electronics

MDPI

Article

Eight-Wheel Mecanum Omnidirectional Autonomous Mobile Robot: Kinematics, Architecture, and Validation

Leonardo D. Ortega-Lomeli, Luis C. Básaca-Preciado *, Ulises Orozco-Rosas, J. D. Castro-Toscano and M. A. Ponce-Camacho

CETYS Universidad, Calz. CETYS s/n, Col. Rivera, Mexicali C.P. 21259, Baja California, Mexico; leonardo.ortega@cetys.mx (L.D.O.-L.); ulises.orozco@cetys.mx (U.O.-R.); josedaniel.castro@cetys.mx (J.D.C.-T.); miguel.ponce@cetys.mx (M.A.P.-C.)

* Correspondence: luis.basaca@cetys.mx

Abstract

Autonomous omnidirectional vehicles that combine redundant holonomic kinematics, ROS 2/micro-ROS implementation, and simulation-to-real validation remain limited in the literature. This paper presents an eight-wheel Mecanum autonomous mobile robot for campus navigation in environments shared with pedestrians. The work formulates forward and inverse kinematics for the redundant eight-wheel topology and implements a distributed architecture in which ROS 2 handles high-level navigation and micro-ROS connects ESP32-based wheel interfaces. The platform integrates LiDAR, stereo vision, inertial, encoder, and ultrasonic sensing within a closed-loop navigation stack. Validation was conducted through Gazebo simulation and physical experiments using an out-and-back navigation protocol. In the physical platform, 91 of 100 missions were completed without safety interruptions, with pose-accuracy success rates of 96% for outbound legs and 81% for return legs under $e_p < 1.5$ m and $|e_\theta| < 15°$. Median errors at the intermediate waypoint were 0.64 m, 0.191 m, and 17°, while final-pose medians after return were 1.016 m, 0.573 m, and 28.5°. These results provide a quantitative baseline for campus-scale redundant Mecanum navigation and identify heading recovery as the main limitation.

Keywords: eight-wheel Mecanum robot; omnidirectional mobile robot; redundant holonomic kinematics; ROS 2; micro-ROS; distributed control; autonomous navigation; experimental validation

1. Introduction

University campuses impose demanding mobility constraints, including short-range trips through narrow corridors shared with pedestrians and service vehicles. At CETYS Universidad (Mexicali), we target a shuttle-class autonomous platform for reliable low-speed operation in spaces shared with pedestrians between campus buildings. Accordingly, we design an omnidirectional autonomous vehicle based on a redundant Mecanum wheel arrangement and a perception-aware navigation stack. The operational profile prioritizes repeatable short-range transits under strict geometric constraints and safety requirements driven by pedestrian detection and obstacle avoidance [1,2]. Typical campus displacements, such as trips from the School of Engineering building to the Admissions building, motivate a system engineered for repeatability and robust operation in structured indoor–outdoor passages [3,4].

We adopt Robot Operating System 2 (ROS 2) as the software substrate owing to its Data Distribution Service (DDS)-based data-centric publish/subscribe model, native support

Academic Editors: Dong Chen and Mingle Xu

Received: 17 April 2026
Revised: 14 May 2026
Accepted: 18 May 2026
Published: 3 June 2026

for distributed composition, and increasing emphasis on support for real-time execution in production deployments [5]. The DDS middleware is relevant in practice because it exposes explicit Quality-of-Service (QoS) policies that allow communication behavior to be tuned in terms of reliability, timeliness, and data availability according to control and safety requirements in distributed robotic systems [6]. For mobile robots operating near pedestrians, such configurability is valuable because command, state-estimation, and safety-related signals may require different trade-offs between delivery guarantees and timeliness across distributed processes.

To bridge resource-constrained edge nodes (e.g., wheel-speed control, encoder acquisition, and current/voltage monitoring), we integrate micro-ROS, which brings major ROS 2 concepts onto microcontroller units (MCUs) and connects MCU-resident nodes with the ROS 2 ecosystem through Micro XRCE-DDS, where XRCE refers to eXtremely Resource-Constrained Environments, while preserving compatibility with standard ROS 2 tools and application programming interfaces (APIs) [7]. This choice is relevant for campus navigation because low-level sensing and actuation functions can be executed on MCUs with real-time operating system (RTOS)-based scheduling, while computationally intensive functions remain on the high-level processor. Recent studies also show that the timing behavior of ROS 2/micro-ROS systems depends strongly on executor, communication scheduling, and QoS configuration, with measurable effects on end-to-end latency, message loss, and message periodicity [8,9].

Development and pre-deployment verification were conducted in Gazebo, a robotics-oriented simulator widely used in ROS-based research and industrial applications. The recent literature highlights Gazebo as one of the most widely used ROS-compatible simulation platforms due to its robust middleware integration, physics-based modeling, and support for repeatable evaluation of perception, control, and navigation algorithms [10,11]. In this work, Gazebo was employed to validate the robot model, assess sensor integration, and test navigation and control routines under controlled conditions before physical trials. This simulation workflow enabled rapid design iteration, systematic debugging, and reduced deployment risk prior to experiments on the real platform [11,12].

Reliable perception and decision-making in campus environments require a multi-sensor fusion that is robust to variations in illumination, occlusions, and dynamic agents. Cameras yield dense, semantically informative observations for lane/marker reading and pedestrian detection [13,14]; LiDAR contributes metrically accurate, illumination-invariant geometry for mapping and collision avoidance, with contemporary fusion and calibration methods tightening spatial alignment and depth consistency [15,16]. Short-range ultrasonic sensors improve proximity detection for low-speed docking and human-proximate maneuvers [17,18]. These design choices align with the trajectory of driver-assistance systems toward cooperative automation and justify redundancy through complementary sensing [3,4].

Beyond perception, omnidirectional locomotion necessitates a tailored kinematic treatment. Whereas four-wheel Mecanum platforms are well characterized, redundant configurations introduce geometric redundancy and richer ground-contact force/velocity distributions. We therefore derive forward and inverse kinematics with redundancy-aware control allocation consistent with ROS 2's component model [5,19,20].

The main contributions of this work are:

1. An omnidirectional kinematic formulation, forward and inverse, specialized to a redundant eight-wheel Mecanum topology, including a real-time feasible control-allocation strategy.
2. A distributed control and communication architecture spanning ROS 2 and micro-ROS nodes, with documented middleware configuration, QoS policy, controller update

rate, embedded sampling period, topic organization, and control parameters that support replication.

3. A simulation-in-the-loop and campus trial evaluation that quantifies positional and angular accuracy across representative routes and maneuvers.

