Limit Breakers' Technical Design Paper for the Robotics Dojo Competition 2025

Nathan Kingori, Nasir Jacob, Mohamed Ibrahim, Iregi Mwenja, Brumley Ogari – (**Limit Breakers**)

A. Design Strategy

The design strategy for the Limit Breakers project involved a collaborative effort among the team members This team brought together expertise in both electrical design and software development to address the competition challenges efficiently.

General Approach to Competition Challenges:

1. Modular Design and Specialisation: The team's approach was based on dividing the robot's development into two core areas: the mobile platform and the navigation system. This modular strategy allowed for focused work on each aspect, ensuring that the platform provided robust mechanical support, while the navigation system was optimised for real-time mapping and decision-making. This clear division of tasks enhanced both development speed and reliability.

2. Mapping and Navigation: Focusing on real-time obstacle detection and autonomous movement, the robot was designed to make quick and accurate decisions during the competition. The RP Lidar provided 360-degree scanning, ensuring that the robot could understand its environment and navigate effectively without human intervention.

3. Testing and Optimization: The team allocated significant time to testing and refining the mobile platform and navigation. The navigation algorithms were tested using ROS2 simulations (Rviz and Gazebo) and real game field trials, ensuring the system's capability and reliability in dynamic environments.

4. Reliability vs. Complexity

- Reliability Over Complexity: The team opted for reliable, proven technologies that ensured system stability. For instance, while more advanced sensors or processors like the NVIDIA Jetson could increase capability, they would introduce complexity and potential failure points. The decision to use the Raspberry Pi 4 balances computational power with reliability, as it can handle tasks like running ROS2 without overwhelming the system.

- Motor Control Simplicity: The mobile platform employed closed-loop control using motor encoders for precise movement. This allowed the robot to maintain accurate speed and position control without introducing the complexities of open-loop systems. The use of an L298N motor driver ensured adequate power delivery without unnecessary complications.

5. Capability vs. Robustness

- Capability in Navigation: The navigation team focused on enhancing the robot's mapping capability, using RP Lidar to create detailed and real-time maps of the environment. This capability allowed the robot to navigate complex game field paths and avoid obstacles autonomously. The team maintained a robust system by avoiding unnecessary features that could compromise stability.

- Robust Mobile Platform: The team emphasised the durability and stability of the mobile platform. The chassis was made from lightweight acrylic, ensuring the robot was both stable and energy-efficient. A round chassis design was used to improve manoeuvrability, allowing the robot to rotate easily in tight spaces, further enhancing its navigation performance.

6. Testing for Reliability

Given the limited preparation time, the team dedicated resources to testing and improving the reliability of the existing systems. The team ensured that the robot can operate consistently under competition conditions by continuously testing the integration of the mobile platform and navigation systems in both simulations and real-world environments. This approach minimised the risk of failure, as potential issues were identified and addressed early in the development process.

B. Vehicle Design

1. Design process

CAD design (Mechanical design) – Autodesk Inventor was used to design the parts.

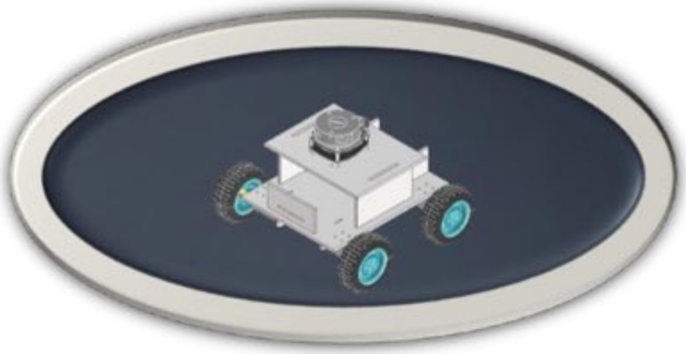

Figure 1 CAD Design Assembly

Electrical circuit design was done.

Budget preparation – bill of materials required was prepared and forwarded for purchase.

Purchase of components – components were purchased and distributed.

