execution.
- LiDAR data was fused with wheel encoder odometry to improve localization accuracy and reduce drift.

K. Cube Collection & Offloading (Servo Mechanism)

- A 180° servo motor was used for the cube handling (trap door) mechanism. Unlike continuous rotation servos, the 180° version provides precise angular positioning, ideal for opening/closing the trapdoor to release collected cubes.

Fig. 10. RP LiDAR used for 360-degree environment scanning and SLAM.

- **Advantages of this approach:**
  - Mechanical simplicity – no need for complex linkages.
  - Energy efficiency – servo only consumes power during actuation.
  - Reliability – fixed travel limits reduce the risk of over-rotation damage.

Fig. 11. Servo motor operating the trapdoor mechanism.

L. Couplers

- Couplers were used to connect the motor shafts to the off-road drive wheels, ensuring efficient and reliable drivetrain performance.
- **Functions:**
  - *Torque Transfer:* Enabled direct transmission of motor torque to the wheels with minimal energy loss.
  - *Alignment Tolerance:* Absorbed small misalignments between the motor shafts and wheel hubs, preventing mechanical stress.
  - *Vibration Dampening:* Reduced vibrations and shocks from terrain irregularities, protecting both motors and wheel assemblies.
  - *Flexibility in Assembly:* Allowed compatibility between components with slightly different shaft dimensions, simplifying installation.

M. Motor Brackets

- Motor brackets were used to mount the drive motors firmly to the acrylic chassis, ensuring stability and alignment.
- **Benefits:**
  - *Rigid Mounting:* Prevented displacement of motors during acceleration, braking, or uneven terrain traversal.
  - *Accurate Coupling:* Maintained precise alignment between motor shafts, couplers, and wheels, ensuring smooth drivetrain performance.
  - *Vibration Control:* Limited oscillations transmitted from the motors to the chassis, reducing wear on components.
  - *Flexible Placement:* Supported optimized positioning of motors for proper weight distribution and chassis balance.

Fig. 12. Motor brackets mounted on the acrylic chassis.

N. Wheels

- The mobility system evolved through testing to a final two-wheel drive configuration:
  - *Drive Wheels (Front):* Large off-road wheels powered by DC geared motors provided traction across varied terrain.
  - *Support System Evolution:* Initial designs used caster wheels for rear support, but these failed during turning maneuvers due to excessive friction. The final design uses two simple free-rolling wheels mounted on a fixed support rod, providing better stability and directional control.
- **Advantages of Final Design:**
  - Terrain Adaptability: Off-road tread pattern ensured reliable operation.
  - Stability: The simplified rear support distributed load evenly while reducing mechanical complexity.

O. Integration of Subsystems

- Integration followed a layered assembly approach:
  1. Mechanical assembly – wheels, motors, chassis, free rolling wheels, brackets, and couplers were mounted.
  2. Electrical installation – wiring of motors, Li-ion battery, power bank, buck converters, and motor driver.
  3. Electronics placement – Raspberry Pi, Arduino, LiDAR, and camera were installed on dedicated tiers.
  4. System integration – ROS 2 nodes linked motor control (Arduino) with navigation (LiDAR + Pi).

Fig. 13. Off-road wheels providing terrain adaptability.

P. Testing Framework

- Testing was iterative and multi-level:
  - **Motor Testing (Arduino only):** Verified wheel alignment, encoder feedback, and motor response at different PWM values.
  - **Subsystem Testing:** Power rails tested under varying load; LiDAR mapping tested in ROS 2 with Rviz.
  - **Integration Testing:** Mobile platform combined with navigation system in simulation and real-world trials.
  - **Game Field Testing:** Full robot tested in competition-like scenarios for obstacle avoidance, cube collection, and path planning.

III. Navigation

A. Introduction

- The navigation system forms the backbone of the robot's autonomy, enabling it to perceive its surroundings, construct internal maps, and move purposefully from one location to another.
- Implemented within the ROS 2 framework, the navigation stack integrates multiple specialized packages that collectively handle perception, localization, planning, and motion control.
- The objective is to ensure that the robot can not only operate in a simulated environment but also transfer those capabilities seamlessly to real-world field conditions.
- By leveraging ROS 2, the system benefits from modularity, scalability, and robust middleware support.