The remainder of this paper is organized as follows. Section 2 reviews the related work on four-wheel Mecanum platforms, redundant holonomic bases, and ROS 2/micro-ROS robotic systems. Section 3 presents the kinematic formulation of the proposed eight-wheel Mecanum platform, including the forward and inverse models. Section 4 describes the distributed control system implemented with ROS 2 and micro-ROS. Section 5 reports the results obtained in the simulation. Section 6 presents the experimental results under real operating conditions. Section 7 discusses the main findings and limitations of the study. Finally, Section 8 concludes the paper and outlines directions for future work.

2. Related Work

This section positions the proposed platform relative to recent research in (i) four-wheel Mecanum robots, (ii) redundant holonomic bases and control allocation, (iii) ROS 2/micro-ROS-based mobile robots, and (iv) the positioning of this work. In addition, recent autonomous navigation research has addressed informative path planning, path generation in non-static environments, and computational acceleration for path-planning algorithms, which provide a broader context for the navigation problem considered here [21–23].

2.1. Four-Wheel Mecanum Platforms

Four-wheel Mecanum bases constitute a common baseline for holonomic ground robots in indoor service and logistics. Recent ROS-based implementations further confirm the relevance of this configuration in structured environments, including LiDAR-based mapping and path planning on Mecanum platforms, as well as industrial material-handling robots that combine omnidirectional locomotion with embedded control and perception modules [24,25]. Prior work has established canonical forward/inverse kinematics and explored practical challenges such as wheel slip, parameter uncertainty, and surface-dependent behavior. Recent studies continue to report improvements in motion control, robustness, and navigation integration, often focusing on mitigating slip-induced odometry drift, improving sensor fusion, and enhancing closed-loop tracking in structured environments [26–28].

2.2. Redundant Holonomic Bases (6–8 Wheels) and Control Allocation

Redundant holonomic platforms with more than four wheels have been explored when the application requires additional wheel–ground contact points, improved load distribution, or larger support areas than those typically provided by compact four-wheel platforms. In this context, multi-wheel Mecanum configurations are not intended to replace the four-wheel layout as a universal solution; rather, they represent application-driven alternatives for systems with larger chassis dimensions, higher payload demands, or greater mechanical support requirements. Tian et al. [29] presented a six-wheel Mecanum omnidirectional platform for heavy-load and flexible-motion applications, reporting advantages in carrying capacity relative to four-wheel platforms and analyzing motion behavior under wheel-failure conditions. Typiak et al. [30] further studied the dynamics of a six-wheeled unmanned rescue vehicle with Mecanum wheels, showing that six-wheel Mecanum layouts have also been considered in applied vehicle scenarios. Recent studies have also discussed six- and eight-wheel omnidirectional layouts in relation to reduced wheel pressure, heavy-load operation, vibration attenuation, and improved support stability in practical deployments [31,32].

The literature also shows that Mecanum wheel topology is not restricted to the conventional four-wheel arrangement. Li et al. [33] proposed a topological design method for Mecanum wheel configurations and emphasized that different application requirements may lead to different wheel layouts, including platforms with more than four wheels. In a related study, Li et al. [34] reported that large-scale equipment manufacturing and heavy industrial transportation may require multi-wheel Mecanum arrangements because the size and load capacity of conventional four-wheel Mecanum AGVs can become insufficient; their work discusses combined systems with 8, 12, 16, and 32 Mecanum wheels. Palacín et al. [31] further included a symmetric eight-wheel Mecanum platform in their kinematic interpretation of multi-wheeled omnidirectional robots, showing that eight-wheel layouts are part of the broader design space of Mecanum-based omnidirectional platforms. In addition, Cao et al. [35] built an experimental eight-wheel Mecanum omnidirectional prototype and studied trajectory tracking using a fuzzy adaptive PID controller, considering kinematic error, slip factor, and motion stability. From an applied multi-platform perspective, Song et al. [36] reported a two-vehicle Mecanum coordination system for large-scale omnidirectional transportation, where two four-wheel Mecanum vehicles operate cooperatively to transport heavy and oversized loads. Although this architecture is not a single rigid eight-wheel chassis, it reinforces the practical motivation for increasing the effective number of Mecanum wheel support points when payload size, mass, and support distribution dominate the design requirements.

This redundancy also introduces overactuation since multiple actuator command vectors can realize the same planar body motion. As a result, control allocation becomes an important problem in redundant holonomic platforms. Recent studies address this issue through generalized-inverse mappings and constrained or weighted optimization schemes that account for actuator limits and contact-dependent operating conditions, which can affect robustness under uneven contact and varying friction [37].

To synthesize these trends, Table 1 summarizes the role of four-, six-, and eight-wheel (4W, 6W, 8W) Mecanum configurations as reported in the literature. The comparison is conceptual and literature-based; it does not report matched experimental performance under identical chassis, mass, sensing, control, and trajectory conditions. Accordingly, the proposed eight-wheel topology is evaluated in this paper as a standalone redundant holonomic platform, rather than as a controlled replacement or experimentally proven superior alternative to conventional four-wheel Mecanum bases.

Table 1. Literature-based conceptual positioning of Mecanum wheel configurations.

Configuration	Typical Role in the Literature	Main Design Rationale	Main Limitation and Role in This Work
4W Mecanum	Canonical compact holonomic baseline for indoor service, logistics, and structured industrial environments [24,25,28].	Simple mechanical layout, mature forward/inverse kinematics, and broad adoption for compact omnidirectional mobility.	It provides fewer wheel–ground contact points and less redundancy than multi-wheel layouts. In this work, it is used as the theoretical baseline from which the eight-wheel kinematic model is extended.
6W Mecanum/redundant bases	Intermediate redundant support architecture reported for heavy-load omnidirectional platforms and applied to unmanned vehicles Additional support points, improved load capacity relative to compact 4W layouts, fault-tolerant behavior under selected wheel-failure conditions, and improved stability in specific layouts.	It introduces more complex kinematic/dynamic modeling, contact behavior, and velocity allocation. In this work, it supports the interpretation of redundancy as a practical design direction in holonomic bases.	
8W Mecanum/multi-Mecanum systems	Redundant layout reported in heavy-load, elongated, tandem, or combined Mecanum-platform systems [31,33–36].	Increased wheel–ground contact points, load sharing across more actuators, compatibility with larger mobile platforms, and redundancy-aware allocation.	It increases mechanical and control complexity and has fewer matched 4W–8W benchmarks in the literature. In this work, it is the proposed topology for kinematic formulation, ROS 2/micro-ROS implementation, and simulation-to-real validation.