Mechanical components fabrication - laser cutting of the acrylic, the parts were glued together,

Electrical components circuit design – electrical circuit connection was done (motors, motor driver, Arduino, Raspberry Pi, Lithium-Ion batteries, power bank)

Integration of mechanical and electrical components - wheels, motors, couplers, motor brackets, batteries, Arduino and chassis (temporary robot assembly) was done

Move the robot using the microcontroller (first testing) – the robot was first moved to test wheel alignment, motors condition and overall assembly.

Navigation development - The navigation stack was implemented on a robotic system using ROS2. The navigation setup integrated multiple software packages, allowing the robot to autonomously move and explore its environment. The primary goal of the navigation system was to enable the robot to map, localize, and navigate in a simulated or real-world environment using ROS2.

Integration of mobile platform and navigation – lidar addition, map creation (mobile platform and navigation team working together).

Final assembly - permanent robot assembly

Testing - game field testing, mapping and navigation

Finalising - final fixes and perfecting the navigation.

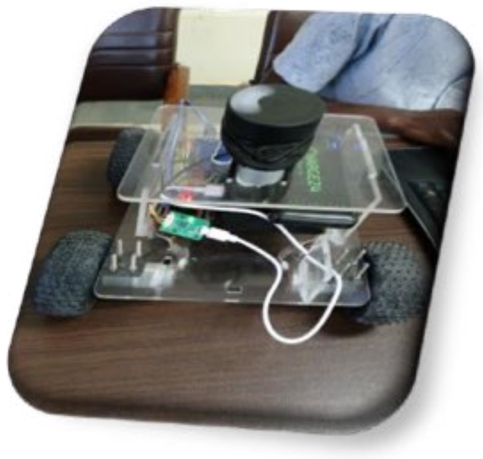

Figure 2 Final Robot assembly (tested)

2. Methodology and design result

Design considerations - During the design, several considerations were applied to ensure that the right components were chosen for the robot. Each part used is discussed below:

Raspberry Pi
Raspberry Pi is the brain for the robot. The following are key considerations for choosing a computer for this kind of project:

Figure 3 Raspberry Pi

Why Raspberry Pi: The Raspberry Pi 4 was the recommended option due to its ubiquity, relatively low cost, and strong community support. While simpler robots might run on a smaller chip like an Arduino, the Pi is powerful enough to handle more complex software, such as ROS (Robot Operating System), which requires more computing power. ROS2 was used in this project and it required a powerful chip like Raspberry Pi.

Power: The Raspberry Pi can be powered by a 5V source, making it convenient for robotic applications. For other options like laptops or the Intel NUC, larger batteries may be required.

Alternatives: Single-board computers like the **NVIDIA Jetson**, which could be better for GPU-heavy tasks like machine learning. A **laptop or an Intel NUC** can also be considered for more demanding projects, but these options could be more complex due to power and size constraints.

Power Considerations

Devices are rated for the maximum power they can draw, and components must work within these limits to avoid overheating or damage.
Power sources used are Power bank (for microcontroller) and LiPo (for motors).
The following are the key concepts and considerations for powering the robot:

1. Voltage Considerations

Operating voltage - It was essential to consider the operating voltage of each device.
5 volts: For microcontrollers (Raspberry Pi and Arduino) and USB devices. Power bank used since it outputs a stable 5V.
12 volts: For the DC motors, which required higher power. LiPo was used.

2. Current Considerations

Current Draw - It was essential to estimate the total current draw of all the components in the robot to choose the right power supply and regulators.
For example, Raspberry Pi and lidar draw up to 5 amps.
Motor Current: Motors draw different amounts of current based on load and torque. The stall current is the maximum current drawn when the motor is under maximum load, and it is crucial to ensure the motor driver can handle it.

3. Battery Choice

Battery Selection: The choice for our mobile robot was a Lithium-Ion Cylindrical Battery. Each is rated 3.7V.
Considerations include:
- Lower cost than other alternatives like LiPo battery.
- Lighter
- Easy to recharge and requires less monitoring compared to LiPO which is sensitive and requires regular monitoring.

