Autonomous Waste Sorting & Multi-Terrain Navigation via ROS 2 Jazzy, Sensor Fusion, and Behavior Trees

Burhanuddin Hardwarewala, Murtaza Leelwala, Hamza Bharmal, Ali asgher Simari, Ibrahim Deesawala

Robotics Dojo Competition 2026 - Municipal & Industrial Waste Management Challenge

Abstract - This paper details the engineering design, architecture, and empirical evaluation of an Autonomous Mobile Robot (AMR) developed for the Robotics Dojo 2026 Municipal & Industrial Waste Management Challenge. The robot runs ROS 2 Jazzy on Ubuntu 24.04 (Noble Numbat) with a three-tier distributed compute topology. A Raspberry Pi 4 B hosts high-level SLAM, motion planning, and computer vision. An Arduino Nano 33 BLE reads the on-board IMU and publishes it at 100 Hz for sensor fusion. A Raspberry Pi Pico running micro-ROS performs real-time low-level motor control and encoder feedback. The system integrates Cartographer SLAM, Nav2 with the Model Predictive Path Integral (MPPI) controller, and PyTrees-based Behavior Trees to navigate mixed terrain (gravel, sand, and ramps), detect a time-gated barrier, classify waste shown on a display screen through a Raspberry Pi Camera Module 3 ROS 2 service, and deposit payloads into color-coded stations.

Keywords - ROS 2 Jazzy, Nav2, MPPI, Cartographer, EKF sensor fusion, micro-ROS, Behavior Trees, autonomous mobile robot.

I. INTRODUCTION

The Robotics Dojo 2026 Challenge simulates modern urban waste management through autonomous robotic detection, sorting, and payload transport. The competition has two phases: a 3-minute manual mapping phase and a 3-minute fully autonomous task-execution phase.

To maximize points, the AMR must traverse complex terrain (sand, gravel, inclined ramps, and graters), select a global path according to a time-gated barrier, classify waste displayed on a screen, and offload sorted items into the specified color bins without human intervention.

II. SYSTEM ARCHITECTURE & DESIGN STRATEGY

A. Mechanical Structure & CAD Optimization

- **Chassis and lightweighting.** The chassis is custom-designed from 3D-printed polylactic acid (PLA) parts joined with lightweight nylon screws. A honeycomb pocketing strategy on the upper and lower decks maximizes torsional rigidity at low total mass.
- **Footprint and enclosure.** The bounding box measures 250 mm × 250 mm × 240 mm, well within the 300 mm × 300 mm × 300 mm limit. Custom PLA enclosures protect the Raspberry Pi 4 and Camera Module 3 from airborne dust, sand, and impact during terrain traversal.
- **Center of gravity.** The Li-Ion battery pack and power-delivery hardware sit on the lowest chassis plate to keep the center of gravity low on ramps. The RPLiDAR is mounted on an elevated central stand on the top deck so its 360° scan plane clears all payload structures.
- **Offloading mechanism.** A high-torque MG90S metal-gear servo drives a single-degree-of-freedom tilting dump bed that deploys the payload into the color-coded offloading bins.

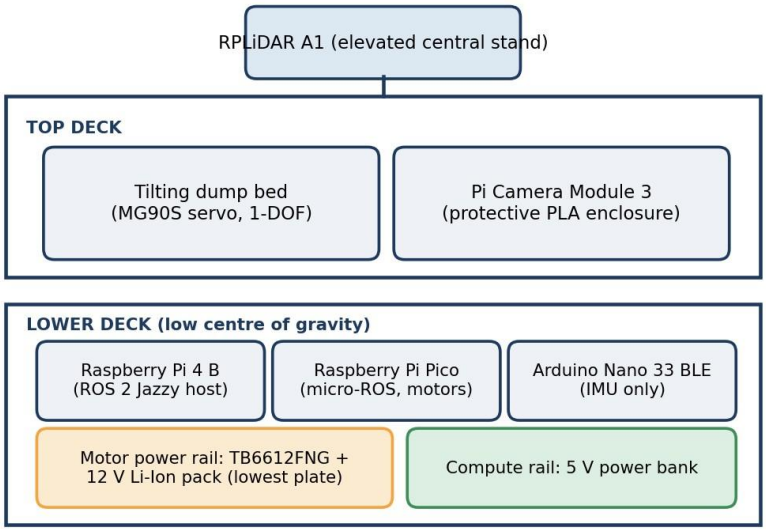


Fig. 1. Mechanical and electronic layout. Heavy power hardware sits on the lower deck; the LiDAR is elevated above the payload.