Note: This table summarizes design roles reported in prior Mecanum-platform studies. It is intended as a literature-based conceptual comparison and not as a direct experimental benchmark between wheel configurations.

2.3. ROS 2, Micro-ROS, and Simulation-to-Real Validation

ROS 2 has enabled increasingly modular robotic systems in which perception, planning, and control are distributed across multiple processes and computing nodes. Because ROS 2 relies on DDS, middleware and QoS configuration become relevant design parameters, especially when message timing affects feedback control, coordination, or safety-related monitoring. Experimental studies have shown that ROS 2 timing behavior depends on both the selected DDS/RMW implementation and the configured QoS policies, with measurable effects on latency, packet loss, and timing variability in robotic deployments [38–40]. In addition, ROS 2 has also been explored as an interoperability layer in broader vehicle-oriented software architectures [41].

At the same time, Linux-based ROS 2 execution is not always sufficient for strict low-level control requirements. Recent evaluations show that standard ROS 2 deployments on Linux do not inherently guarantee deterministic execution, and that improved real-time behavior often requires additional mechanisms such as kernel optimization, dedicated scheduling strategies, or tightly synchronized communication layers [38,42]. This limitation is particularly relevant in wheeled mobile robots, where inverse kinematics only provides desired wheel-speed references, while the actual realization of those references still depends on high-rate encoder feedback and low-jitter actuation loops.

The micro-ROS layer addresses this architectural gap by extending ROS 2 to resource-constrained microcontrollers through Micro XRCE-DDS and a client–agent communication model, while remaining compatible with standard ROS 2 tools and abstractions [7]. Recent work further shows that the temporal behavior of micro-ROS is strongly affected by executor and callback scheduling, and that priority-driven scheduling can improve its real-time performance relative to the default execution model [8]. For mobile robots, this makes micro-ROS especially suitable for actuator-proximate sensing and control tasks, where wheel-level feedback loops must run with tighter timing guarantees than those typically achievable in a high-level Linux environment alone.

Although simulation remains central in this literature, fewer studies report both simulation and physical validation under integrated robotic operation. Some recent ROS 2-based works have demonstrated improvements through combined simulator and actual-machine experiments, but these validations are often conducted in controlled settings rather than in human-shared navigation scenarios [39,42]. Application-oriented Mecanum studies likewise tend to validate in structured environments such as greenhouses or factory-floor material-handling settings, rather than in campus-scale human-shared navigation scenarios [24,25]. This leaves a relevant gap between simulation-centric demonstrations and fully integrated real-world evaluations of distributed ROS 2/micro-ROS robotic systems, particularly for redundant holonomic mobile platforms.

2.4. Positioning of This Work

Compared with the dominant four-wheel Mecanum literature, this work addresses a redundant eight-wheel omnidirectional platform and explicitly formulates its forward and inverse kinematics for an overactuated mobile base. Compared with prior studies on redundant holonomic robots, the proposed approach integrates wheel-level embedded control via micro-ROS with ROS 2-based high-level navigation, thereby linking redundancy-aware motion generation to distributed robotic software. Finally, whereas many related studies remain limited to simulation, benchtop validation, or tightly controlled indoor trials, this work reports a simulation-to-real methodology and campus-scale experiments in human-shared navigation scenarios.

3. Kinematics

This section develops the kinematic model of an omnidirectional platform actuated by eight Mecanum wheels. Building on the classical four-wheel formulation, we define a dedicated geometric mapping for an eight-wheel layout composed of three wheel pairs along the longitudinal axis (front, central, rear) located at two distinct offsets, $\pm L_1$ (front/rear) and $\pm L_2$ (central), and at lateral offsets $\pm W$. This topology makes all wheels contribute to planar motion and yields an *overactuated* system: eight wheel speeds control a three-degree-of-freedom body twist, introducing kinematic redundancy not present in standard four-wheel models.

We first define frames, notation, and assumptions. We then derive an explicit 8×3 geometric matrix $\mathbf{J}_8$ for the platform. From $\mathbf{J}_8$ we obtain (i) inverse kinematics (wheel angular velocities from a desired body twist) and (ii) direct kinematics (body twist from measured wheel speeds). For direct kinematics, redundancy is resolved via a Moore–Penrose pseudoinverse (least-squares) mapping, yielding an implementation-ready estimator used by the motion controller, simulation, and field experiments.

3.1. Frames, Notation, and Assumptions

We define a body-fixed frame $\{\mathcal{B}\}$ at the chassis center, with x forward, y left, and z upward. The planar body twist is

$$\boldsymbol{\nu} = \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix}, \tag{1}$$

where V_x and V_y are the body-frame linear velocities along the x- and y-axes, respectively (m/s), and ω is the yaw rate about the z-axis (rad/s).

Wheel angular velocities are stacked as

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_{\mathrm{FL}} \\ \omega_{\mathrm{FR}} \\ \omega_{\mathrm{FCL}} \\ \omega_{\mathrm{FCR}} \\ \omega_{\mathrm{RCL}} \\ \omega_{\mathrm{RCR}} \\ \omega_{\mathrm{RL}} \\ \omega_{\mathrm{RR}} \end{bmatrix}, \tag{2}$$

where ω_i (rad/s) denotes the angular speed of wheel i. The indices denote *front-left* (FL), *front-right* (FR), *front-central-left* (FCL), *front-central-right* (FCR), *rear-central-left* (RCL), *rear-central-right* (RCR), *rear-left* (RL), and *rear-right* (RR).

Geometric parameters are wheel radius $r > 0$ (m), half-width $W > 0$ (m), and longitudinal offsets $L_1 > 0$ (m) for the front/rear pairs and $L_2 > 0$ (m) for the central pairs. Rollers are mounted at $\pm 45^\circ$ with the standard Mecanum orientation, consistent with the sign convention adopted in the kinematic derivation.

Assumptions. The derivation assumes planar motion of a rigid chassis, equal wheel radius, and ideal wheel–ground contact according to the standard Mecanum wheel kinematic constraint, while neglecting deformation and other higher-order effects.