Figure 4 Lithium-Ion Batteries

4. Wiring and Connectors

Ensured that all wiring and connectors were rated for the currents required. For example, breadboards and jumper wires should only be used for very low current, while heavier-duty connectors like XT60 are required for high-current applications.

5. Power Bank Choice

Power bank was used to power the Raspberry Pi. The following Product Power Parameters were considered:
Capacity: 20000mAh (74Wh) Input 1(Micro-USB): 5V/2A
Input 2(Type-C): 5V/3A
Output 1(USB-A): 5V/3A (Max)
Output 2(USB-A): 5V/3A (Max)
Output 3(Type-C): 5V/3A (Max)
The output did not exceed 5V / 3A which is safe for the Raspberry Pi.

Figure 5 Power Bank Powering Raspberry Pi

5. Safety Features

- Power switch: Allowed for safe disconnecting of the battery.
- Fuses: Protected the circuit from short circuits by breaking the circuit if too much current flowed, preventing damage and fire hazards.

Motors

Designing a motor system for the robot involved several key considerations and calculations to ensure optimal performance. These considerations were broken down into several layers, each contributing to the motor's overall operation and control.

1. Motor Selection

- Motor Type: Different types of motors (for example: brushless, stepper) require varying control methods, so it was essential to choose the right type for the application. We chose a DC 12V 200 rpm motor.

Figure 6 Motor Selected (DC 12V 200 RPM with Encoder Module)

- Voltage and Current Requirements: Motors typically required higher voltage and current than what a microcontroller could provide. For example, a 12V DC motor was used and the motor's current rating was considered to avoid overloading the control circuitry.
- Motor with Encoder: Motors equipped with encoders provide feedback on speed and position, enabling closed-loop control for more accurate and stable motor behaviour. We chose a motor with encoder which made speed and position control easier.

2. Motor Control

- Open-Loop Control: This simpler method maps the desired speed or position to a specific PWM value, but it lacks feedback, which makes it less accurate, especially under varying loads. Pulse Width Modulation (PWM) is used to control motor speed by modulating the on/off signal to the motor. The duty cycle of the PWM signal determines the motor's effective voltage and, thus, its speed.

Duty Cycle = (On Time / Total Time)

The motor's effective voltage is proportional to the PWM duty cycle. A higher duty cycle means the motor will run faster, and a lower duty cycle means it will run slower.

- Closed-Loop Control (Feedback Control): To achieve more precise control, closed-loop feedback is necessary. Encoders on the motor provide real-time speed or position data, which is used to adjust the PWM signal dynamically. In a closed-loop system, the motor's actual performance is measured, and adjustments are made to match the desired performance.

The most common closed-loop control method is Proportional-Integral-Derivative (PID) control. The controller adjusts the motor's input based on the difference between the target speed (or position) and the actual measured value.

The motor's speed is calculated based on encoder feedback, which sends pulses corresponding to the motor's rotation. By counting the number of pulses in a given time frame, the controller can determine the motor’s speed.

The encoder produces a certain number of pulses per revolution (PPR). For example, if the encoder gives 100 pulses per motor revolution, and the gear ratio multiplies the output revolutions, you can calculate the total output revolutions using the formula:

Speed (RPM) = (Encoder Counts / PPR) x Gear Ratio

This equation converts the encoder feedback into the motor's speed in revolutions per minute.

Closed- loop control was used to control the motors.

3. Speed, Torque and Load Considerations

The torque the motor generates depends on the current it draws and the mechanical load it encounters. The motor's torque needs to be considered if sufficient for the expected load, considering factors like gear ratios, friction, and inertia.

Used a 200-rpm dc motor. Encoders used to control its speed. Compared to a 130-rpm motor, it was better to have a higher speed motor whose speed could be reduced than a low-speed motor (130 rpm) whose speed cannot be increased beyond the rated value.

Suppose the 65mm wheel diameter, and max speed = wr, Then.

$w_1 = 130*2\pi/60 = 13.61$ rad/s

$w_2 = 200*2\pi/60 = 20.94$ rad/s

max velocity_1 = w1xr = 0.44m/s

max velocity_2 = w2xr = 0.68m/s

4. Power

The motor and the motor driver should have a sufficient and stable power supply. For example, the L298N motor driver has a voltage drop that needs to be accounted for, meaning the motor may receive less than the input voltage.

