Zingira Bot Technical Report

Autonomous Waste Detection Sorting and Navigation Robot

Dickson Kabiru, Joan Ouma, Nikita Kamau, Lewis Karani
Robotics Dojo, AFRICA-ai-JAPAN Project (JICA), Jomo Kenyatta University of Agriculture and Technology, Kenya

Abstract

Zingira Bot is a compact autonomous mobile robot developed for the Robotics Dojo 2026 waste-management challenge. The system combines a Raspberry Pi 5 running ROS 2 Jazzy, LiDAR, camera-based perception, differential drive, and a custom ROS 2 Control hardware interface. During operation, the robot maps the arena, plans collision-free routes, detects waste, and selects the appropriate collection destination. This report documents the mechanical and system architecture, software design, electrical-design interface requirements, test methodology, and results-reporting framework.

1 Introduction

The Robotics Dojo 2026 Challenge models an urban waste-management task in a constrained arena. The robot must recognize plastic, organic, and metal items, traverse variable terrain, avoid dynamic obstacles, and deposit each item in the matching color-coded zone. The challenge includes a three-minute manual mapping phase followed by a three-minute autonomous phase. The arena contains sand, grass, ramps, and a time-based barrier that requires a route-selection decision.

Zingira Bot was designed within a maximum of 300 mm by 300 mm by 300 mm by 300 mm form factor. Its design objective is reliable, repeatable autonomous navigation while preserving enough compute capacity for real-time perception and motor control.

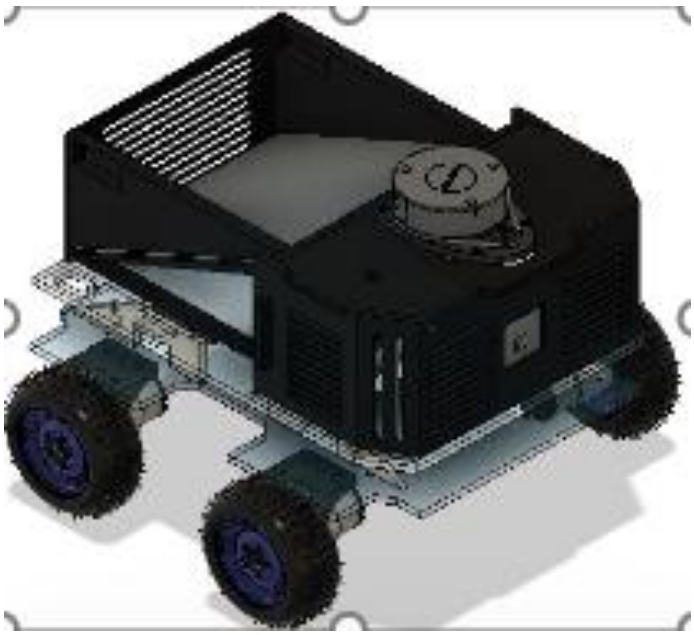


Figure 1. Zingira Bot assembled platform.

2 System Architecture

The system is organized into four cooperating layers: mechanical mobility, power and embedded hardware, sensing and perception, and autonomous software. The Raspberry Pi 5 acts as a high-level computer. It receives LiDAR, camera, and state information, runs ROS 2 nodes, and issues motion commands through a custom hardware interface to the low-level controller and motor drivers.

Subsystem	Primary function	Key elements
Mobility	Traverse mixed terrain and ramps	Differential drive, 85mm wheels, chassis, wheel couplers
Compute and control	Run autonomy and coordinate devices	Raspberry Pi 5, ROS 2 Jazzy, Arduino or equivalent low-level controller
Perception	Map the arena and identify waste	Planar LiDAR, Raspberry Pi, IMX219 PI Camera, MPU-6050 IMU, Hailo-8L accelerator
Actuation and safety	Drive motors and provide safe shutdown	High-current motor drivers, main switch, power-distribution network, LEDs

2.1 Mechanical Design

The chassis uses 300mm x 260mm x 5mm clear acrylic, to support the drive system, sensing hardware, and payload compartment. Large wheels and secure couplers provide clearance and traction for sand, grass, and ramps. The design is constrained by the competition envelope and must retain adequate structural stiffness under acceleration, braking, and obstacle-avoidance maneuvers.