Figure 1 illustrates the omnidirectional motion capabilities enabled by the adopted eight-wheel Mecanum configuration. In addition to the resulting platform motions, Figure 1 shows the roller orientation of each wheel in top view, with the front of the vehicle at the top. From front to rear, the left–right wheel pairs follow the pattern \/, /\, \/, and /\. The rollers follow an alternating mirrored arrangement along the longitudinal axis, which is the

mechanical basis that allows the lateral force components to combine or cancel depending on the commanded wheel angular velocities. This wheel configuration is consistent with the sign convention assumed in the inverse kinematic model developed in the subsection that follows. Under this arrangement, different combinations of wheel angular velocities generate longitudinal, lateral, diagonal, and rotational motions in the plane. To support this interpretation, Table 2 summarizes representative kinematic validation cases, including prescribed body-twist components and the corresponding wheel angular velocities for the eight-wheel platform.

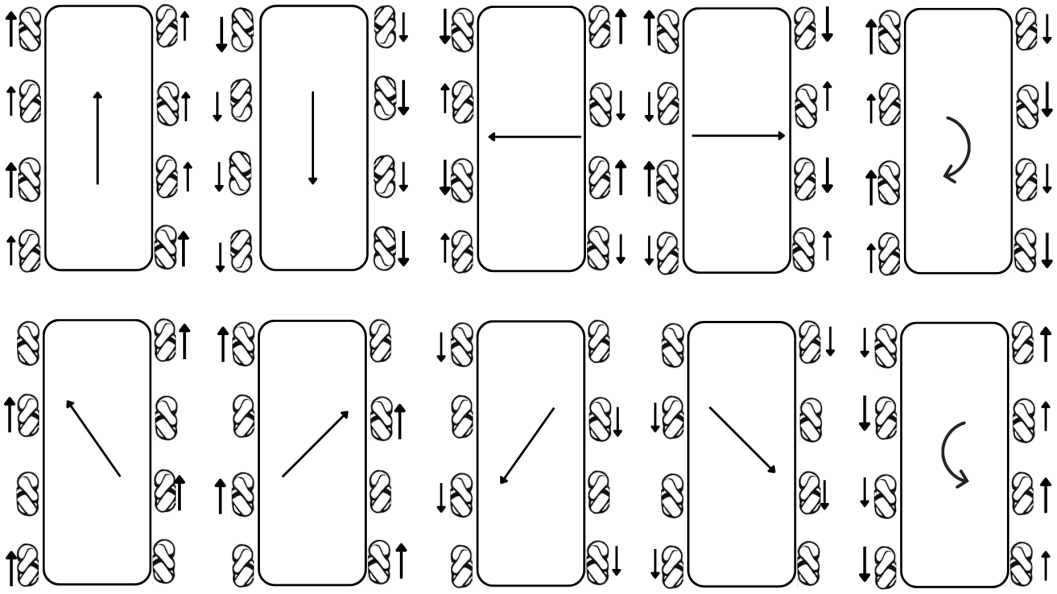

Figure 1. Representative omnidirectional motion patterns generated by different wheel angular-velocity combinations in the proposed eight-wheel Mecanum platform.

Table 2. Representative kinematic validation cases for the eight-wheel Mecanum autonomous vehicle. The table reports prescribed body twists and the corresponding wheel angular velocities obtained from the inverse kinematic model, which were used to verify the expected platform motions during simulation tests and experimental tests.

Direction	V_x	V_y	ω	ω_{FL}	ω_{FR}	ω_{FCL}	ω_{FCR}	ω_{RCL}	ω_{RCR}	ω_{RL}	ω_{RR}
Forward	2	0	0	19.685	19.685	19.685	19.685	19.685	19.685	19.685	19.685
Backward	−2	0	0	−19.685	−19.685	−19.685	−19.685	−19.685	−19.685	−19.685	−19.685
Right	0	−2	0	19.685	−19.685	−19.685	19.685	19.685	−19.685	−19.685	19.685
Left	0	2	0	−19.685	19.685	19.685	−19.685	−19.685	19.685	19.685	−19.685
Forward-left diagonal	2	2	0	0.000	39.370	39.370	0.000	0.000	39.370	39.370	0.000
Forward-right diagonal	2	−2	0	39.370	0.000	0.000	39.370	39.370	0.000	0.000	39.370
Backward-left diagonal	−2	2	0	−39.370	0.000	0.000	−39.370	−39.370	0.000	0.000	−39.370
Backward-right diagonal	−2	−2	0	0.000	−39.370	−39.370	0.000	0.000	−39.370	−39.370	0.000
Clockwise rotation	0	0	−2	21.063	−21.063	12.500	−12.500	12.500	−12.500	21.063	−21.063
Counterclockwise rotation	0	0	2	−21.063	21.063	−12.500	12.500	−12.500	12.500	−21.063	21.063

3.2. Inverse Kinematics for Mecanum Platforms (4→8 Wheels)

Inverse kinematics map the desired body twist in (1) to the wheel angular velocities in (2). For a conventional four-wheel Mecanum platform (rollers at ±45° and the usual sign convention), each wheel speed is a linear combination of the forward component V_x, the lateral component V_y (with signs dictated by the roller orientation), and the yaw

contribution proportional to the lever arm $(L + W)$, where L denotes the half-length of the wheelbase (m) and W the half-width (m). The resulting four-wheel inverse kinematics are

$$\begin{aligned} \omega_{FL} &= \frac{1}{r}\left(V_x - V_y - \omega\,(L+W)\right), \\ \omega_{FR} &= \frac{1}{r}\left(V_x + V_y + \omega\,(L+W)\right), \\ \omega_{RL} &= \frac{1}{r}\left(V_x + V_y - \omega\,(L+W)\right), \\ \omega_{RR} &= \frac{1}{r}\left(V_x - V_y + \omega\,(L+W)\right). \end{aligned} \tag{3}$$

In (3), the factor $1/r$ converts tangential rim speed to wheel angular speed (rad/s), and the sign pattern reflects the roller mounting. Equation (3) follows the standard Mecanum derivation in [20].

