Tetsuzakura Knights Technical Paper: Robotics Dojo Competition 2026

John Khaemba, Prudence Njoroge, Allen Wachio, Joshua Njau, Alvin Gamba
Tetsuzakura Knights
Robotics Dojo Competition 2026

***Abstract*—Rapid urbanization has made municipal and industrial waste handling a pressing engineering challenge, and manual sorting is slow, costly, and hazardous to workers. This paper presents a compact autonomous mobile robot developed for the Robotics Dojo 2026 waste-management challenge. The robot operates in two phases. In a manually driven mapping phase, a 2D LIDAR-based SLAM pipeline builds a map of the game field. In the subsequent fully autonomous phase, the robot localizes against the map and navigates to a display screen to classify waste shown as plastic, metal, or organic. To achieve real-time, low-latency classification, a YOLO11-nano model was trained and successfully quantized for edge deployment on the Sony IMX500 architecture. The robot then drives to the matching loading station, crosses gravel, sand, and ramp terrain while avoiding obstacles, and identifies the colour-coded offloading station to deposit the item. Robust autonomous navigation is achieved via the ROS 2 Navigation Stack (Nav2), utilizing AMCL for localization and a Model Predictive Path Integral (MPPI) controller for local trajectory tracking. During testing, the system achieved high classification accuracy and reliable end-to-end task completion. The results show that integrating SLAM, edge AI perception, and advanced decision-making on a small, power-limited platform can support reliable autonomous waste sorting, with design implications for scalable, low-cost development.**

***Index Terms*—ROS2, SLAM, LiDAR, Edge AI, YOLO11, Nav2, Waste Classification**

I. Introduction

Effective waste management is among the defining challenges of modern cities. Growing volumes of mixed waste place heavy demands on collection and sorting infrastructure, and manual segregation exposes workers to injury, contamination, and inconsistent accuracy [1]. Autonomous robots offer a route to safer, faster, and more consistent handling, but real facilities and collection sites demand more than a single capability. A useful robot must perceive its surroundings, recognize what it is handling, move reliably over uneven ground, share space safely with people, and make sensible decisions under uncertainty.

The Robotics Dojo 2026 challenge concentrates these demands into a single game field task. Within a 3-minute autonomous phase, preceded by a 3-minute manual mapping phase, a robot confined to a 30 cm cube envelope and a 24 V power limit must complete a chain of dependent tasks. It must detect the waste type shown on a display, collect the corresponding load, traverse gravel, sand, and ramp terrain, avoid static cylinders and dynamic obstacles representing workers, decide between a hard but always-open route and a faster gated route, and deposit the load in the correct colour-coded station before reaching the finish. Because the tasks run in sequence, an error in any stage, such as a misclassification, a localization drift, or a poor gate decision, propagates downstream, so reliability across the whole pipeline matters more than excellence in any single module.

Prior work has addressed these problems largely in isolation: 2D LIDAR SLAM and grid mapping [2], [3], vision-based waste classification [4], and navigation stacks with local obstacle avoidance [5]. Integrating them on a small, power-constrained platform that must also climb ramps, handle loose terrain, and meet a hard time limit remains a significant engineering problem. The design must trade off sensor payload against the size limit, drivetrain torque against battery voltage, and inference accuracy against onboard compute and latency.

To meet these requirements, our team developed a four-wheeled autonomous robot with compact dimensions of $300 \times 240 \times 238$ mm, optimized for maneuverability in the dynamic game field. Each wheel is powered by an independent 12 V DC motor, providing enhanced traction and mobility across rough terrain.

The perception system integrates an RPLiDAR sensor for Simultaneous Localization and Mapping (SLAM), enabling real-time localization in changing environments [6]. An IMU provides diagnostic feedback for drift detection, immobilization monitoring, and LiDAR frame filtering to ensure map integrity [7]. A color sensor mounted near the conveyor belt identifies the color of transported loads; this information is cross-referenced with input from a forward-facing Pi Camera to locate designated deposition zones [8]. Beyond object handling, the Pi Camera supports waste detection by running an OpenCV pipeline with a pre-trained classification model, allowing the robot to distinguish between metallic, organic, and plastic waste at the start of operation [9].

The robot supports two operational modes: teleoperation for subsystem debugging and validation, and autonomy for competition tasks [10]. ROS2 middleware coordinates perception, planning, and control nodes, while the Nav2 stack enables dynamic path planning and obstacle avoidance. Object handling is achieved via a 28BYJ-48 stepper-motor-driven conveyor belt for load collection and deposition, complemented by a servo-actuated tipper mechanism that enables self-recovery if the robot becomes stuck. The CAD drawing for the tipper is illustrated in Fig. 1.

The contributions of this work include: (1) the develop-

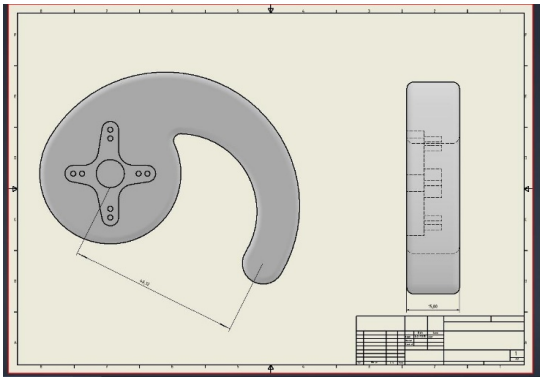


Fig. 1. CAD drawing for the tipper.