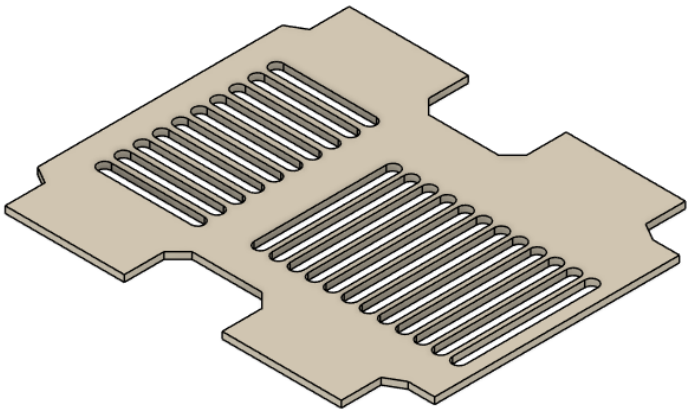


Figure 2. Base chassis layer.

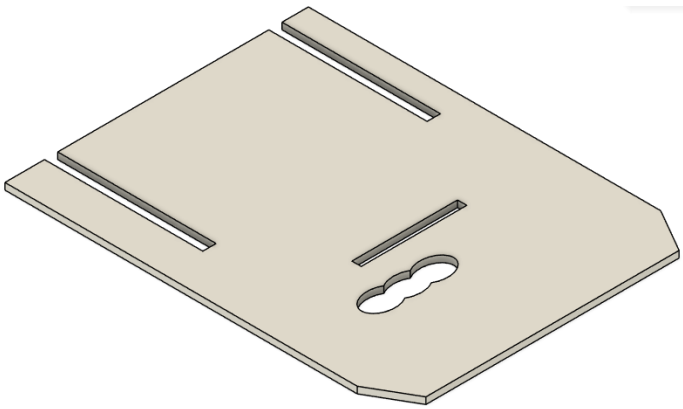


Figure 3. Upper chassis layer.

2.2 Electrical Design

The electrical system distribute power safely to the Raspberry Pi 5, low-level controller, LiDAR, camera, IMU, LEDs, and motor drivers while isolating high-current motor loads from sensitive logic supplies. A master switch provides an immediate system shutdown. Motor drivers (L298N) supply a max of 3A to the motors, therefore can handle the measured motor (JGB37-520 110 RPM) stall current 2.7A, and all devices must share a controlled common-ground strategy.