3.2.1. Extension to Eight Wheels

To account for the additional central wheel pairs located at longitudinal offsets $\pm L_2$, we extend the four-wheel structure by applying the same roller sign pattern at the corresponding lever arms. The resulting eight-wheel inverse kinematics are

$$\begin{aligned} \omega_{FL} &= \frac{1}{r}\left(V_x - V_y - \omega\,(L_1+W)\right), \\ \omega_{FR} &= \frac{1}{r}\left(V_x + V_y + \omega\,(L_1+W)\right), \\ \omega_{FCL} &= \frac{1}{r}\left(V_x + V_y - \omega\,(L_2+W)\right), \\ \omega_{FCR} &= \frac{1}{r}\left(V_x - V_y + \omega\,(L_2+W)\right), \\ \omega_{RCL} &= \frac{1}{r}\left(V_x - V_y - \omega\,(L_2+W)\right), \\ \omega_{RCR} &= \frac{1}{r}\left(V_x + V_y + \omega\,(L_2+W)\right), \\ \omega_{RL} &= \frac{1}{r}\left(V_x + V_y - \omega\,(L_1+W)\right), \\ \omega_{RR} &= \frac{1}{r}\left(V_x - V_y + \omega\,(L_1+W)\right). \end{aligned} \tag{4}$$

3.2.2. Compact Matrix Form and Geometric Matrix Definition

Equations (4) can be written compactly as

$$\boldsymbol{\omega} = \frac{1}{r}\,\mathbf{J}_8\,\boldsymbol{\nu}, \qquad \mathbf{J}_8 \in \mathbb{R}^{8\times 3}. \tag{5}$$

Here, $\mathbf{J}_8$ is the *geometric matrix* (geometric kinematic matrix) of the eight-wheel Mecanum platform: each of its eight rows corresponds to one wheel in (2), and its three columns correspond to the twist components (V_x, V_y, ω) in (1). The entries of $\mathbf{J}_8$ encode the wheel positions (lateral offsets $\pm W$ and longitudinal offsets $\pm L_1$, $\pm L_2$) and the roller orientation, which determines the $\pm V_y$ signs as well as the yaw lever arms $(L_i + W)$.

Because (5) maps three motion components into eight wheel-speed outputs, the platform is overactuated. In this work, we use (4) and (5) as a symmetric allocation consistent with the adopted sign convention and the mechanically symmetric actuation.

3.3. Direct Kinematics for Mecanum Platforms (4→8 Wheels)

Direct kinematics estimate the body twist (1) from measured wheel speeds (2). For a conventional four-wheel Mecanum platform (rollers at $\pm 45^\circ$ and the usual sign convention), the standard closed-form mapping is

$$\begin{aligned} V_x(t) &= \tfrac{r}{4}\left(\omega_{FL} + \omega_{FR} + \omega_{RL} + \omega_{RR}\right), \\ V_y(t) &= \tfrac{r}{4}\left(-\omega_{FL} + \omega_{FR} + \omega_{RL} - \omega_{RR}\right), \\ \omega(t) &= \tfrac{r}{4\,(L+W)}\left(-\omega_{FL} + \omega_{FR} - \omega_{RL} + \omega_{RR}\right). \end{aligned} \tag{6}$$

Equation (6) follows [20].

3.3.1. Extension to Eight Wheels

Using the same sign convention as in (4) and the longitudinal offsets L_1 and L_2, the eight-wheel twist components are obtained in closed form as

$$\begin{aligned} V_x(t) &= \tfrac{r}{8}\Big(\omega_{FL} + \omega_{FR} + \omega_{FCL} + \omega_{FCR} \\ &\qquad + \omega_{RCL} + \omega_{RCR} + \omega_{RL} + \omega_{RR}\Big), \\ V_y(t) &= \tfrac{r}{8}\Big(-\omega_{FL} + \omega_{FR} + \omega_{FCL} - \omega_{FCR} \\ &\qquad - \omega_{RCL} + \omega_{RCR} + \omega_{RL} - \omega_{RR}\Big), \\ \omega(t) &= \tfrac{r}{4\left((L_1+W)+(L_2+W)\right)} \\ &\quad \Big(-\omega_{FL} + \omega_{FR} - \omega_{FCL} + \omega_{FCR} \\ &\qquad - \omega_{RCL} + \omega_{RCR} - \omega_{RL} + \omega_{RR}\Big). \end{aligned} \tag{7}$$

3.3.2. Least-Squares Interpretation

For completeness, when using the compact form (5), the eight-wheel direct kinematics can be written as the least-squares estimate

$$\boldsymbol{v}(t) = r\,\mathbf{J}_8^{\dagger}\,\boldsymbol{\omega}(t), \qquad \mathbf{J}_8^{\dagger} = (\mathbf{J}_8^{\top}\mathbf{J}_8)^{-1}\mathbf{J}_8^{\top}, \tag{8}$$

assuming $\mathbf{J}_8$ has full column rank. In this work, we implement (7).

4. Control System

The control system is structured as a distributed closed-loop architecture in which high-level navigation and wheel-speed control run in ROS 2, while encoder acquisition, PWM actuation, and embedded communication are executed on ESP32 microcontrollers through micro-ROS. Unlike a conventional four-wheel Mecanum base, the proposed eight-wheel platform is overactuated and therefore requires both redundancy-aware wheel-speed allocation and body-motion reconstruction from redundant wheel measurements. The overall signal flow and feedback structure are summarized in Figure 2.

At the body level, the navigation stack generates the commanded planar twist

$$\boldsymbol{v}_{\mathrm{cmd}} = \begin{bmatrix} V_x & V_y & \omega \end{bmatrix}^{\mathsf{T}},$$

The control architecture distributes high-level navigation, localization, perception, and wheel-speed control in ROS 2, while the ESP32 micro-ROS nodes provide actuator interfacing and encoder-based wheel-speed feedback. Nav2 generates the commanded body twist, which is mapped into wheel-speed references through the inverse kinematics in (4). Measured wheel speeds are transformed back into body motion through the direct kinematics in (7) and fused with IMU data for state estimation.

The ROS 2 side was executed using ROS 2 Humble with the `rmw_cyclonedds_cpp` RMW implementation and `ROS_DOMAIN_ID=0`. The ESP32 nodes were connected to the ROS 2 graph through the micro-ROS Agent using the Micro XRCE-DDS client–agent

communication model. This configuration was used during the reported simulation and physical validation experiments.