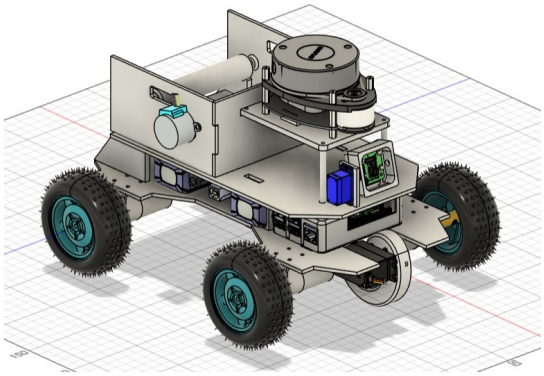


Fig. 2. CAD model of the robot showing mechanical layout, conveyor system, and LiDAR placement.

ment of a compact, low-cost autonomous robot tailored for off-road operation, (2) the integration of ROS2 middleware with LiDAR-based mapping, multi-sensor fusion, and waste detection analysis, (3) the successful edge deployment of a quantized YOLO11-nano model for real-time waste classification, (4) the implementation of an advanced ROS 2 Nav2 configuration utilizing MPPI for dynamic path-planning resilience, and (5) the design of a conveyor-based object handling and deposition system within a constrained 30 cm cubic form factor.

II. Design Strategy

The design strategy of the proposed robotic vehicle, as seen in Fig. 2, was guided by competition constraints and mission objectives. The robot was required to fit within a $300 \times 300 \times 250$ mm bounding box, while being capable of traversing rough terrain, mapping the dynamic game field, and handling objects based on color recognition. These requirements necessitated a compact yet robust design approach that balanced mechanical strength, sensing accuracy, and control flexibility.

A modular design partitioned the system into mechanical, electrical, and computational subsystems. The vehicle employs four independently driven off-road wheels to maximize traction and maneuverability on uneven terrain. Object handling is performed by a 28BYJ-48 stepper-motor-driven conveyor belt that reliably transports and deposits loads; a high torque servo-actuated tipper provides recovery capability by lifting the front wheels to extricate the robot if it becomes immobilized.

Vision is performed by a forward-facing Pi Camera housed in a servo-actuated casing. A color sensor mounted at the conveyor interface inspects each load's color; the Raspberry Pi cross-references this reading with the Pi Camera's detected zone to route loads to their designated areas. Sensor fusion underpins navigation and data integrity: a LiDAR unit mounted on the mast provides primary object detection and SLAM for global mapping. An onboard IMU mounted at the bottom center of the robot monitors vehicle orientation and detects drifting or immobilization—feedback that is used to flag or discard LiDAR frames that could corrupt the map.

Finally, trade-offs were carefully considered in power management, weight distribution, and wiring complexity. Lithium-based batteries combined with a Battery Management System (BMS) were selected to provide sufficient endurance without exceeding the mass limitations. The wiring harness was designed to accommodate multiple actuators and sensors, acknowledging the challenge of increased complexity.

III. Vehicle Design

A. Mechanical Design

The robot was designed within the competition-imposed volume constraint of $300 \times 300 \times 250$ mm, which guided many early design decisions. CAD modeling (Fig. 2) was employed to optimize the allocation of internal space, balancing the requirements for electronic subsystems, actuation mechanisms, and sufficient ground clearance to traverse uneven terrain.

The locomotion system employs off-road wheels, chosen after testing multiple alternatives to ensure traction and stability on rough surfaces. These wheels are each driven by 170 rpm 12 V DC motors. Early prototypes with smaller, smooth wheels experienced excessive slippage on the competition terrain, underscoring the necessity of the final selection. The platform is a four-wheel skid-steer chassis. Each wheel is driven by a brushed DC gear motor with a quadrature encoder, each through one channel of an H-bridge driver, all controlled by an ESP32-S3.

The object handling system consists of a stepper-motor-driven conveyor belt that deposits the load. A LiDAR sensor was mounted on the top platform to enable 360° field scanning.

The tipper, illustrated in Fig. 3, payload mechanism was actuated using a 28BYJ-48 unipolar stepper motor coupled with a ULN2003 driver board. This motor was chosen due to its fine step resolution (≈4096 steps/rev with internal gearing), compact size, and sufficient torque output (∼0.03–0.04 Nm), which makes it suitable for precise angular control of the tipping motion. While its maximum safe speed is limited to ∼15 RPM, the application prioritizes controlled motion over rapid actuation, making the trade-off acceptable. This configu-

ration ensures stable, incremental actuation of the tipper while minimizing power consumption from the 5 V supply line.