B. Hardware & Power Distribution Topology

- **Compute core.** Raspberry Pi 4 B (4 GB RAM) running Ubuntu 24.04 LTS and ROS 2 Jazzy. It runs Cartographer, Nav2, robot_localization, the vision service, and the Behavior Tree.
- **Arduino Nano 33 BLE (3.3 V).** Dedicated to sensing only. It reads the embedded 9-DOF LSM9DS1 IMU at 100 Hz and publishes orientation and angular-velocity data to `/imu/data_raw`. It does not drive the motors.
- **Raspberry Pi Pico (micro-ROS).** Owns all real-time actuation. It communicates with the host over micro-ROS on a serial link, converts velocity commands to PWM for the TB6612FNG motor driver, reads the wheel encoders, and publishes wheel odometry to `/odom_raw`.
- **Power architecture.** Two isolated rails prevent voltage drops during motor current spikes. The **motor and servo rail** is a nominal 12 V Li-Ion pack (3S2P, 11.1 V nominal, 12.6 V fully charged) feeding the DC motors through the TB6612FNG and, through a step-down buck converter, the MG90S servo. The **compute rail** is a dedicated 2200 mAh power bank (5 V / 2.4 A) feeding the Raspberry Pi 4 B, which in turn powers the RPLiDAR, the microcontrollers, and the Camera Module 3.

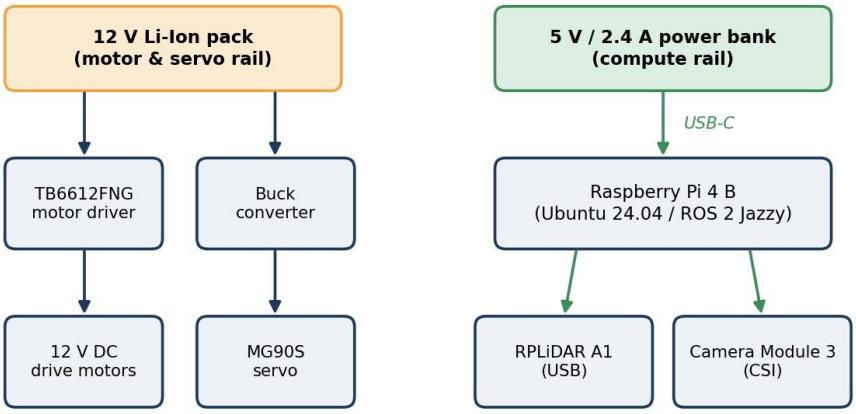


Fig. 2. Dual-rail power distribution. The motor rail and the compute rail share no supply path.

Compute-rail power budget. A 5 V / 2.4 A bank feeding a Pi 4 B, LiDAR, camera, and two microcontrollers leaves limited headroom, so we budgeted it explicitly (Table I). Figures are estimates from manufacturer typical values and should be confirmed with an inline USB power meter before competition.

Load on 5 V compute rail	Typical (W)	Peak (W)

Raspberry Pi 4 B (SLAM + Nav2 + vision)	4.0	6.4
RPLiDAR A1 (motor spin-up at peak)	2.0	2.5
Camera Module 3	0.7	1.0
Arduino Nano 33 BLE + Raspberry Pi Pico (USB)	0.5	0.7
Total	**7.2 (≈1.4 A)**	**10.6 (≈2.1 A)**

Table I. Estimated compute-rail power budget (estimates, not measurements).