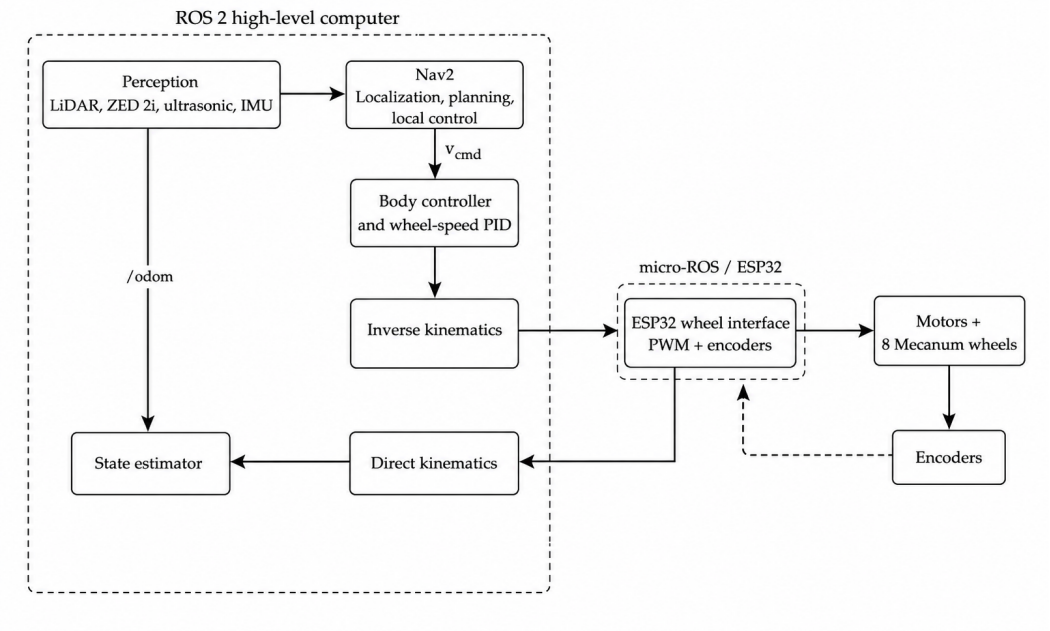

Figure 2. Control architecture of the proposed eight-wheel Mecanum platform. Solid arrows indicate the main sensing, control-command, and kinematic data flow, while the dashed arrow represents encoder feedback from the wheel actuators to the ESP32-based micro-ROS interface.

4.1. Wheel-Speed Control Implementation

The wheel-speed regulation was implemented as a distributed ROS 2/micro-ROS loop. The high-level ROS 2 side executes a wheel-speed control node written in Python 3.10, whereas the ESP32 micro-ROS nodes provide actuator interfacing and encoder-based wheel-speed feedback. For each of the eight wheels, the ROS 2 controller subscribes to the measured wheel angular-velocity topic, subscribes to the corresponding wheel-speed reference topic, and publishes a control-effort command consumed by the ESP32 micro-ROS node. The main implementation parameters of this distributed wheel-speed control loop are summarized in Table 3.

The ROS 2 wheel controller was implemented using the `simple_pid` library. Eight independent controllers were instantiated, one per wheel, using the joint list `fl_joint`, `fr_joint`, `fcl_joint`, `fcr_joint`, `bcl_joint`, `bcr_joint`, `rl_joint`, and `rr_joint`. The controller subscribes to encoder-derived wheel-speed measurements through the `<joint_name>` topics and to wheel-speed references through the `<joint_name>_velocity` topics. It publishes the resulting control command through the `<joint_name>_PID` topics. All wheel-speed measurements, references, and control-effort commands use `std_msgs/msg/Float32`. The encoder-measurement and wheel-reference subscriptions use a best-effort QoS policy with a queue depth of 10, as reported in Table 3.

The controller timer was configured with a period of 0.100 s, corresponding to a nominal control-loop frequency of 10 Hz. The controller output was limited to the range $[-41.89, 41.89]$, consistent with the wheel angular-speed command range used by the embedded interface. In the configuration used during the reported experiments, the PID gains were set to $K_p = 0.0$, $K_i = 0.6$, and $K_d = 0.0$. Therefore, the implemented controller used a PID software structure with only the integral action active. When the wheel-speed reference was zero, the controller output was forced to zero, and the accumulated integral action was reset by toggling the controller automatic mode.

On the embedded side, each ESP32 micro-ROS node receives the corresponding `<joint_name>_PID` command and maps it to the PWM command sent to the motor driver. The same ESP32 node estimates wheel angular velocity from the quadrature encoder and publishes the measured value back to ROS 2 through the `<joint_name>` topic. Thus, the closed-loop velocity regulation is distributed: feedback processing and control-effort generation occur in ROS 2, whereas encoder acquisition, PWM actuation, and microcontroller-to-ROS 2 communication occur through micro-ROS.

Table 3. ROS 2/micro-ROS wheel-speed control implementation parameters.

Category	Implemented Value	Execution Layer	Purpose
Middleware configuration	ROS 2 Humble; `rmw_cyclonedds_cpp`; `ROS_DOMAIN_ID=0`; micro-ROS Agent/Micro XRCE-DDS	ROS 2/Linux; ESP32/micro-ROS	Defines the DDS/RMW configuration and the client–agent bridge between the ROS 2 graph and the ESP32 micro-ROS nodes.
High-level control node	`pid_controller_node`; `MultiThreadedExecutor`; eight independent wheel controllers	ROS 2/Python	Computes wheel-level control efforts from measured wheel angular velocity and wheel-speed references.
Wheel topics	Measurements: `<joint_name>`; references: `<joint_name>_velocity`; control effort: `<joint_name>_PID`	ROS 2 ↔ micro-ROS	Implements the distributed feedback loop between the ROS 2 controller and the ESP32 wheel nodes.
Message type and QoS	`std_msgs/msg/Float32`; best-effort subscriptions with depth 10; publisher queue depth 10	ROS 2/Python	Defines the message format and QoS policy used for wheel measurements, references, and control-effort commands.
Control-loop timing	Timer period: 0.100 s; nominal frequency: 10.000 Hz	ROS 2/Python	Sets the update rate at which wheel-level control efforts are computed.
Controller configuration	$K_p = 0.0$, $K_i = 0.6$, $K_d = 0.0$; output limits $[-41.89, 41.89]$; zero-reference output forced to zero with integral reset	ROS 2/Python	Defines the PID software structure used in the reported configuration, with only integral action active.
Embedded wheel interface	ESP32 micro-ROS node; encoder update period 100 ms; PWM frequency 50 Hz; PWM range 1000 µs to 2000 µs; neutral command 1500 µs	ESP32/micro-ROS/Arduino	Receives the ROS 2 control-effort command, generates PWM actuation, estimates wheel angular velocity from the encoder, and publishes feedback to ROS 2.
Timing-scope limitation	Latency, jitter, packet loss, callback execution time, and worst-case execution time were not independently benchmarked	System-level validation	Clarifies that the reported values correspond to configured sampling and control parameters, not to measured end-to-end real-time guarantees.

Note: The table reports implementation parameters extracted from the ROS 2 Python controller and the ESP32 micro-ROS wheel-node code. These values describe configured sampling rates, control parameters, topics, QoS policies, and middleware configuration.

4.1.1. State Estimation

Wheel-speed measurements are used to reconstruct planar body motion, while IMU data provide inertial information for heading and angular-rate estimation. Because Mecanum platforms are susceptible to slip, wheel odometry alone is insufficient for reliable state reconstruction over longer traverses. For that reason, wheel-derived motion and inertial measurements are fused to produce `/odom` and `/state_estimate`, which are consumed by the navigation stack and used in performance evaluation.

