ROBOTICS DOJO COMPETITION 2026

Malkiabots

Autonomous Waste-Sorting Robot

Organic · Plastic · Metal

Mercy Murimi · Joy Oluga · Lyn Kamau · Jordana Owilla · Melisa Kinda

Problem Statement

Manual sorting exposes workers to hazardous material and is inconsistent over long shifts. Commercial sorting robots exist, but they are large machines fixed to a conveyor. A single misclassified item also contaminates the stream it is placed in.

Competition Requirements

300 mm — Cube envelope the whole robot must fit in

≤150 g — Waste items of varied shape, up to about 8 cm

3 classes — Organic, plastic and metal, each off-loaded at its own zone

Autonomous — No operator input, over different terrain types

The core difficulty is perception: items are irregular, partly hidden or soiled, and seen under uneven light, yet each must be identified while the robot is moving.

Our Solution

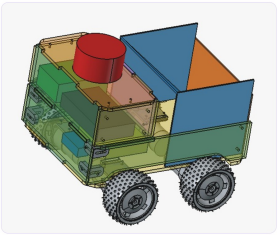
A four-wheel-drive robot that maps the field with LiDAR and navigates with Nav2. Camera images are streamed to a laptop, which classifies each item. The robot then drives to the matching zone, where a servo lifts a gate and the item slides down a 43.5° ramp.

Task sequence (behaviour tree)

1. **Search** — Patrol the mapped field with Nav2
2. **Detect** — Pi Camera spots a candidate item
3. **Classify** — Model labels organic, plastic or metal
4. **Select zone** — Organic, plastic or metal target pose
5. **Transport** — Nav2 drives there, avoiding obstacles
6. **Off-load** — Servo lifts the gate; item slides down

Low confidence → re-scan
Path blocked → re-plan, recover

Robot Design



254 × 271 × 228 mm overall (margins of 46, 29 and 72 mm)

2.97 kg estimated, with a 150 g payload

4 × 110 RPM JGB37-520 geared motors with encoders, 65 mm treaded wheels

74 Wh 4 × 18650 Li-ion (14.8 V) for motors; 27 W bank for the Pi

Base: motors and battery, kept low. **Middle (front half):** Pi 5 and power bank. **Upper deck:** RPLiDAR at 176 mm for an unblocked 360° scan. **Rear bay:** the off-loading ramp. Laser-cut tab-and-slot panels assemble and swap quickly.

System Architecture

- **RPLiDAR** — 360° scan · /scan
- **Pi Camera** — 640 × 480 · 15 fps
- **Laptop (CPU)** — YOLO26n detector ⇄ MJPEG over Wi-Fi
- **Teleop / goals** — keyboard over SSH

Raspberry Pi 5 · ROS 2 Jazzy
High-level thinking: maps, plans, decides

`SLAM` `Nav2` `EKF (robot_localization)` `Behaviour tree` `MJPEG camera stream` `/waste/detections` `base_node bridge` `sllidar_ros2`

Teleop and Nav2 both publish `/cmd_vel`, so the lower layers work the same in both modes.

`/cmd_vel` → `M 1 z · 10 Hz`
← telemetry 20 Hz · USB serial

Arduino Nano
Real-time control, no blocking delays

`PI speed loop · 50 Hz` `4 encoders` `MPU6050 IMU` `Gate servo` `500 ms watchdog`

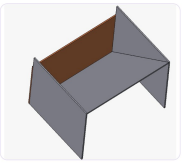
- **2 × motor driver** — front pair · rear pair
- **4 × motor** — skid-steer, left / right sides

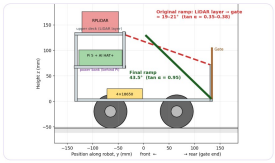
Two isolated supplies
Li-ion pack → motor drivers; 27 W USB-C PD bank → Pi and its USB devices. No buck converter; all grounds joined.

Sensor fusion
EKF fuses wheel speed and gyro turn rate (not the drifting yaw angle) into `/odometry/filtered`.

Readable protocol
Plain text lines at 115200 baud: debuggable from any terminal, with bandwidth to spare.

Off-loading Mechanism





Redesign: the first ramp, from the LiDAR deck to the rear gate, sloped only 19–21° (tan α ≈ 0.37), too shallow for cardboard or plastic. Mounting it on the chassis gave 43.5°. **Tested on the robot:** items slide off cleanly when the gate lifts.