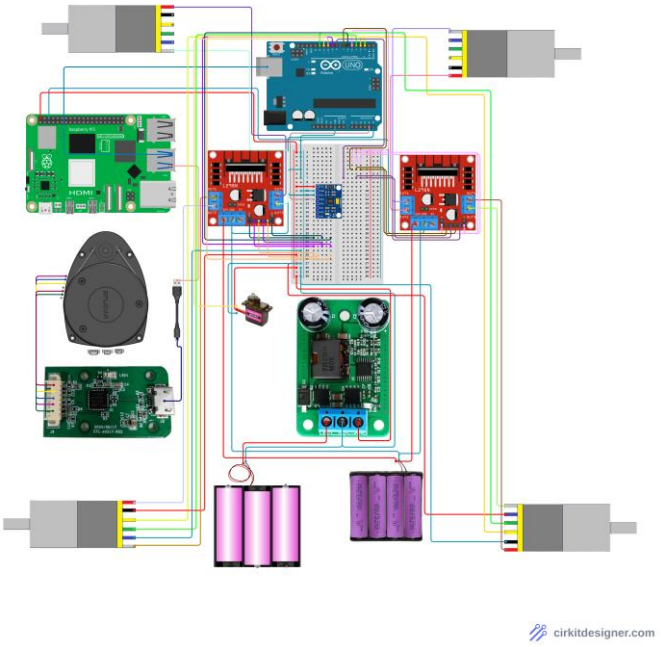


Figure 4. Zingira Bot circuit diagram

2.3 Sensor and User Interface

A planar LiDAR provides a range of data for mapping and obstacle avoidance. A camera provides waste detections, and the MPU-6050 contributes to orientation and acceleration measurements for localization to support uneven terrain.

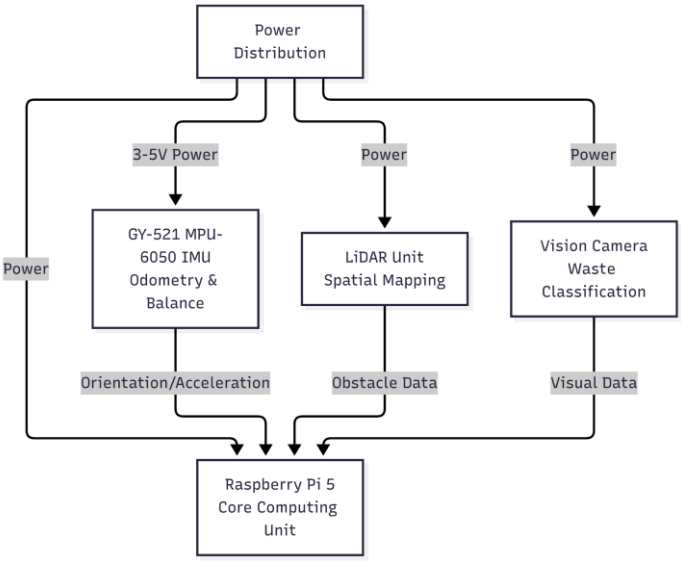


Figure 5. Sensor and compute block diagram.

3 Software Architecture and Algorithms

Zingira Bot uses ROS 2 Jazzy as a modular message-based middleware. The architecture separates high-level mission behavior from low-level motor actuation. This separation allows navigation and perception components to be tested or modified without replacing the embedded motor-control implementation.

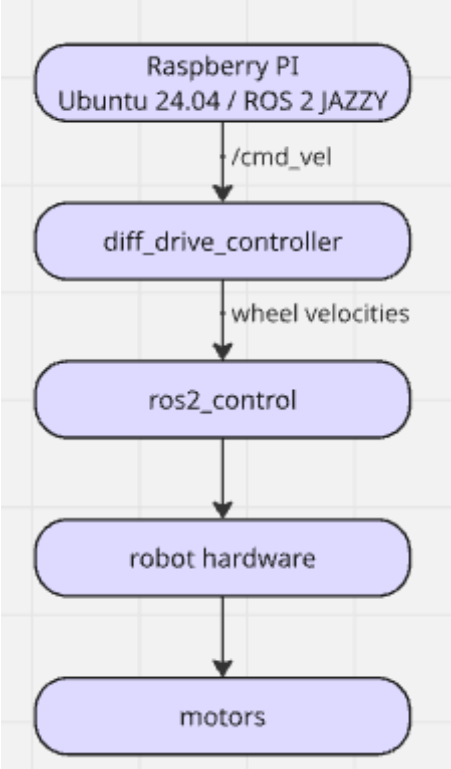


Figure 6. ROS 2 software architecture.

3.1 Mapping Localization and Navigation

During the mapping phase, SLAM Toolbox consumes LiDAR scans and wheel odometry to create a two-dimensional occupancy grid while estimating the robot's pose. During autonomous operation, Nav2 uses the map, localization estimate, costmaps, and obstacle observations to plan and execute routes. TF2 maintains the coordinate transforms between the map, odometry, base, and sensor frames.