4.1.2. Navigation and Obstacle Handling

The Nav2 stack performs map-based localization, global path planning, and local motion control with obstacle avoidance. From the estimated state and perception data, Nav2 computes the commanded body twist, which is subsequently transformed into wheel-level references through (4). The LiDAR and ZED 2i stereo camera (Stereolabs Inc., San Francisco, CA, USA) constitute the main sensing sources for localization and environment perception, whereas ultrasonic sensors provide short-range proximity cues for low-speed safety reactions near obstacles and pedestrians.

4.1.3. ROS 2/Micro-ROS Node Organization

The implementation is distributed across three main node groups. First, a high-level ROS 2 controller receives `/cmd_vel` and state information, applies the inverse kinematic allocation, and publishes `/wheel_speed_ref`. Second, the ESP32 micro-ROS wheel nodes provide encoder acquisition and PWM actuation, while the ROS 2 wheel-speed controller closes the velocity loop using the encoder-derived wheel-speed measurements published by the embedded nodes. Third, a state-estimation node combines wheel-derived odometry and IMU data to publish `/odom` and `/state_estimate`. This partition preserves ROS 2 interoperability while assigning actuator-proximal sensing, PWM generation, and embedded communication to the microcontroller layer.

Figure 3 shows the current physical implementation of the autonomous platform and highlights the main onboard components used in the proposed system. The vehicle integrates a Velodyne VLP-16 LiDAR (Velodyne Lidar, Inc., San Jose, CA, USA) for environmental perception, a stereo camera for visual sensing, an ESP32 microcontroller for low-level control and communication tasks, a laptop computer as the main onboard processing unit, a battery pack for power supply, and the eight Mecanum wheels that provide omnidirectional mobility. This hardware arrangement supports the distributed ROS 2/micro-ROS architecture and the experimental validation presented in this work.

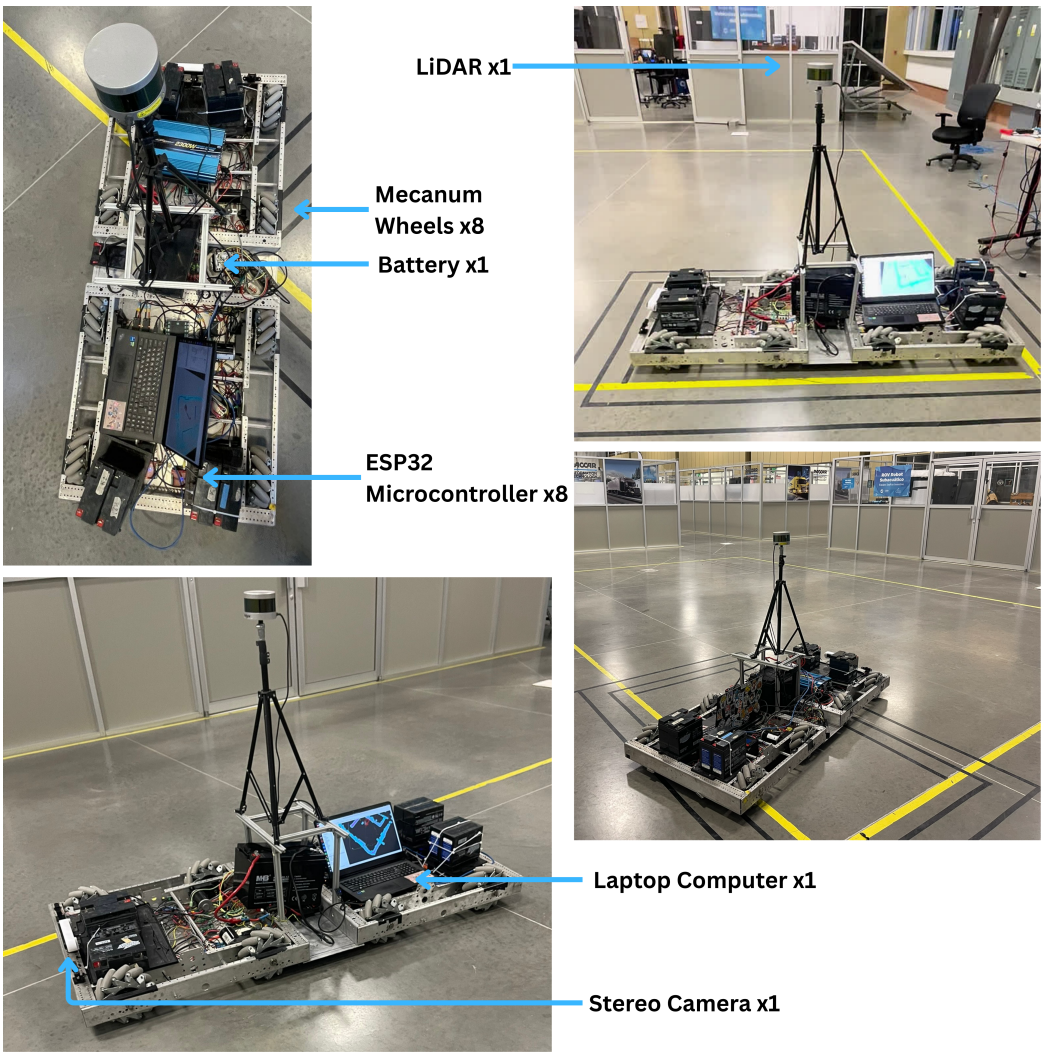

Figure 3. Top view of the autonomous eight-wheel Mecanum platform showing the main onboard hardware components, including the Velodyne VLP-16 LiDAR, stereo camera, Mecanum wheels, battery pack, onboard laptop computer, and ESP32 microcontroller.

The resulting architecture is not presented as evidence of experimental superiority over four- or six-wheel baselines; rather, it establishes the control structure required by a redundant eight-wheel Mecanum platform and the distributed ROS 2/micro-ROS implementation used in this work.

5. Results in Simulation

This section presents the simulation-based evaluation of the proposed autonomous platform prior to physical deployment. Beyond providing a visual confirmation of navigation execution, the purpose of these trials is to assess whether the integrated ROS 2 architecture, perception stack, and closed-loop control scheme operate consistently under controlled conditions. In this sense, simulation serves as an intermediate validation stage between the analytical developments presented in the previous sections and the physical experiments reported later. It allows the repeatability of the navigation task to be examined across multiple runs, while also enabling observation of how translational and angular errors evolve over the mission. To support this analysis, the section is organized into three parts: the simulated environment and visualization setup, the acquisition of navigation data over repeated out-and-back missions, and the statistical comparison of the terminal errors at the intermediate and final poses.