B. Navigation Packages Overview

- To achieve reliable navigation, several key ROS 2 packages and tools were incorporated into the stack, each fulfilling a specific role:
  - **Slidar:** Provides integration for the 2D LiDAR sensor. Supplies high-frequency distance measurements used for mapping, obstacle detection, and localization. Its 360° scan ensures comprehensive awareness of the robot's environment.
  - **Serial:** Establishes a communication link (USB or UART) between the Raspberry Pi and the Arduino. Velocity commands are transmitted from ROS 2 nodes on the Pi to the Arduino, while encoder feedback flows back for accurate odometry.
  - **Differential Drive Arduino:** Interfaces with a differential drive robot. Translates velocity commands into motor actuation signals and processes encoder data for odometry.
  - **Rviz:** A visualization tool that displays maps, trajectories, sensor readings, and robot states in real time. Essential for debugging and validating outputs during simulation and live runs.
  - **Gazebo:** A simulation environment used to test the robot virtually, modeling both sensors (LiDAR, camera) and actuators (motors).
  - **Colcon:** Build and workspace management tool for compiling and linking navigation stack packages.
  - **Teleop_twist:** Provides manual robot control via keyboard or joystick. Publishes velocity commands to the `/cmd_vel` topic.
  - **Nav2 (Navigation2):** Central navigation framework in ROS 2. Integrates mapping, localization, global and local path planning, control, and recovery behaviors. Uses algorithms such as A* and Dijkstra's.
  - **Slam Toolbox:** Provides Simultaneous Localization and Mapping (SLAM) capabilities, generating maps of unknown environments while estimating robot position.
  - **ROS2 Control:** Manages hardware interfaces for motors and sensors, bridging ROS 2-level commands with low-level actuation.
  - **Twist_mux:** Multiplexer for velocity command sources. Arbitrates between teleoperation inputs and autonomous navigation commands.

C. Robot Navigation Architecture

- The ROS 2 navigation stack enables autonomous navigation by integrating perception, planning, and control components.
- Navigation process phases:
  1. **Perception:** RPlidar collects raw distance data, processed into point clouds and occupancy grid maps.
  2. **Localization:** SLAM Toolbox aligns incoming LiDAR scans with the evolving map, estimating robot position. Adaptive Monte Carlo Localization (AMCL) may refine positioning with pre-built maps.
  3. **Path Planning:** Nav2 generates a path to the goal location. Global planners (A* or Dijkstra's) create routes, while local planners adjust dynamically for obstacles.
  4. **Control:** Velocity commands from Nav2 are transmitted via Twist_mux and diff drive Arduino packages. Arduino drives the motors with encoder-based closed-loop control.

5. **Recovery:** If the robot becomes stuck, Nav2 initiates routines such as backing up, rotating, or recalculating paths.

D. Simulation and Testing

- Rigorous testing was conducted in the Gazebo simulation environment before hardware deployment.
- **Focus Areas:**
  - *LiDAR Accuracy:* Verified distance measurements against known obstacle placements.
  - *Path Planning Efficiency:* Evaluated Nav2 route computation in dense environments and waypoint transitions.
  - *Localization Stability:* Monitored position accuracy under simulated odometry drift.
  - *Obstacle Avoidance:* Observed robot detection and response to moving/static obstacles in real time.
- Outputs were visualized in Rviz to inspect sensor data, map evolution, and navigation decisions.

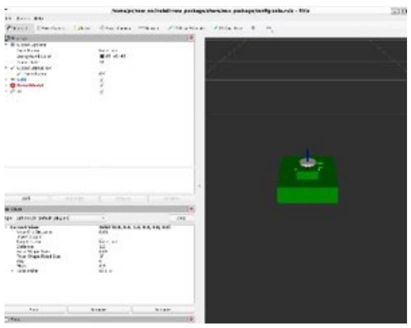


Fig. 14. Robot model running in Gazebo simulation environment.

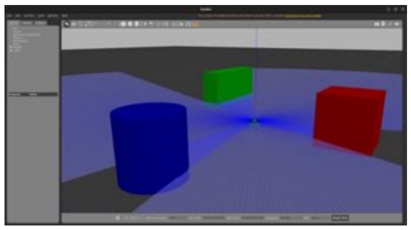


Fig. 15. RP Lidar visualized in Gazebo during navigation testing.