Navigation should be tuned and reported using the final parameter file. The team should record the planner and controller selected, costmap resolution, inflation radius, maximum linear and angular velocity, and the lidar obstacle-range settings.

3.2 Mission Decision Algorithm

Mission behavior is implemented as a Behavior Tree. The tree sequences mapping completion, autonomous navigation, waste-response actions. At the time-based barrier, it waits for the gate to open so that it can take the easier way to the offloading bay.

Step	Decision or action	Output
1	Read current mission state, pose, and costmap	Robot state estimate
2	Check for a safety stop, blocked path,	Stop or recovery request when needed

	or localization failure	
3	If a waste detection meets the selected confidence threshold (0.55), identify its class and target zone	Plastic, organic, or metal destination
4	At the barrier, compare gate availability and estimated route costs	Wait or reroute decision
5	Send navigation goal and monitor progress	Velocity commands or recovery action
6	On task completion, stop motors and publish completion state	Safe idle state

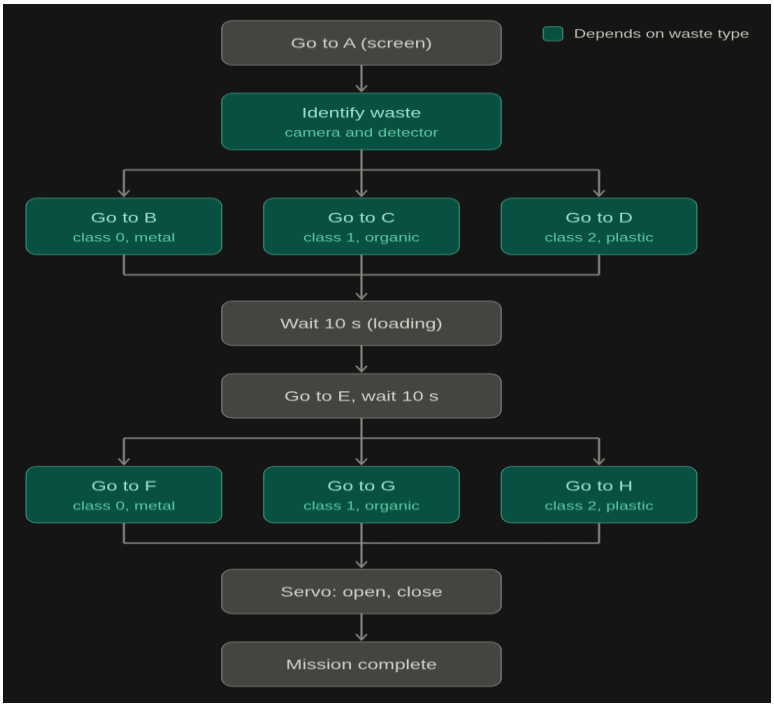


Figure 7. Behavior Trees flow charts

3.3 Hardware Interface and Motor Control

The custom Robot2System interface implements the ROS 2 Control System Interface contract. Its lifecycle validates parameters, establishes serial communication, initializes encoder state, and explicitly commands a stop before activation and deactivation. The read operation converts encoder ticks to wheel position and velocity in SI units. The write operation converts commanded wheel velocities into the low-level command format used by the motor controller.