Lithium-ion batteries was selected to power the motors, which was chosen for their lower cost and met power requirements of the motors.

5. Communication

Serial (UART): Used to send speed commands to the Arduino motor controller from the Raspberry Pi. CAN, I2C, or PWM can also be used in other setups depending on the motor controller.

Motor Driver

A motor driver was essential for controlling the motor's speed and direction. Key considerations include:

- Voltage Matching: The motor driver matched the motor's voltage requirement (12V for motor chosen).
- Current Capacity: The driver supported the current requirements of the motor. For example, the L298N motor driver used, supports 2A of continuous current with spikes up to 3A.

Figure 7 Motor Driver (LN298N)

RP Lidar

RP Lidar was the primary sensor used in the robot to map its environment, such as a game field. Mapping was essential for:

- Environmental Awareness: Mapping enabled the robot to understand and navigate its environment efficiently, avoiding obstacles and planning paths allowing it to reach specific points in the game field and navigate effectively.
- Autonomous Movement: With a map of its surroundings, the robot could move autonomously, making real-time decisions about where to go or how to react to dynamic changes in its environment.
- Task Optimization: Mapping allowed for more precise and optimised task execution, whether it was navigating a maze, following a path, or interacting with objects on the game field.

Figure 8 RP Lidar

Why use RP Lidar?

- High Accuracy: RP Lidar provides accurate 360-degree scanning of the environment, giving the robot a detailed map of its surroundings.
- Low Latency: It offers fast real-time updates, which is essential for responsive movements in dynamic environments.
- Lightweight and Compact: It is small and light, which minimises the weight burden on the robot, allowing for faster and more agile movement.
- It is also affordable.

Chassis

The chassis supports the robot's electrical and electronic components. It was made from acrylic and is round in shape.

Acrylic was used because it is:

Lightweight: Acrylic is much lighter than metals like steel or aluminium, reducing the overall weight of the robot, which helps in mobility and energy efficiency.
- Cost-Effective: Acrylic is relatively inexpensive, making it a cheaper option for the competition's small budget.
- Non-Conductive: Acrylic is a poor conductor, preventing electrical shorts or interference with electronic components.
- Ease of Machining: Acrylic is easy to cut, shape, and drill, allowing for greater flexibility in design and quicker assembly. It was cut using a laser cutting machine which was fast and efficient. Assembling was also easy and quick using glue.

Why rectangular shape?
- Improved Manoeuvrability: A round chassis allowed for better movement in tight spaces since the corners won't get caught on obstacles. This shape is necessary as the robot needs to rotate frequently in the game field.
- Even Weight Distribution: A round design helped distribute weight more evenly, improving balance and stability, during turning or when carrying loads.
- Optimised for omnidirectional movement as the robot used castor wheels making smooth turns possible.

Couplers
Couplers were used to connect the motors and rubber-driven wheels. They were used because of:
- Efficient Power Transfer: Couplers ensure efficient transmission of torque from the motor to the wheels, minimising energy loss.
- Vibration Dampening: They help to absorb small misalignments and vibrations between the motor shaft and wheels, prolonging the lifespan of both the motor and the wheels.
- Couplers make it easier to align components (wheels and motor shafts) that may not have perfectly matching shafts sizes, improving flexibility in the mechanical design.