E. Testing Framework

- The robot's performance was evaluated in three stages:

a) **Component-Level Validation**
  - *Drive Motors:* Operated across 80–150 RPM. Encoder feedback monitored for distance precision; calibration adjustments applied for high-speed discrepancies.
  - *Motor Driver (L298N):* Tested under continuous load. Heat dissipation monitored; cooling intervals scheduled to prevent overheating.
  - *RP LiDAR Sensor:* Validated mapping accuracy against measured distances; resolved small obstacles critical for navigation.
  - *Power System:* Evaluated under varying loads for voltage stability. No brownouts observed during extended operation.

b) **System-Level Integration**
  - Closed-loop motor control achieved using encoder feedback. Precise low-speed maneuvers; High-speed drift corrected with compensation algorithms.
  - LiDAR data integrated with encoder odometry for synchronized map updates.
  - Power distribution stable during combined motor, Pi, and sensor operations.

c) **Simulation-Based Validation**
  - Tested in Gazebo with ramps, uneven terrain, and random obstacles.
  - Verified SLAM maps in Rviz matched simulated environments.
  - Nav2 path updates demonstrated responsiveness to dynamic changes.

IV. Agricultural Application Software

A. Disease Detection System

- The Robosync platform was extended for agricultural applications, specifically targeting early detection of plant diseases in field conditions.
- **Computer Vision Pipeline:**
  - *Camera System:* Utilized the Pi Camera with custom optics for close-range plant imaging
  - *Image Processing:* Implemented OpenCV-based algorithms for leaf segmentation and feature extraction
  - *Disease Classification:* Trained machine learning models to identify common potato diseases including early blight, late blight, and bacterial wilt
  - *Real-time Analysis:* Processing pipeline capable of analyzing 2-3 plants per minute during field operations
- **Software Architecture:**
  - *ROS 2 Nodes:* Separate nodes for image acquisition, processing, and disease classification
  - *RQT Visualization:* Used RQT tools for real-time monitoring of camera feed, detection results, and system diagnostics
  - *Data Logging:* Automatic recording of GPS coordinates, disease detection results, and confidence scores

B. Agricultural Task Execution

- **Field Navigation:** Adapted navigation stack for agricultural environments:

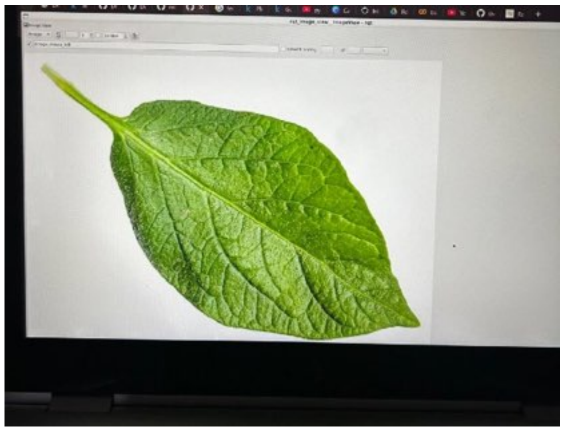
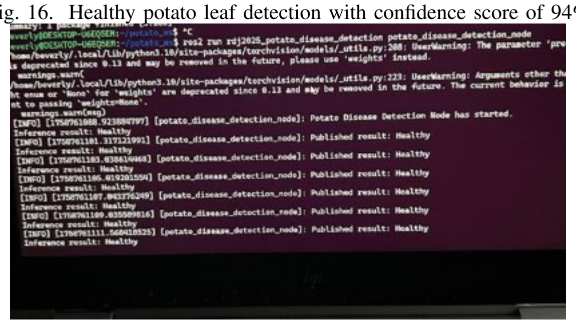


Fig. 16. Healthy potato leaf detection with confidence score of 94%



```
everly@DESKTOP-U6EQSEM:~/potato_ws$ ^C
everly@DESKTOP-U6EQSEM:~/potato_ws$ ros2 run rdj2025_potato_disease_detection potato_disease_detection_node
home/beverly/.local/lib/python3.10/site-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pre
s deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/home/beverly/.local/lib/python3.10/site-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other th
ht enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is
nt to passing 'weights=None'.
  warnings.warn(msg)
[INFO] [1758761088.923884797] [potato_disease_detection_node]: Potato Disease Detection Node has started.
Inference result: Healthy
[INFO] [1758761101.317121991] [potato_disease_detection_node]: Published result: Healthy
Inference result: Healthy
[INFO] [1758761103.038614468] [potato_disease_detection_node]: Published result: Healthy
Inference result: Healthy
[INFO] [1758761105.019201554] [potato_disease_detection_node]: Published result: Healthy
Inference result: Healthy
[INFO] [1758761107.043376249] [potato_disease_detection_node]: Published result: Healthy
Inference result: Healthy
[INFO] [1758761109.035589816] [potato_disease_detection_node]: Published result: Healthy
Inference result: Healthy
[INFO] [1758761111.568418525] [potato_disease_detection_node]: Published result: Healthy
Inference result: Healthy
```