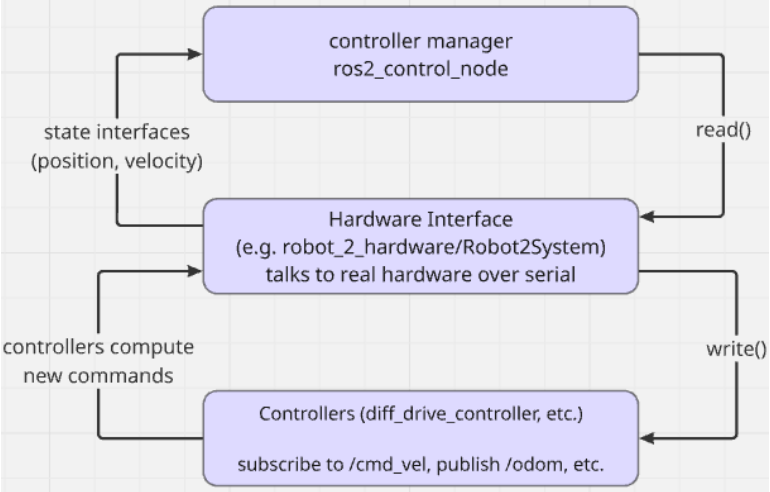


Figure 8. ROS 2 Control hardware interface flow.

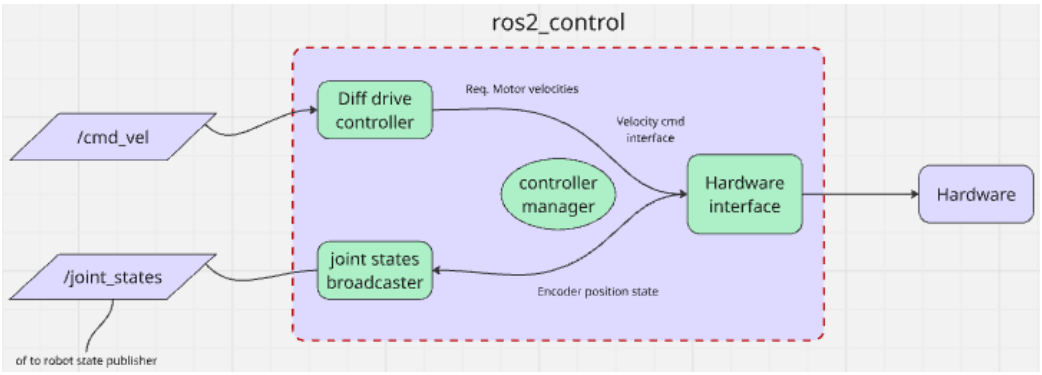


Figure 9. Internal data flow of the custom hardware interface.

4 Vision and Waste Classification

The perception pipeline uses the Raspberry Pi IMX219 Camera, and a YOLO detector optimized for edge execution on the Hailo-8L accelerator through the AI HAT+ platform. Offloading inference from the CPU helps preserve compute capacity for navigation, SLAM, and control. The onboard CPU can be used for the execution of the detection as an alternative to the AI HAT+ if it fails. The vision node publishes object class, confidence, image location, and, where available, relative pose to a ROS 2 topic consumed by the mission behavior.

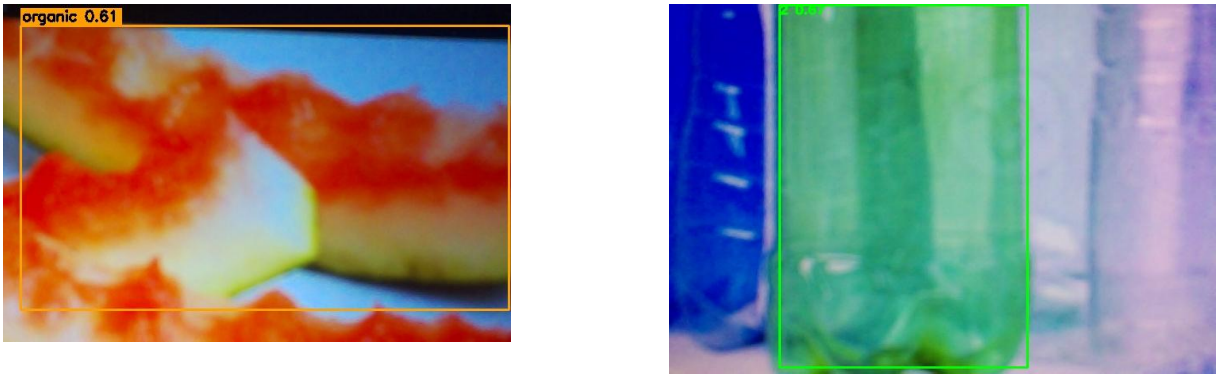


Figure 10. Annotated image showing the class and confidence score of the AI model