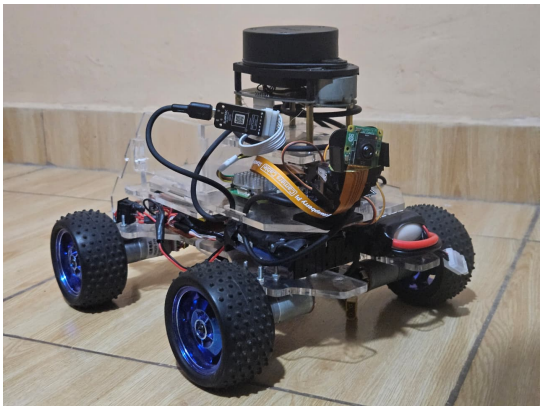


Fig. 3. Side view of the Tipper attached to the robot.

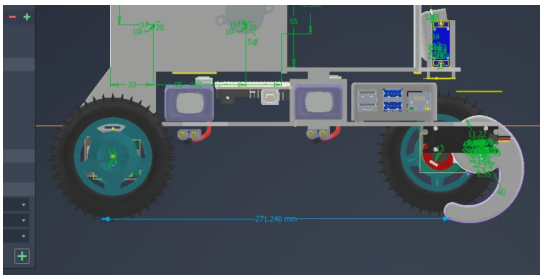


Fig. 4. CAD geometry of the tipper and wheelbase distance.

The following schematic shows the torque calculation setup, including the wheelbase $L_{\text{wheelbase}}$, center of mass r_{com}, and tipper arm r_{arm} at different rotation angles θ.

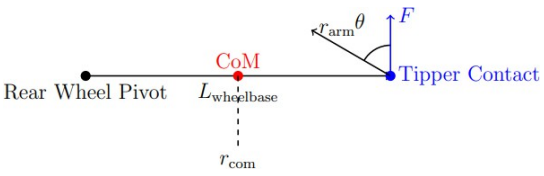


Fig. 5. Mathematical torque diagram: wheelbase $L_{\text{wheelbase}}$, center of mass r_{com}, tipper arm r_{arm}, and lift angle θ of the servo.

Fig. 6 illustrates the Robot Assembly.

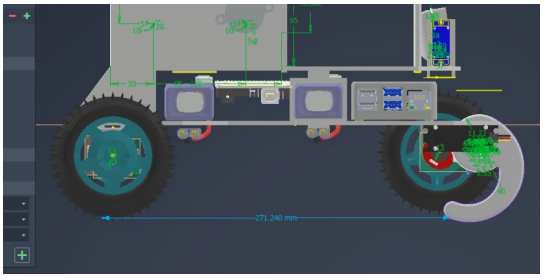


Fig. 6. Robot Assembly side view.

B. Electrical and Power System

The robot uses a two-tier computing architecture that separates real-time actuation from computationally intensive perception. An ESP32-S3 microcontroller (ESP32-S3-DevKitC-1, 16 MB flash, octal PSRAM; Arduino framework, PlatformIO) executes everything that needs deterministic timing: four wheel-speed loops at 20 Hz, quadrature encoder counting, servo and stepper actuation, inertial and colour sensing, and attitude and pose estimation.

A host computer (Raspberry Pi) on the robot runs ROS 2 Jazzy with LiDAR SLAM, Nav2, vision processing, and velocity-command arbitration. This partitioning keeps wheel control independent of perception load. Each drive wheel is a brushed DC gearmotor with a quadrature encoder (748 counts per wheel revolution) and one H-bridge channel, where IN1 carries a 20 kHz PWM signal and IN2 sets direction. Per-wheel polarity constants ensure that a positive speed always means forward motion. An MPU6050 inertial sensor (0x68) and a TCS34725 colour sensor (0x29) share one I^2C bus, and a tilt servo, a dump servo, and a 28BYJ-48 conveyor stepper complete the actuator set.

The ESP32 connects to ROS 2 as a micro-ROS client over a 460,800-baud serial link, appearing as a single node, `base_controller`, with five best-effort publishers (wheel ticks, IMU, filtered odometry, colour, self-check status) and six reliable subscriptions (velocity command, tilt, dump, conveyor, self-check trigger, heading-hold enable). Best-effort sensor delivery lets a congested link drop a sample instead of stalling the controller. The estimated payload of about 21.8 kB/s uses roughly 47% of the 46.1 kB/s link capacity. A four-state connection machine pings the host agent every 500 ms, creates the ROS entities on contact, and destroys and recreates them after four consecutive failed pings. The robot therefore recovers from an agent restart or cable disconnect without operator action. Fig. 7 illustrates the micro-ROS stacks.

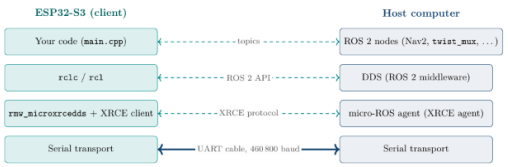


Fig. 7. The micro-ROS stacks. Each layer on the ESP32 has a partner on the host; only the bottom pair is a physical link.