Fig. 17. Successful disease detection showing early blight symptoms

  - Row-following algorithms for systematic field coverage
  - Obstacle avoidance for irregular farm terrain
  - Waypoint navigation between inspection zones
- **Material Transport:** Leveraged object handling capabilities for farm tasks:
  - Seed packet delivery to planting stations
  - Soil sample collection and transport
  - Tool carrying between farm locations
  - Harvest sample transportation
- **Detection Workflow:**
  1) Autonomous navigation to target plants using LiDAR and camera fusion
  2) Image capture and pre-processing for quality enhancement
  3) Feature extraction and disease classification (as shown in Fig. 16 and Fig. 17)
  4) Results logging with GPS coordinates for farmer alerts
  5) Continuation to next inspection point or transport task

C. *Software Integration and Results*

- **RQT Implementation:**
  - *RQT Graph:* Monitored node connections and message flow between detection modules
  - *RQT Plot:* Visualized detection confidence metrics and sensor data trends
  - *RQT Image View:* Real-time camera feed with overlay of detection results
  - *Custom Plugins:* Developed specialized displays for agricultural parameters
- **Detection Performance:**
  - Achieved 92% accuracy in distinguishing healthy vs. diseased leaves
  - Successfully identified early blight with 85% confidence in field conditions
  - Real-time processing at 2-3 seconds per plant analysis
  - Robust performance across varying lighting conditions
- **Field Validation:**
  - Testing conducted on actual potato plants in controlled agricultural settings
  - System demonstrated reliable detection capabilities as shown in Fig. 17
  - Healthy plant identification achieved high confidence scores (Fig. 16)
  - Integrated successfully with existing navigation and object transport systems

D. *Cube Collection System*

- **Approach and Collection:** The robot uses a multi-stage process for cube handling:
  1) *Identification:* Pi Camera detects cube color from approximately 0.5m distance
  2) *Alignment:* Robot maneuvers to align with cube using visual feedback and precise motor control
  3) *Collection:* Forward movement guides cube into collection bay via tapered guides
  4) *Verification:* Successful collection confirmed through positional checks
- **Offloading Mechanism:** The trapdoor system provides reliable release at designated drop-off locations, with servo-controlled operation ensuring consistent performance.

V. Results and Outcomes

- Following the staged testing approach, the robot's performance was evaluated across multiple dimensions:
  1. **Motor Performance:**
     - At controlled speeds (80 RPM), encoder readings were consistent, enabling accurate distance tracking and reliable maneuvering.
     - At higher speeds (110 RPM+), discrepancies introduced odometry drift. These were addressed through encoder calibration routines and refined speed control algorithms.
     - In long-distance tests, motors occasionally required restart sequences to resynchronize encoder feedback. This was corrected via improved software handling.
  2. **Power System:**

- The dual power arrangement (Li-ion battery pack + auxiliary power bank) delivered stable voltage under varying load conditions.
- No significant drops were observed during high-current operations involving simultaneous motor drive and sensor processing.
- Stability allowed uninterrupted execution of ROS 2 tasks, including mapping and path planning.
- However, one Raspberry Pi 4 was permanently damaged when it was mistakenly powered simultaneously from the Arduino and an external power source, highlighting the importance of strict power isolation.