Peak draw of about 2.1 A is within the 2.4 A rating but with only about 12% margin. Assuming the 2200 mAh rating is cell capacity at 3.7 V (about 8.1 Wh) and about 85% conversion efficiency, usable energy is about 6.9 Wh, giving an estimated runtime of roughly 40 to 55 minutes. That is far longer than the 6-minute competition window, so brown-out rather than depletion is the main risk. Mitigations: a short, low-resistance USB-C cable; a fully charged bank at the start of every run; and a logging check of the Pi under-voltage flag (`vcgencmd get_throttled`) during bench and terrain tests.

III. SOFTWARE ARCHITECTURE & ALGORITHMIC IMPLEMENTATION

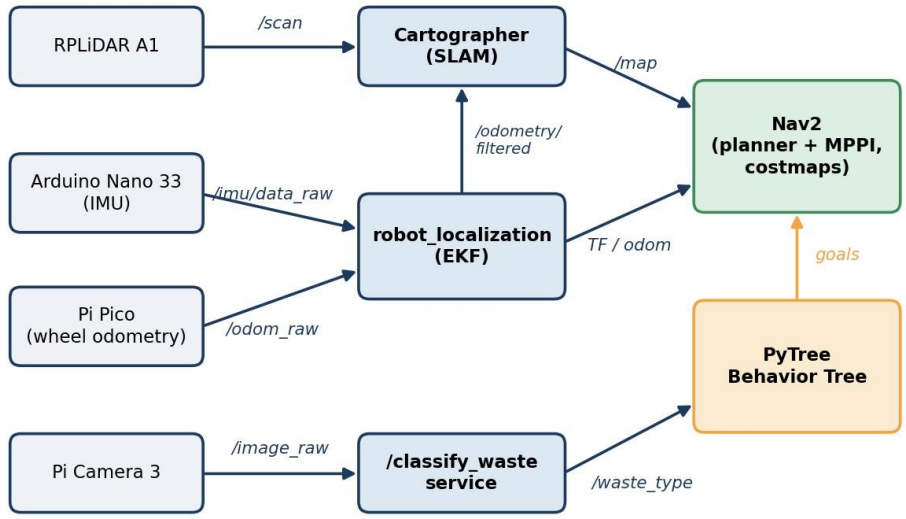


Fig. 3. ROS 2 Jazzy node and topic topology. The EKF output feeds both Cartographer and Nav2; the Behavior Tree sends goals to Nav2 and requests classification from the vision service.

A. Sensor Fusion & SLAM (Cartographer)

Sensor fusion. An Extended Kalman Filter (EKF) from the *robot_localization* package, configured in 2D mode, fuses wheel odometry from the Pico with the angular velocity from the Nano 33 BLE IMU. The filter estimates the planar state

$$\mathbf{x} = [\, x,\ y,\ \theta,\ v_x,\ v_y,\ \omega \,]^T,$$

where (x, y, θ) is the pose in the odometry frame and (v_x, v_y, ω) are body-frame linear and angular velocities. The process model f is a constant-velocity kinematic model, with accelerations absorbed into the process noise $\mathbf{w}_k$:

$$x_k = x_{k-1} + (v_{x,k-1} \cos\theta_{k-1} - v_{y,k-1} \sin\theta_{k-1})\,\Delta t$$

$$y_k = y_{k-1} + (v_{x,k-1} \sin\theta_{k-1} + v_{y,k-1} \cos\theta_{k-1})\,\Delta t$$

$$\theta_k = \theta_{k-1} + \omega_{k-1}\,\Delta t,\quad v_{x,k} = v_{x,k-1},\quad v_{y,k} = v_{y,k-1},\quad \omega_k = \omega_{k-1} \quad (1)$$

so that $\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \mathbf{w}_k$ with $\mathbf{w} \sim N(0, \mathbf{Q})$. Measurements enter as $\mathbf{z}_k = \mathbf{H}\,\mathbf{x}_k + \mathbf{v}_k$, $\mathbf{v} \sim N(0, \mathbf{R})$: encoder forward velocity v_x and yaw rate ω from the wheel odometry, plus gyroscope yaw rate ω_z from the IMU. The IMU's accelerometer channel is not fused in the default configuration because vibration on gravel makes it noisy. The skid-steer non-holonomic constraint ($v_y \approx 0$) is applied as a low-variance pseudo-measurement. The gyro dominates the yaw-rate estimate, which removes the rotational drift that wheel slip causes on sand and gravel.