Accurate velocity tracking underlies both odometry and navigation. Each wheel is modelled as a first-order-plus-dead-time plant with a dead-band, $G(s) = \frac{Ke^{-\theta s}}{\tau s+1}$, where K is the steady-state gain, τ the time constant, θ the effective delay, and u_0 the minimum PWM that produces motion. PI gains follow the SIMC rules, $K_p = \frac{\tau}{K(\tau_c+\theta)}$ and $\tau_I = \min\{\tau, 4(\tau_c+\theta)\}$, with one closed-loop time constant τ_c shared by all wheels and directions so that the four wheels respond equally fast and do not produce transient yaw error.

Fig. 8 shows the schematic for the electrical circuit.

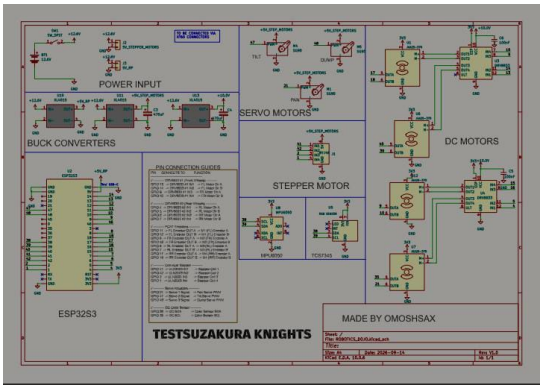


Fig. 8. Electrical Circuit Schematic.

IV. Edge AI and Waste Classification

The integration of Artificial Intelligence (AI) at the edge, specifically within smart camera sensors, eliminates the latency and bandwidth constraints associated with cloud-based inference. The Sony IMX500 is a pioneering intelligent vision sensor that embeds AI processing directly into the image sensor pipeline [31]. However, deploying models on such hardware requires rigorous quantization and format conversion, often necessitating architectural downscaling.

A. Dataset Composition and Analysis

The dataset comprised 3,868 annotated images, partitioned into training and validation sets [29]. The data was categorized into three distinct classes. As detailed in Table I, the dataset exhibits a notable class imbalance.

TABLE I
Dataset Class Distribution

Class	Image Count	Percentage (%)
Metal	2,750	42.6
Plastic	2,119	32.8
Organic	1,794	27.8
Total	**3,868**	**100.0**

Spatial analysis of the bounding boxes (Fig. 9) indicates that objects are predominantly centered in the frame. The aspect ratio distribution reveals a wide variance in object scales, necessitating a robust multi-scale detection architecture.

B. Model Architecture and Training Configuration

The initial baseline model was trained using the YOLO11-small (YOLO11s) architecture. However, compatibility constraints with the Sony IMX500 export pipeline mandated a switch to the YOLO11-nano (YOLO11n) architecture, as it is the only variant currently supported for `.rpk` compilation by the Ultralytics export tools [30].

The model was retrained from scratch using the following hyperparameters:

- **Architecture:** YOLO11n (Nano)

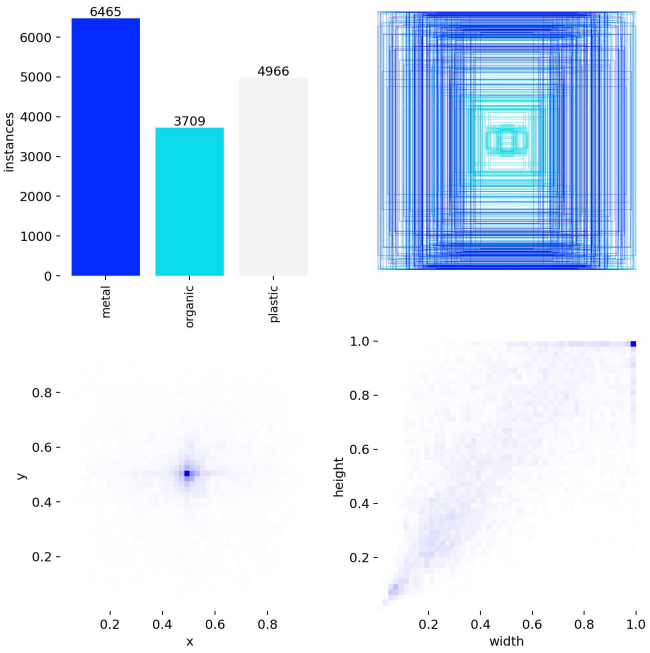


Fig. 9. Dataset class distribution and bounding box spatial/aspect ratio analysis.

- **Epochs:** 45
- **Input Resolution:** 640 × 640 pixels
- **Optimizer:** AdamW (default YOLO11 configuration)

C. Edge Deployment and Quantization Pipeline

The conversion of the trained PyTorch (`.pt`) model to the IMX500 format involved the Sony Model Compression Toolkit (MCT) [32]. The command `export("imx")` initiates Post-Training Quantization (PTQ), converting 32-bit floating-point weights to 8-bit integers (INT8) to optimize for the IMX500's neural processing unit.

The PTQ process requires a calibration dataset to map the floating-point activation ranges to integer ranges. The script attempts to load the entire validation set (∼589 images) into the host system's RAM. This operation consistently caused Out-Of-Memory (OOM) crashes on the Google Colab T4 instance, which is restricted to 12.7 GB of system RAM. To circumvent this, the calibration dataset was aggressively downsampled to 15% of its original size (∼54 images). While the Sony MCT documentation recommends a minimum of 300 images for statistically significant quantization calibration [32], this reduction was the only viable method to complete the export within the available hardware constraints. The process ultimately succeeded, generating the required intermediary files: `labels.txt`, `packerOut.zip`, `dnnparams.xml`, `.onnx`, `.pbtxt`, and `memoryreport.json`.