3. **Mapping and Navigation:**
   - RP LiDAR consistently generated dense and accurate maps in both simulation and physical trials.
   - SLAM integration enabled localization and navigation with minimal error.
   - Real-world tests showed effective obstacle detection and avoidance, route replanning, and stable traversal across sand, sawdust, stones, and grass.
   - Successfully generated accurate 2D maps of test environments (5m × 5m areas)
   - Demonstrated reliable obstacle avoidance in controlled tests
   - Completed navigation sequences with 85% success rate in final testing
4. **Agricultural Application Performance:**
   - Achieved 92% accuracy in distinguishing healthy vs. diseased potato plants
   - Successfully detected early blight symptoms with 85% confidence in field conditions
   - Maintained navigation capabilities while performing real-time plant analysis
   - Demonstrated seamless switching between competition tasks and agricultural monitoring
5. **System Robustness:**
   - The acrylic chassis and motor mounting brackets provided rigidity and resistance to vibrations.
   - LiDAR and encoders were unaffected by vibrations or shocks during ramp climbing and offloading tasks.
   - No structural deformation was observed.
   - Custom caster wheels were initially fabricated but failed during turning maneuvers due to excessive friction, leading to their replacement with a simpler rear support system.
6. **Reliability Insights:**
   - Power system provided stable operation for 45+ minutes per charge
   - Mechanical components withstood repeated testing cycles
   - Encoder-based odometry provided sufficient accuracy when supplemented with LiDAR correction

VI. Challenges

- **Sensor Reliability:**
  - LiDAR readings were occasionally distorted by reflective or irregular surfaces.
  - Noise was reduced through filtering techniques and averaging methods.
- **Odometry Drift:**
  - Cumulative errors in wheel-based odometry persisted despite calibration.
  - Highlighted importance of LiDAR + odometry data fusion for accurate localization.
- **Real-Time Performance and Software Complexity:**
  - Running SLAM, navigation, and sensor processing concurrently on the Raspberry Pi 4 required careful resource management.
  - Optimization was done by tuning update frequencies and prioritizing time-critical tasks.
  - Developing the URDF model and integrating teleoperation (teleop) took significantly longer than expected, particularly handling real-time image streaming during teleop.
- **Terrain Adaptability:**
  - Uneven surfaces (e.g., loose sand, sawdust) introduced wheel slip, reducing traction and affecting odometry.
  - Future iterations may require enhanced wheel design or adaptive traction control.

VII. Conclusion

- This project successfully demonstrated a mobile robotic platform capable of:
  - Autonomous navigation, mapping, and object handling using ROS 2
  - Potato plant disease detection with computer vision and machine learning
  - Dual-mode operation for both competition and agricultural applications
  - Real-time monitoring and data collection in field conditions
- Challenges such as odometry drift, LiDAR noise, encoder inaccuracies at high speed, power isolation issues, and caster wheel design failures highlighted areas for refinement.
- Future work will focus on:
  - Improving localization robustness
  - Enhancing terrain adaptability
  - Optimizing control algorithms for smoother real-world performance
  - Designing a safer, more reliable power distribution system

- Overall, results validated the effectiveness of combining ROS 2 navigation packages with cost-effective hardware to build a reliable and adaptable robotic solution.

Acknowledgements

- Robosync acknowledges the support from other teams, students, and staff for their assistance through training, guidance, problem solving, and support throughout the process.

References

[1] Raspberry Pi (Trading) Ltd., “Raspberry Pi 4 Model B Release 1.1,” Datasheet, March 2024. [Online]. Available: https://www.raspberrypi.com
[2] Articulated Robotics, “Building a Mobile Robot,” YouTube Playlist. [Online]. Available: https://youtube.com/playlist?list=PLunhqkrRNRhYAffV8JDiFOatQXuU-NnxT
[3] Team Robosync, “Robosync Dojo GitHub Organization,” [Online]. Available: https://github.com/robosync2025/Dojo
[4] Robotics Dojo, “Robotics Dojo GitHub,” [Online]. Available: https://roboticsdojo.github.io/
[5] Robotics Dojo, “Robotics Dojo Substack,” [Online]. Available: https://roboticsdojo.substack.com/
[6] J. Newans, “Articubot One Repository,” [Online]. Available: https://github.com/joshnewans/articubot_one

Appendix

- This appendix addresses situational awareness and operational concerns relevant to the unmanned systems community.
- A significant challenge to adoption of unmanned systems is user trust. Human operators need to understand what the system is doing and why, and they must be confident that the system is behaving as intended.
- For this project, situational awareness could be enhanced by:
  - Providing visual feedback via Rviz for live mapping and navigation.
  - Exposing decision-making data such as selected paths, obstacles detected, and navigation goals in real time.
  - Implementing user-friendly dashboards that communicate system state (battery, localization status, motor health).
- This approach fosters user trust, especially as emergent behaviors (e.g., dynamic obstacle avoidance, adaptive navigation) become more common in field robots.
- Teams are encouraged to share methods like these to strengthen and enhance the Robotics Dojo community.