Ramp and Gate Calculations

Slide-off condition
`tan 43.5° = 0.95 > μs`, a 1.9× margin for μs up to 0.5. An item clears the 80 mm opening in about `0.22 s`.

Gate actuation
38 g gate + waste friction: `F = 0.68 N`. With a 42.5 mm crank, the servo needs only `0.29 kg·cm` (0.88 with a 3× safety factor).

Why this design
One actuator, very low power, and few parts that can jam. It also fits inside the 300 mm limit.

Locomotion and Stability

10.5 N — needed on a 15° slope

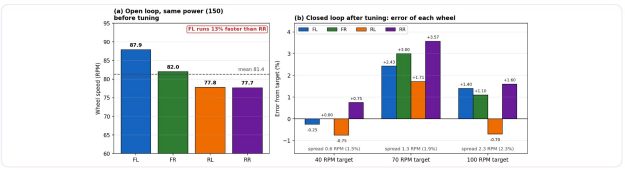
17.1 N — tyre grip limit (μ = 0.6)

47.1 N — motors, 4 wheels (4.5× margin)

Because grip runs out long before torque, we chose large treaded 65 mm wheels. The spare torque covers skid-steer turning on high-friction ground.

Tip-over angles, θ = arctan(b/h)
Climbing **49°** · sideways **51°** · descending nose-first **18°** (22° loaded). The centre of mass sits forward with the electronics, so steep descents are the one case to avoid.

Motor Speed Tuning



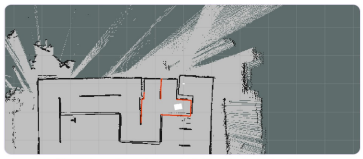
≤ 0.8% — every wheel at 40 RPM

≤ 2.3% — wheel-to-wheel at 100 RPM

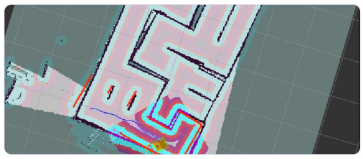
−6° to −3° — yaw over 0.8 m, gyro heading hold

Per-wheel feed-forward trims plus a PI loop (no D: 20 ms encoder speed is too noisy). Speed capped at 91 RPM / 0.31 m/s to keep headroom.

Mapping and Navigation



Occupancy grid from the RPLiDAR. Live scan (red) sits on the mapped walls.



Costmaps: walls inflated for the robot's footprint; local costmap around the robot.

Autonomous navigation achieved: SLAM supplies map → odom, the EKF supplies odom → base_link, and Nav2 plans across the costmap, re-plans as it moves, and recovers (clear, spin, back up, wait) when blocked.

Waste Classification



A plastic bottle in a **blue** box: **plastic · 63%**.



An organic item in a **green** box: **organic · 57%**.

3 classes — blue plastic · green organic · grey metal

63% — confidence on a plastic bottle

57% — confidence on an organic item

15–45× — less Wi-Fi bandwidth with MJPEG than raw video

The Pi streams compressed MJPEG frames over Wi-Fi to a laptop, where a YOLO26 nano model running on the CPU boxes each item with a class and confidence and publishes the annotated frame on `/waste/image` for RViz. Low-confidence labels send the behaviour tree back to re-scan before the robot commits to a zone.

Simulation



Early URDF from the CAD meshes, loaded cleanly in RViz and driven in Gazebo on a model of the game field.

The ROS package ran in Docker against a **simulated Nano**, with unit-tested odometry, so the bridge worked on the Pi almost first time.

Safety Features

- **500 ms watchdog** stops the motors if commands stop
- **Stall cutoff** after 600 ms of no motion
- **I²C timeout:** a lost IMU can't freeze the Nano
- **Power cap & ramp:** ≤200/255, ≤12 per 20 ms
- **Encoder guard** falls back to feed-forward
- **Auto-reconnect** if the USB link drops

Lessons Learned

- **Calculate before building.** Checking the slide-off condition early would have caught the shallow first ramp.
- **Keep settings in one place.** The Nano resets when the port opens, so settings are now stored in ROS and sent on every connection.
- **Use the logs.** A wheel at full power with zero speed in the log showed a loose encoder.
- **Check hardware support.** The Pi 5 has no H.264 encoder, so we stream MJPEG instead.

Project Status

- [x] Drive base & tuning
- [x] SLAM mapping
- [x] Behaviour tree
- [x] Ramp off-load test
- [x] IMU & EKF
- [x] Nav2 autonomy
- [x] Waste classifier
- [x] Odometry calibration

All tested on the robot