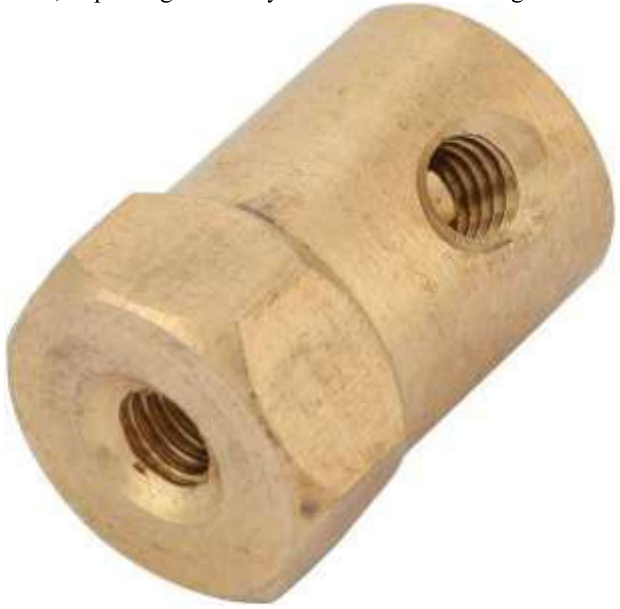

Figure 10 Coupler

Motor Brackets
Motor brackets were used to mount the motors securely on the chassis. They were used for:
- Stability: Properly mounting the motors ensures that they stay securely in place, even when the robot is moving over uneven terrain or at high speeds.
- Alignment: Motor brackets help to align the motor correctly with other components, such as wheels and couplers, ensuring smooth and efficient operation.
- Vibration Reduction: A secure mount reduces vibrations that can cause wear and tear on both the motor and the chassis, improving the robot's durability.
- Brackets provide flexibility in positioning motors, allowing adjustment of motor placement based on the robot's overall layout.

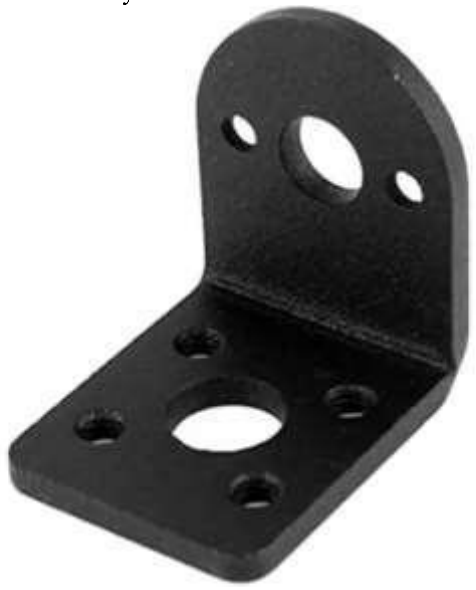

Figure 11 Motor Bracket

Rubber Wheels
Driven by the DC motors to move the robot from one point to another. 65 cm in size. Chosen because of its ease to be coupled with the motors using couplers. Its size (65 cm diameter) was ideal for stability and lidar mapping (not too high or too low).

Figure 12 Rubber Wheels

Castor Wheels
Two castor wheels were used. Castor wheels were used because:

1. Manoeuvrability and free rotation: Castor wheels allowed the robot to turn smoothly without needing all wheels to be powered. This was useful for multi-directional movement.

2. Stability: They provide additional support and stability, as the robot had two main drive wheels, preventing the robot from tipping over.

3. Reduced Friction: Castor wheels can swivel in any direction, minimising the friction that could occur when the robot turns, allowing smoother and easier movement.

4. Cost: Castor wheels are simpler and cheaper than powered wheels.

5. Compact Design: They require less space than the driven wheels.

Lessons learnt:

- Accuracy in CAD designs dimensions.
- Ensuring purchase of the components having the same dimensions as the designed model. This involves checking the datasheets and seller's descriptions.
- Use of bolts and screws is encouraged over glue due to its ease of removal in case of errors during placement. Glue is difficult to remove and could alter aesthetics, material or dimensions.

C. Navigation

1. Introduction

This section covers the navigation stack implemented on a robotic system using ROS2. The navigation setup integrates multiple software packages, allowing the robot to autonomously move and explore its environment. The primary goal of the navigation system is to enable the robot to map, localize, and navigate in a simulated or real-world environment using ROS2.

2. Navigation Packages Overview

The ROS2 navigation system relies on several key packages to ensure robust and accurate movement. Below is an overview of the key packages used:

Sllidar: This package integrates a 2D LIDAR for scanning the environment. The LIDAR provides distance measurements to obstacles, which are critical for mapping and localization tasks.