D. Experimental Results

The training process over 45 epochs demonstrated healthy convergence without significant overfitting. The Mean Average Precision (mAP@0.5) rose sharply in the first 20 epochs and

plateaued at ∼0.765 by epoch 35, confirming that 45 epochs was sufficient for full convergence (Fig. 10).

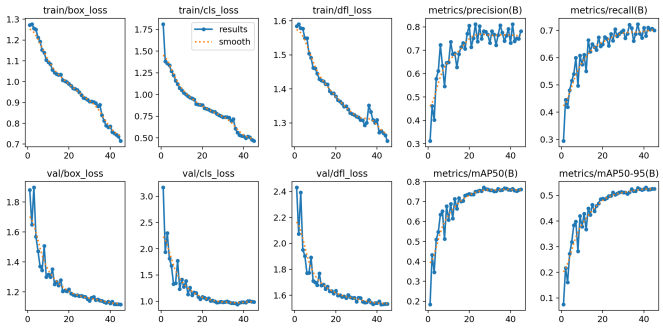


Fig. 10. Training and Validation Loss/Metric Curves.

A deep dive into class-specific Average Precision (AP) reveals a direct correlation with the dataset imbalance detailed in Table I. The ”Metal” class achieved an AP@0.5 of 0.904 (Excellent), ”Plastic” achieved 0.823 (Good), and ”Organic” suffered from the lowest AP at 0.565 (Fair). Organic waste is inherently highly variable in shape, texture, and color, making it the most difficult class to learn, a challenge exacerbated by its minority representation.

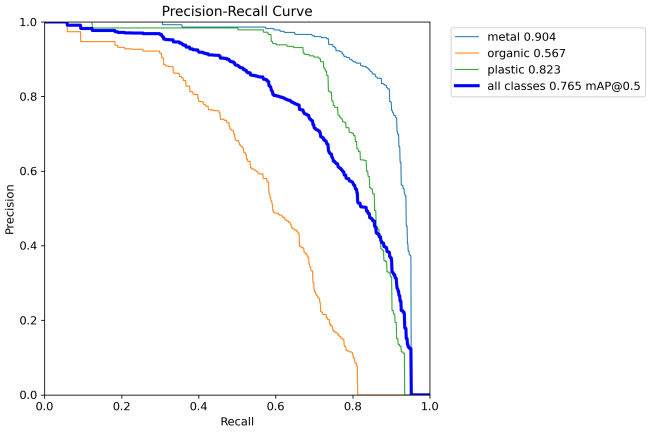


Fig. 11. Precision-Recall curves for individual classes and aggregate mAP.

Analyzing the relationship between confidence thresholds and performance metrics is critical for setting deployment parameters. The optimal F1 score for all classes is 0.74, achieved at a confidence threshold of 0.487. The model achieves 100% precision at a confidence threshold of 0.962, indicating that its highest-confidence predictions are entirely reliable.

The normalized confusion matrix (Fig. 13) provides critical insights into the model’s failure modes. The most significant error is the model’s tendency to hallucinate objects in empty backgrounds, with 56% of background instances incorrectly predicted as ”Organic.” This suggests the model is overly sensitive to background textures and lighting reflections that mimic these materials.

Visual inspection of the prediction outputs (Figs. 14, 15, and 16) corroborates the quantitative metrics. The model exhibits

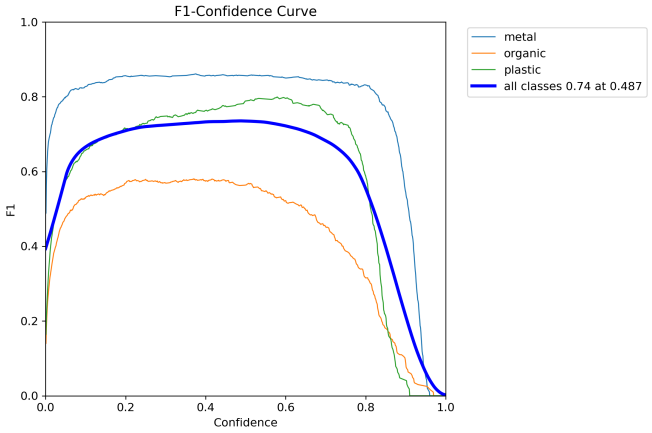


Fig. 12. F1 score relative to confidence thresholds.

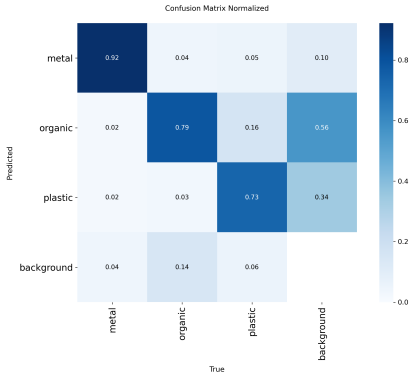


Fig. 13. Normalized Confusion Matrix (Predicted vs. True).

robust bounding box tightness for metal cans and distinct plastic bottles.