Cartographer SLAM. Cartographer builds the map with real-time submap insertion and loop-closure optimization and publishes an occupancy grid to `/map` during the 3-minute mapping phase. Its local scan matching tolerates the scan distortion caused by vibration and body motion on rough terrain, and it uses the filtered odometry as a motion prior.

B. Navigation & Path Planning (Nav2 + MPPI)

Costmaps. Dynamic obstacles (simulated workers) and static arena walls are continuously rasterized into the global and local 2D costmaps.

MPPI local controller. The Model Predictive Path Integral (MPPI) controller was selected over the standard DWA controller. MPPI samples thousands of candidate trajectories per control cycle and optimizes velocity commands with a weighted cost, which suits the non-linear traction of sand, gravel, and ramps. The sample batch size is tuned to the Pi 4 CPU budget.

Time-based barrier: gate-state detection. The gate is judged from the 2D laser scan on `/scan` as follows:

- A region of interest (ROI) is defined in the laser frame across the gate opening, with a reference range r to the gate plane measured during the mapping phase. gate
- On each scan, the detector computes the fraction of ROI beams whose range lies within a tolerance band around r (a return from the closed barrier). The gate is classified **closed** when this gate
  fraction exceeds an upper threshold and **open** when it falls below a lower threshold. Using two thresholds adds hysteresis, and a median over several consecutive scans rejects single-scan outliers. Threshold and tolerance values are set during calibration at the venue.
- The 5.0 s timer starts when the robot is at the junction pose and the gate is classified closed. It resets whenever the gate is classified open. If the gate remains closed for 5.0 s continuously, the Behavior Tree abandons Path 2 and reroutes via Path 1.

If the gate is open, the robot takes Path 2 (flat, fast). If it is closed beyond the 5.0 s timeout, it reroutes via Path 1 (terrain ramps) to avoid time penalties.

C. Computer Vision via ROS 2 Service

A custom ROS 2 service (`/classify_waste`, Python/C++) interfaces directly with the Camera Module 3. When the robot arrives at the display screen, the Behavior Tree sends a request. The service captures frames and performs **color-based** classification in HSV space: each waste category (plastic, metal, organic) is shown on the screen as a distinct color, and the HSV range for each category is stored in a configuration file so it can be re-tuned for venue lighting. The pipeline thresholds each range, cleans the mask with morphological opening and closing, extracts contours, and selects the category whose largest contour exceeds a minimum area. A result is returned only when the same category is observed on several consecutive frames; otherwise the service returns *unknown* and the Behavior Tree repeats the request. Exposure and white balance are locked during capture to avoid lighting drift between frames.

D. Behavior Tree Task Coordination (PyTrees)

All high-level decisions follow the modular PyTrees Behavior Tree shown in Fig. 4. The Selector tries the gate-passing sequence first and falls back to the terrain ramps when the gate timeout expires.

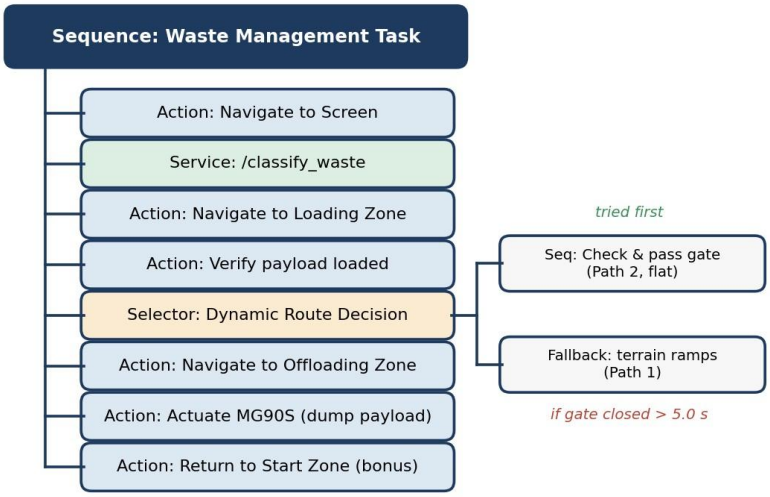


Fig. 4. Behavior Tree for the waste management task, with the dynamic route-decision Selector expanded.