Serial: The serial package opens a communication link (typically via USB or UART) between the Raspberry Pi and Arduino.Velocity commands, such as linear and angular velocities, are sent from the Pi to the Arduino over this serial connection.The Arduino processes these commands to control the motors and sends encoder data back to the Pi for odometry calculation.

Diff drive Arduino: This package interfaces with a differential drive robot. It sends commands to the motors based on velocity inputs, typically received from ROS2 navigation or teleoperation packages, and reads feedback from encoders for odometry. It uses the *serial package* to manage communication between the Raspberry Pi and the Arduino.

Gazebo: A simulation environment used to simulate the robot in a virtual environment, including sensors like LIDAR and actuators like motors. Gazebo is tightly integrated with ROS2, allowing real-time testing of navigation algorithms.

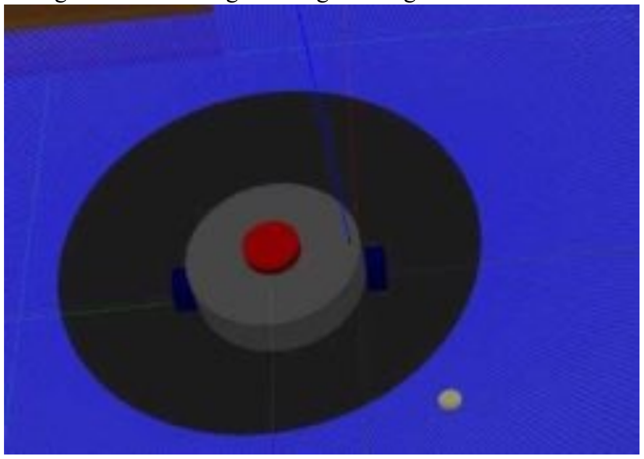
Figure 14 Robot in Gazebo

Rviz: A visualization tool in ROS2 that allows the user to view the robot's sensor data, map, and planned trajectories in real time. This is useful for debugging and tuning the navigation stack.

Colcon: A build tool for ROS2. It is used to compile and link the various packages within the navigation stack.

Teleop_twist: This package enables manual control of the robot using a keyboard or joystick. It publishes velocity commands (`/cmd_vel`) that are interpreted by the diff drive package to control the robot's motors.

Twist_mux: This package multiplexes various sources of velocity commands (`/cmd_vel`), ensuring that the correct source (e.g., teleoperation or autonomous navigation) has control over the robot at the appropriate time.

Nav2: The main package for ROS2 navigation. It provides functionality for planning, controlling, and recovering the robot's trajectory. Nav2 integrates sensors (like LIDAR) and odometry data for localization, map updates, and autonomous path planning.

Slam Toolbox: This package handles Simultaneous Localization and Mapping (SLAM), allowing the robot to build a map of its environment while localizing itself within that map. It's crucial for operations in unknown or dynamic environments.

ROS2 Control: Used to manage robot hardware resources (such as motors and sensors), ensuring that commands from navigation or teleoperation systems are properly executed on the hardware side.

3.Robot Navigation Architecture

The ROS2 navigation stack is designed to enable autonomous navigation by integrating perception, planning, and control components. The navigation process consists of the following phases:

1. Perception: The robot uses the RPlidar sensor to gather distance data from its environment. This data is processed by the SLAM Toolbox to generate a 2D occupancy grid map.
2. Localization: Once the map is generated, the robot localizes itself within the environment. Localization is based on the matching of LIDAR scans to the pre-built map.ACML may be used.
3. Path Planning: Using Nav2, the robot determines a path to a goal position. Nav2 uses algorithms like,Dijkstra's or A* to plan optimal paths, considering the known map and obstacles.
4. Control: The robot's motors are controlled via the diff drive Arduino package, which receives velocity commands (from Nav2 or Teleop_twist) through the Twist_mux. The robot then follows the planned trajectory.
5. Recovery: Nav2 includes recovery behaviors in case the robot encounters issues (e.g., getting stuck). This could involve backing up, rotating in place, or re-planning a new path.