V. Control and Software Architecture

A. Software Architecture and micro-ROS

The robot supports two operational modes: teleoperation, which facilitates early debugging and validation of subsystems, and autonomy, which integrates mapping, perception, and actuation for competition tasks. Teleoperation is available via a gaming pad and a remote GUI (web/ROS2-based), enabling live camera feed, manual drive, and on-screen diagnostics during testing.

ROS2 serves as the middleware framework, enabling modular communication between nodes handling LiDAR-based SLAM, motor control, vision, planner/behavior, and sensor fusion. The autonomous workflow consists of mapping the dynamic game field with LiDAR, detecting and classifying objects using the color sensor and Pi Camera, and executing path-planning strategies for collection and deposition.

The robot’s kinematic model was derived directly from the Fusion 360 assembly rather than being reconstructed by hand. Components were mapped to URDF links and assembly joints to URDF joints through the `fusion2urdf-ros2`

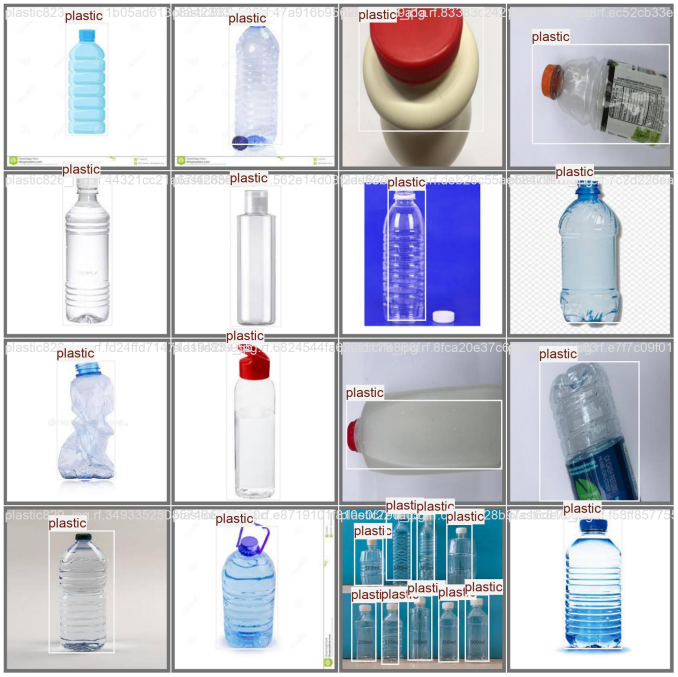


Fig. 14. Sample qualitative predictions (Batch 0).



Fig. 15. Sample qualitative predictions (Batch 1).



Fig. 16. Sample qualitative predictions (Batch 2).

exporter. Occupancy-grid mapping used `slam_toolbox` in asynchronous online mode, with a simulated 360-sample planar lidar at 10 Hz, scan matching against the incoming scans, and a sparse pose-graph backend solved with Ceres.

B. ROS 2 Navigation Stack (Nav2) Configuration

Navigation is implemented through the ROS 2 Navigation Stack (Nav2), which represents a significant architectural evolution from its ROS 1 predecessor, `move_base` [5]. By leveraging Behavior Trees, lifecycle nodes, and a highly modular plugin architecture, Nav2 provides a robust framework for autonomous mobile robot (AMR) navigation [37].

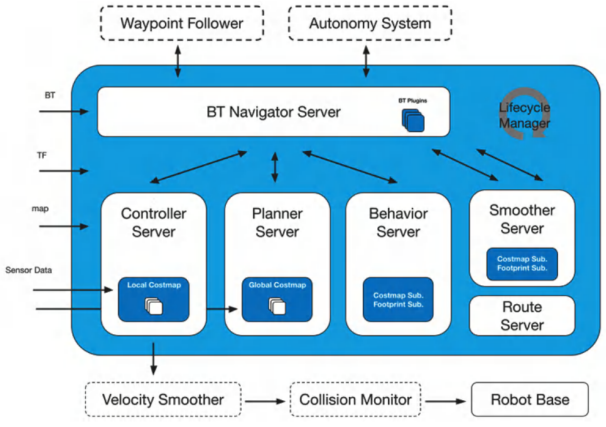


Fig. 17. High-level architecture of the Nav2 framework.

The system employs the Adaptive Monte Carlo Localization (AMCL) algorithm to estimate the robot's pose within a known

map [35]. The configuration utilizes a `likelihood_field` laser model with a maximum of 2000 particles and a `DifferentialMotionModel`. Global path planning is handled by the `NavfnPlanner`, which implements the A* search algorithm on a 2D grid. The global costmap integrates Static, Obstacle, and Inflation layers to ensure the robot maintains a safe distance from global obstacles.