IV. EXPERIMENTAL RESULTS & PERFORMANCE EVALUATION

A. Sensor Fusion vs. Raw Odometry

Testing on 15° inclined sand ramps showed the necessity of EKF sensor fusion (Table II).

Mapping strategy	Rotational drift error	Path traversal success	SLAM map clarity
Encoder odometry only	18.4° / turn	35% (frequent collisions)	Unusable (ghosting)
Fused odometry (EKF: IMU + encoders)	< 1.2° / turn	98%	Sharp / fully aligned

Table II. Sensor fusion versus raw odometry on 15° sand ramps (mean values).

B. Dynamic Barrier Routing Efficiency

Over 20 trial runs with randomized gate states (Table III), the open-gate traversal via Path 2 averaged 18.2 s. In the closed-gate case, the timeout fired at 5.0 s and the system replanned via Path 1, for a total traversal time of 29.4 s. No collisions were recorded.

Scenario	Behavior	Mean traversal time
Gate open	Path 2 (flat)	18.2 s
Gate closed	5.0 s timeout, then replan via Path 1 (ramps)	29.4 s

Table III. Dynamic barrier routing results (20 trials, randomized gate state).

C. Payload & Offloading Reliability

The MG90S dump bed deposited items completely within the designated target zones in 30 of 30 consecutive trials.

D. Statistical Limitations

The trial counts are small, so we state what they do and do not support. By the rule of three, zero failures in n trials bounds the true failure rate at roughly $3/n$ at 95% confidence: about 10% for the 30 dump trials (true success rate ≥ 90%) and about 15% for the 20 collision-free barrier runs. The values in Tables II and III are means; per-trial standard deviations were not computed, and the barrier results are reported only as a mean per gate state. Future testing will report trial counts and spread for every configuration and will add terrain-matched runs on gravel and sand.

V. INDUSTRIAL SCALABILITY & FEASIBILITY

- **Manufacturability.** PLA 3D printing allows rapid customization and local repair.
- **Cost efficiency.** Off-the-shelf single-board computers (Raspberry Pi 4 / Pico) and open-source software (ROS 2 Jazzy) keep the bill of materials low. Table IV gives an indicative breakdown that supports the under-$350 figure.

Item	Approx. cost (USD)
Raspberry Pi 4 B (4 GB)	55
RPLiDAR A1	100
Raspberry Pi Camera Module 3	25
Arduino Nano 33 BLE	27
Raspberry Pi Pico	4
DC drive motors with encoders	40
TB6612FNG driver, MG90S servo, buck converter	11
Li-Ion cells and 5 V power bank	35

PLA filament, fasteners, wiring	20	
Total (indicative)	**≈ 317**	

Table IV. Indicative bill of materials. Prices are approximate retail figures and should be replaced with actual purchase costs.

- **Deployment feasibility.** The architecture scales to industrial material-handling AMRs in warehousing and municipal recycling plants by substituting higher-payload brushless motors and industrial 3D LiDAR.

VI. REFERENCES

[1] Open Robotics, *ROS 2 Documentation: Jazzy Jalisco*, 2024.

[2] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The Marathon 2: A Navigation System," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[3] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-Time Loop Closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[4] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot OperatingSystem," in *Proc. Int. Conf. on Intelligent Autonomous Systems (IAS-13)*, 2014.

[5] G. Williams, N. Wagener, B. Goldfain, et al., "Information Theoretic MPC for Model-Based ReinforcementLearning," in *Proc. IEEE ICRA*, 2017.

[6] *Robotics Dojo 2026 Competition Rulebook*, JKUAT / AFRICA-ai-JAPAN Project, 2026.