4. Simulation and Testing

Testing the navigation system is performed in the Gazebo simulation environment. Gazebo allows for creating complex environments with obstacles, ramps, and walls, simulating the robot's interaction with its surroundings.

In the simulation, the following components are tested:

LIDAR sensor accuracy: Ensures that obstacle detection works as expected.

- Path planning: Verifies that Nav2 can compute paths in various types of environments.
- Localization robustness: Tests how well the robot localizes itself using SLAM Toolbox and odometry.
- Obstacle avoidance: Ensures the robot can avoid obstacles in real-time by recalculating paths when

necessary.

All simulated outputs can be visualized in Rviz, providing insight into sensor readings, map quality, and the robot's trajectory.

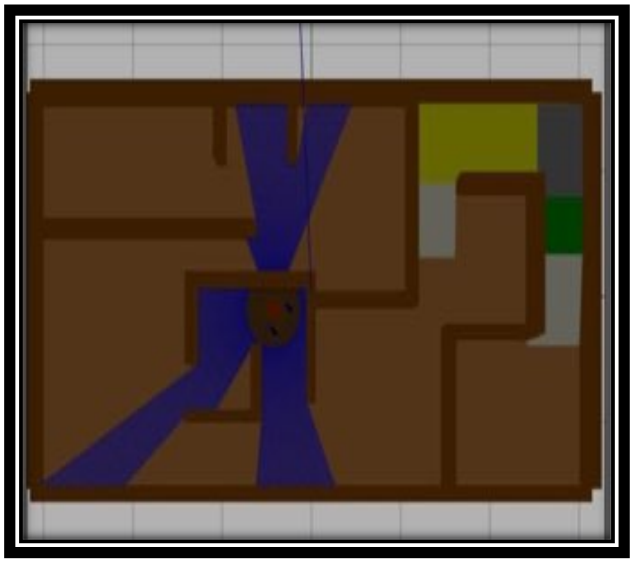

Figure 15 Gazebo Simulation showing the robot and the map (game field)

5. Challenges

Several challenges are encountered when building the navigation stack:

Sensor noise: LIDAR data can be noisy, which affects the accuracy of mapping and localization.

Odometry drift: The encoders used by the diff drive system can introduce errors over time, affecting the robot's localization accuracy.

Real-time performance: Ensuring that the entire navigation stack runs in real time, especially with complex environments and many obstacles, requires fine-tuning of the control loop and sensor processing rates.

6. Conclusion

The ROS2 navigation stack leverages multiple packages to create a robust, autonomous navigation system. The integration of LIDAR-based perception, SLAM for mapping, and differential drive control allows the robot to navigate its environment efficiently. Future work may involve optimizing the system for hardware deployment and improving recovery behaviors in complex environments.

D. Experimental Results

To ensure the reliability and performance of the Project we carried out extensive testing, including unit tests, integration tests, and simulations using ROS2, Rviz, and Gazebo, as well as real-world field trials.

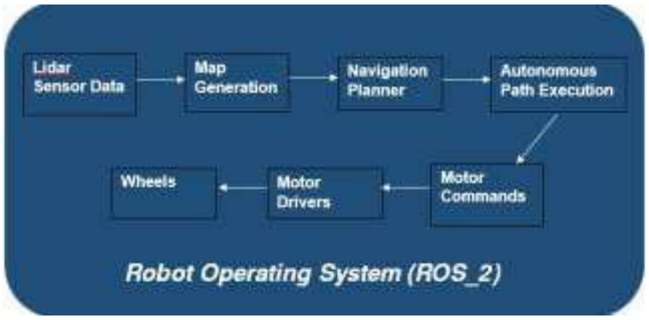

Figure 16 ROS2 Interaction

1. Testing Procedures

1. Unit Testing: Individual components like the motors, sensors (RP Lidar), and motor drivers (L298N) were tested independently. This included:

- Motor Testing: The motors were tested under different speeds, ranging from 100 RPM to 200 RPM, with feedback from encoders to validate speed control and accuracy.
- Sensor Testing: The RP Lidar was tested for its mapping accuracy in static and dynamic environments. The sensor was validated by comparing its readings with known distances and obstacles on the game field.