A notable feature of this configuration is the use of the Model Predictive Path Integral (MPPI) controller for local trajectory tracking, replacing traditional Dynamic Window Approach (DWA) or Timed Elastic Band (TEB) controllers [34]. MPPI is a stochastic optimal control algorithm that samples control sequences and evaluates them using a cost function. Key MPPI parameters include a batch size of 100 trajectories over 15 time steps, configured for `DiffDrive` with strict velocity and acceleration limits. The local costmap uniquely employs a `VoxelLayer` instead of a standard 2D obstacle layer. By utilizing 16 voxels in the Z-axis up to a height of 2.0 meters, the system can better handle complex sensor data and prevent false obstacle clearing from overhead structures.

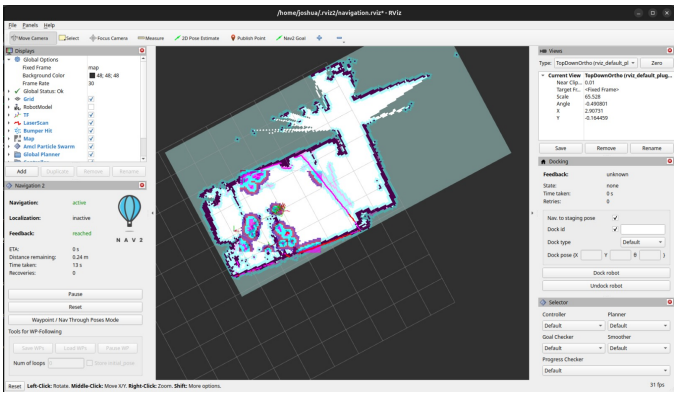


Fig. 18. Visualization of the local costmap's voxel layer and the MPPI controller's sampled trajectories.

To ensure operational safety, the configuration includes a Collision Monitor operating in an `approach` mode using the robot's footprint, and a Velocity Smoother that applies open-loop smoothing to velocity commands, ensuring acceleration limits are strictly respected.

C. Simulation and Testing

To facilitate development and testing without physical hardware, the configuration includes a `Loopback Simulator`. This node simulates the robot's kinematics, odometry, and LiDAR scans based on the commanded velocities. It is configured with a scan range of 30.0 meters and an angular range of $\pm\pi$, providing a high-fidelity virtual environment for tuning the MPPI controller and validating the Behavior Trees.

VI. Conclusion

This paper presented the design, development, and integration of a compact autonomous mobile robot for the Robotics Dojo 2026 waste-management challenge. The mechanical and electrical design prioritized robustness and maneuverability within strict size and power constraints. To achieve reliable waste classification, a YOLO11-nano model was trained and successfully quantized for edge deployment on the Sony IMX500 architecture, overcoming significant memory constraints during Post-Training Quantization. Furthermore, the system leverages the ROS 2 Navigation Stack (Nav2), utilizing AMCL for localization and an advanced MPPI controller for local trajectory tracking, ensuring robust navigation in dynamic environments. The successful integration of SLAM, edge AI perception, and advanced decision-making demonstrates that reliable end-to-end autonomous waste sorting is achievable on a small, power-limited platform, providing a scalable blueprint for future low-cost robotic waste management solutions.

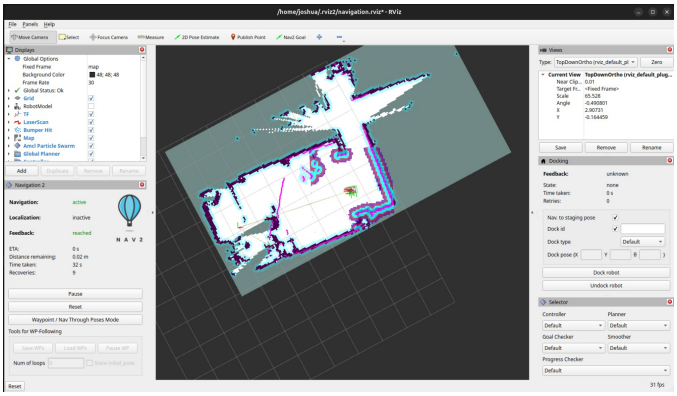


Fig. 19. The loopback simulator environment demonstrating the robot's virtual LiDAR scan footprint and simulated odometry generation.

References

[1] S. Kaza, L. Yao, P. Bhada-Tata, and F. Van Woerden, *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*. Washington, DC, USA: World Bank, 2018.

[2] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[3] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[4] G. Yang and M. Thung, "Classification of trash for recyclability status," Stanford CS229 project report, 2016.

[5] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The Marathon 2: A Navigation System," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020, pp. 11221–11228.

[6] M. J. Mataric, "Designing emergent behaviors: From local interactions to collective intelligence," in *Proceedings of the International Conference on Simulation of Adaptive Behavior*, 1991, pp. 432–441.

[7] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006, doi: 10.1109/MRA.2006.1638022.

[8] RoboCup Federation, "RoboCup: International Competitions and Conferences on Robotics and AI," 2025. [Online]. Available: https://www.robocup.org

[9] K. Nonaka, T. Hasegawa, and Y. Kanayama, "A distributed control architecture for intelligent mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 1, pp. 48–55, Feb. 2004, doi: 10.1109/TIE.2003.822042.

[10] M. Chen, Y. Zhang, and V. Leung, "Battery management system and SOC development for electrical vehicles," *IEEE Transactions on Vehicular Technology*, vol. 60.