5 Test Methodology

Testing combines simulation and field evaluation. A Gazebo Sim digital twin models the six-meter by three-meter field using separate visual and collision meshes. The simulated robot uses a differential-drive plugin, name spaced ROS 2 topics, and a LiDAR model. A one-millisecond simulation step and real-time factor of one are used where the host platform can sustain them.

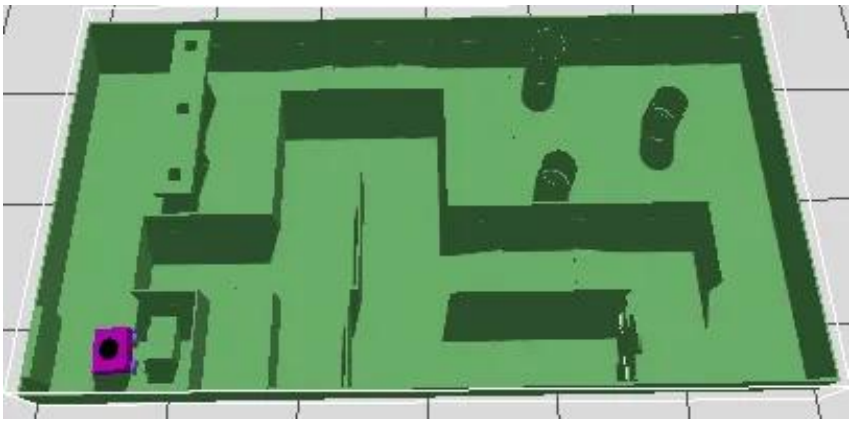


Figure 11. Gazebo simulation model of the competition arena.

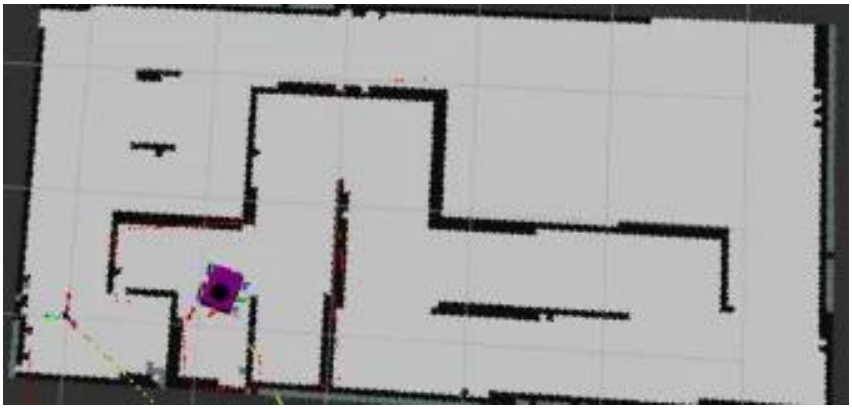


Figure 12. Example virtual map produced during simulation.

Physical testing Included mapping of the real gamefield.