2. Integration Testing: Once individual components were tested, the integration of the mobile platform and navigation system was done. Key aspects tested included:

- Synchronisation between motor control and real-time mapping.
- Power stability from LiPo batteries and the power bank for smooth operation without unexpected shutdowns or overheating.
- Closed-loop control of the motors using encoder feedback for precision and path accuracy.

3. Simulation: Before deploying in real-world environments, the team utilised Gazebo and Rviz simulations to test the robot's mapping and navigation capabilities. The robot was simulated navigating complex game fields with obstacles, testing its path-planning algorithms and responsiveness to environmental changes.

- Gazebo Simulation: Focused on physical interactions like obstacle avoidance, smooth movement, and collision detection.
- Rviz: Used for visualising Lidar-based mapping and real-time sensor data to assess the navigation system's accuracy.

2. Testing Outcomes

- Motor Performance: At lower speeds (200 RPM), the motor encoders provided closely matching readings, ensuring reliable movement. However, at higher speeds (200 RPM), encoder discrepancies increased, requiring calibration. The motor restart issue at longer distances was identified, which temporarily fixed encoder inconsistencies.

- Power Supply: The Lithium-ion battery and power bank were tested for voltage stability under varying loads. Both power sources provided adequate power without any significant dips, allowing consistent performance in the motors and Raspberry Pi.

- Mapping and Navigation: The RP Lidar produced highly accurate maps of the game field in both simulation and physical tests, allowing the robot to navigate autonomously with minimal error. In real-world tests, the robot successfully detected and avoided obstacles, demonstrating the efficacy of the integrated mapping and control systems.

3. Reliability and Robustness

- Robustness Analysis: The acrylic chassis proved durable during field tests, showing no signs of warping or damage under stress. The motor brackets provided stable mounting, minimising vibrations that could lead to component wear over time.

- Reliability Modelling: Based on the testing data, failure points were identified in the motor control system at higher speeds, specifically related to encoder accuracy. To mitigate this, the team implemented calibration strategies and adjusted the control algorithms to improve speed synchronisation.

- Failure Analysis: The team performed failure analysis on key components such as the motors and power system. No major failures were detected, but minor issues like motor speed discrepancies and temporary power fluctuations were noted and addressed in the design.

The robot's performance met the expected design criteria, with minor improvements required to enhance speed accuracy at higher RPMs and to further optimise power management during extended operation.

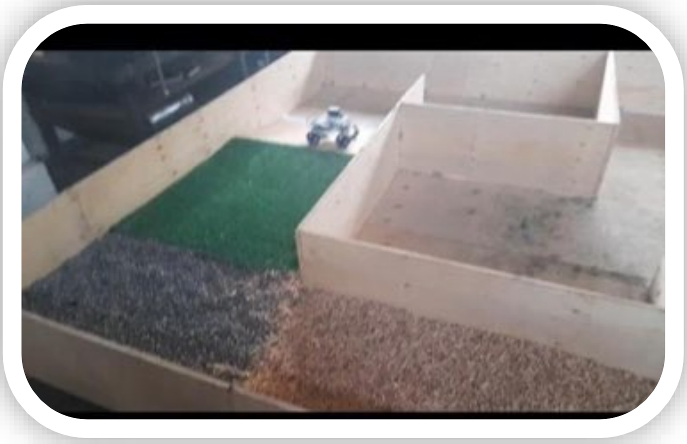

Figure 17 Robot under game field testing (navigation) before the competition

E. Acknowledgements

Limit Breakers acknowledges the support from other teams, students and staff for their assistance through trainings, guidance, problem solving and support throughout the process.

F. References

[1] DATASHEET Raspberry Pi 4 Model B Release 1.1 March 2024 Copyright 2024 Raspberry Pi (Trading) Ltd. All rights reserved. raspberry-pi-4-datasheet.pdf (raspberrypi.com)
[2] Articulated Robotics - Building a Mobile Robot. https://youtube.com/playlist?list=PLunhqkrRNRhYAffV8JDiFOatQXuU-NnxT&si=Pa_dVk3U2P824uX5