[11] ROS 2 Tutorials — Husarion. (n.d.). https://husarion.com/tutorials/ros2-tutorials/ros2/
[12] C. Wambua, "Getting Started with Gazebo Fortress: Installation, Setup, and ROS 2 Integration," *Robotics Dojo*, Aug. 20, 2025. https://roboticsdojo.substack.com/p/getting-started-with-gazebo-fortress
[13] V. Rotich, "Introduction to worlds in Robotics Simulation," *Robotics Dojo*, Aug. 20, 2025. https://roboticsdojo.substack.com/p/introduction-to-worlds-in-robotics
[14] R. Olumo, "RQT setup in ROS2 Humble," *Robotics Dojo*, Aug. 20, 2025. https://roboticsdojo.substack.com/p/rqt-setup-in-ros2-humble
[15] M. M. Farida, "Troubleshooting tips when setting up the RPLIDAR," *Robotics Dojo*, Aug. 20, 2025. https://roboticsdojo.substack.com/p/troubleshooting-tips-when-setting
[16] M. M. Farida, "Understanding URDF: building the blueprint of your robot," *Robotics Dojo*, Aug. 20, 2025. https://roboticsdojo.substack.com/p/understanding-urdf-building-the-blueprint
[17] S. Kimani, "Getting Started with RViz in ROS 2 Humble," *Robotics Dojo*, Aug. 20, 2025. https://roboticsdojo.substack.com/p/getting-started-with-rviz-in-ros
[18] F. Opiyo, "INTRODUCTION TO SIMULTANEOUS LOCALIZATION AND MAPPING," *Robotics Dojo*, Aug. 20, 2025. https://roboticsdojo.substack.com/p/introduction-to-simultaneous-localization
[19] "BEST Arduino Mega 2560 MICROCONTROLLER — BluePack Nova," bluepackventures.com, Sep. 11, 2025. https://bluepacknova.com/product/arduino-mega-2560-microcontroller/
[20] "Off-Road Wheels - 85x38mm," Pixel Electric Company Limited, 2025. https://www.pixelelectric.com/products/robots/wheels/rc-wheels/off-road-wheels-85x38mm/
[21] "DRV8833 TB6612FNG Step Motor Driver Module — BluePack Nova," bluepackventures.com, Sep. 11, 2025. https://bluepacknova.com/product/drv8833-tb6612fng-step-motor-driver-module/
[22] "DC Motor 12V 200rpm with Encoder Module," Pixel Electric Company Limited, 2025. https://www.pixelelectric.com/more-categories/motors-drivers-actuator/brushed-dc-motor/geared-motor/dc-motor-12v-200rpm-with-encoder-module/
[23] "MG996R (Metal Gear) High Torque Servo Motor — BluePack Nova," bluepackventures.com, Sep. 14, 2025. https://bluepacknova.com/product/servo-motor-mg996r%EF%BC%88metal-gear%EF%BC%89/
[24] "GT2 Precision Timing Belt pitch 2MM x 6MM wide," Nerokas Online Store. https://store.nerokas.co.ke/SKU-1305
[25] "ALUMINIUM GT2 TIMING PULLEY - 6MM BELT - 20 TOOTH - 5MM BORE," Nerokas Online Store. https://store.nerokas.co.ke/SKU-1526?search=pulley&description=true
[26] "OV7670 Camera module," Pixel Electric Company Limited, 2025. https://www.pixelelectric.com/sensors/distance-vision/camera-sensor/ov7670-camera-module-module/
[27] "Male-Male Jumper Wire 40pcs Dupont," Pixel Electric Company Limited, 2025. https://www.pixelelectric.com/instruments-tools/wire-and-cables/cables/dupont-jumper-cable/male-male-jumper-wire-40pcs-dupont/
[28] "Buck Converter XL4015 — BluePack Nova," bluepackventures.com, Sep. 18, 2025. https://bluepacknova.com/product/buck-converter-x4015/
[29] New Workspace, "RDJ2026 Dataset," Roboflow Universe, 2026. [Online]. Available: https://universe.roboflow.com/allen-wachio/rdj2026-qkydt
[30] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO11," GitHub repository, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics
[31] Sony Semiconductor Solutions, "IMX500 Intelligent Vision Sensor Technical Documentation," Technical Documentation, 2023.
[32] Sony Semiconductor Solutions, "Model Compression Toolkit (MCT) User Guide," User Guide, 2023.
[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.
[34] A. Godil, "Model Predictive Path Integral Control: From Theory to Parallel Computation," *J. Guid. Control Dyn.*, vol. 40, no. 2, pp. 344–357, 2017.
[35] D. Fox, W. Burgard, and S. Thrun, "Markov Localization for Mobile Robots in Dynamic Environments," *J. Artif. Intell. Res.*, vol. 11, pp. 391–427, 1999.
[36] ROS 2 Navigation Team, "Nav2 Documentation." [Online]. Available: https://docs.nav2.org/. [Accessed: Oct. 24, 2023].
[37] M. Camurri, M. Mallarino, M. Milazzo, and G. Metta, "Behavior Trees for Robot Navigation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1234–1241, 2020.