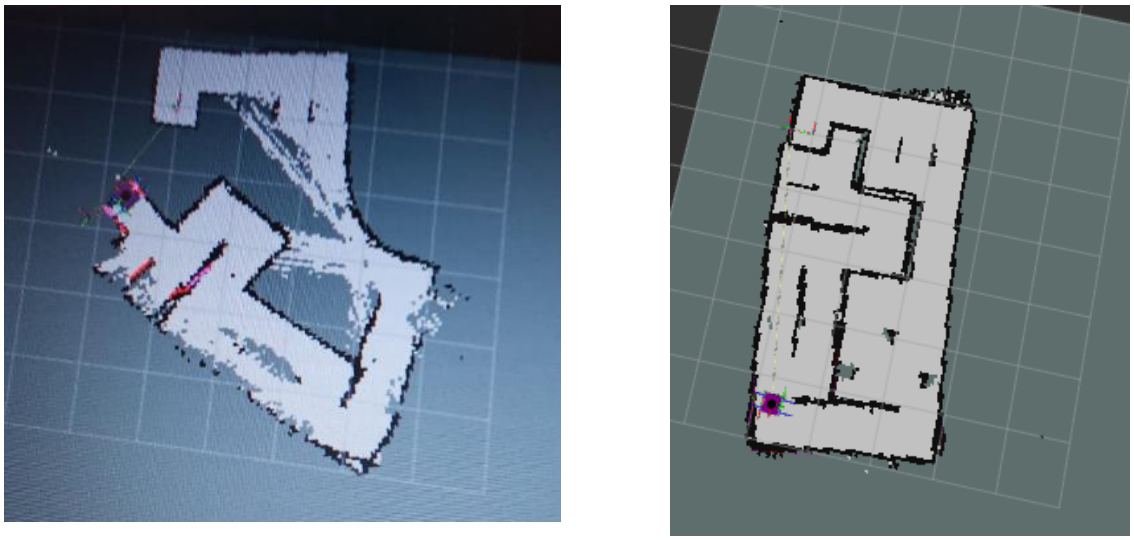


Figure 13. Image showing the worst map and the best map developed during the physical testing of the robot at the gamefield

6 Discussion

Qualitative observations from early field trials indicate that the robot can traverse sand, and ramps, while wheel slip on uneven surfaces can degrade encoder-only odometry.

7 Conclusion and Future Work

Zingira Bot integrates mechanical design, embedded hardware, ROS 2 navigation, custom motor control, and edge-AI waste classification into a compact autonomous platform. The architecture supports modular development: perception, navigation, and hardware components can be tested independently and integrated through ROS 2 interfaces.

Zingira Bot successfully achieves its core objectives of mapping and autonomous navigation, though its development required overcoming key hardware and power constraints. When traversing grassy terrain, wheel slippage initially caused severe distortions in the mapping data; this was resolved by integrating an MPU-6050 IMU gyroscope to provide accurate orientation data. Additionally, meeting the high-spec power demands of the Raspberry Pi 5 (5.1V, 5A) posed a financial challenge given our tight budget. We engineered around this by using a cost-effective 5V, 5A buck converter to create a centralized power system that cleanly drives both the Pi 5 and all 5V peripherals, including the encoders, IMU, and the offloading mechanism's servo motor.

Future work may include improved wheel-slip compensation, refined sensor fusion, expanded waste datasets, more robust object-pose estimation, and route-selection policies calibrated from measured traversal time and gate availability.

References

[1] Open Robotics, ROS 2 Jazzy Jalisco Documentation. https://docs.ros.org/en/jazzy/index.html

[2] Open Robotics, Navigation 2 Documentation. https://navigation.ros.org/

[3] Open Robotics, Gazebo Sim Documentation. https://gazebosim.org/docs

[4] J. Ouma, D. Kabiru, N. Kamau, and L. Karani, Raspberry Pi IMX219 Camera Benchmark and Edge Artificial Intelligence Insights, Robotics Dojo Substack, 2026.

[5] D. Kabiru, ROS 2 Control Architecture and Custom Hardware Interfaces, Robotics Dojo Substack, Sep. 2026.

[6] D. Kabiru, Simulating the Competition Gamefield Development in Gazebo, Robotics Dojo Substack, Sep. 2026.

[7] J. Newans, Building a Mobile Robot Series. https://articulatedrobotics.xyz/