5.1. Simulated Environment and Visualization Setup

Figure 4 illustrates the visualization framework used during the simulation trials. All experiments were executed in Gazebo using a corridor-like environment designed to emulate constrained indoor navigation conditions. In parallel, RViz2 was used to monitor the perception and navigation stack in real time. The visualization includes the global occupancy grid `/map` shown in cyan, the local costmap inflation shown in light blue, the LiDAR point clouds, and the robot model at its current estimated pose. In addition, the Navigation2 panel reports that both localization and navigation remain in the *active* state during operation, confirming that the platform receives navigation goals, updates its pose estimate, and executes the corresponding motion commands throughout the simulated runs.

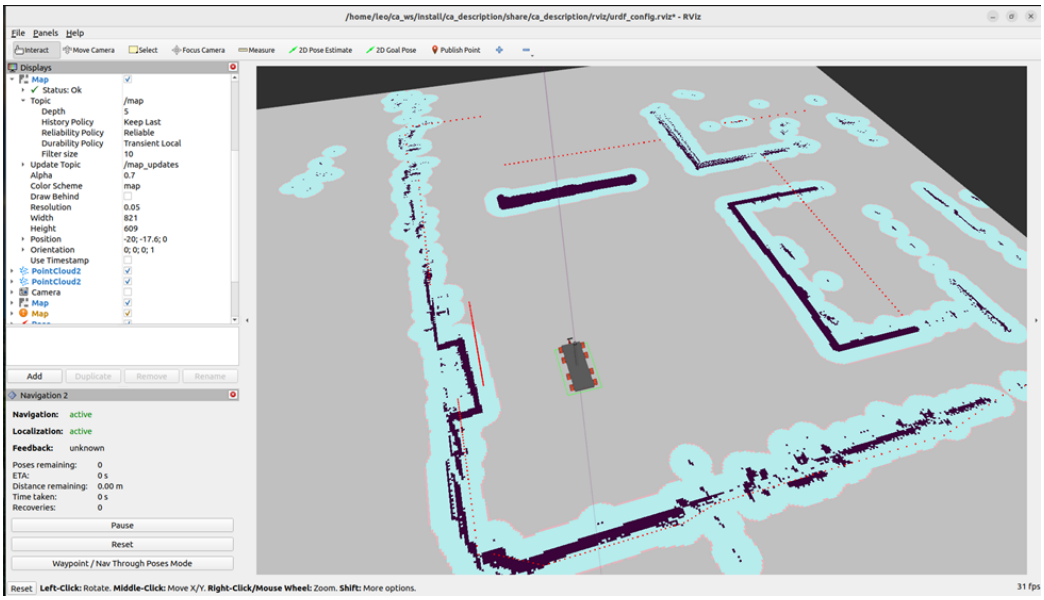

Figure 4. Sensor data visualization in RViz2 during Gazebo trials. *Note:* The global map (`/map`, cyan), local costmap inflation (light blue), LiDAR point clouds, and the robot model are displayed. The Navigation2 panel reports *active* localization and navigation during the run.

Complementing the RViz2 view, Figure 5 shows the Gazebo world used for the sensor integration tests. The layout reproduces corridor-like passages with turns and narrow segments intended to stress perception, local planning, and short-range obstacle handling. Ultrasonic range measurements are visualized as fan-shaped beams near the chassis, evidencing their correct operation in proximity sensing. At the same time, the WitMotion HWT905 IMU provides tri-axial acceleration, angular velocity, and orientation signals. These signals were verified under controlled maneuvers and incorporated into odometry and closed-loop control. Therefore, the simulated environment was not limited to geometric path playback, but rather included the principal sensing and estimation elements required by the navigation architecture.

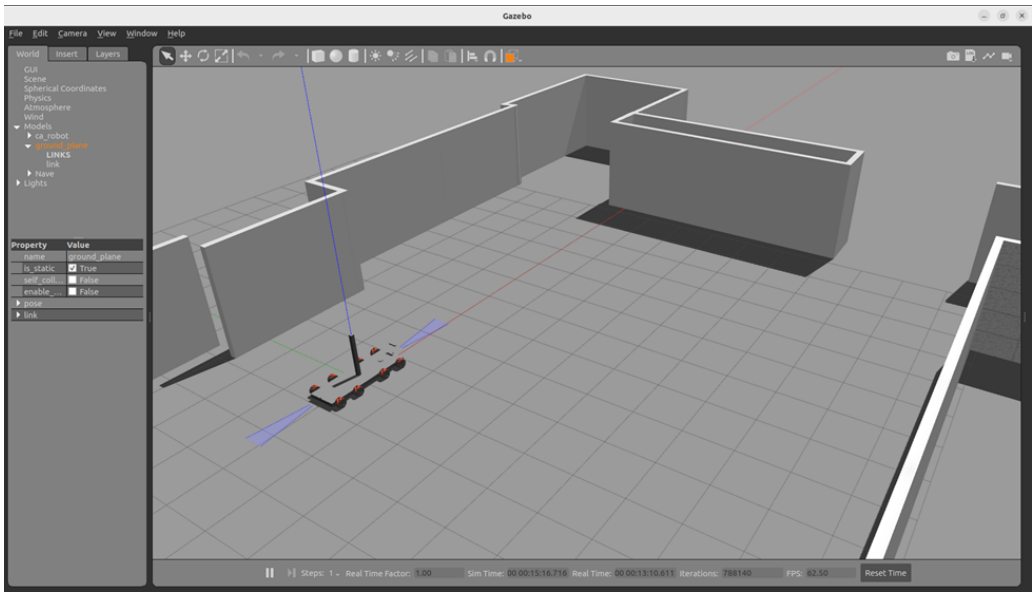

Figure 5. Gazebo sensor integration used in simulation trials. Note: Short-range obstacle detection was performed using MB1260 XL-MaxSonar-EZL0 ultrasonic range sensors (MaxBotix Inc., Brainerd, MN, USA).are visualized as fan-shaped range beams around the chassis, while the IMU (HWT905, WitMotion Shenzhen Co., Ltd., Shenzhen, Guangdong, China) provides acceleration, angular velocity, and orientation signals that are consistent with the commanded motions and feed odometry and control.

5.2. Navigation Data Acquisition

A total of 100 autonomous navigation trials were executed in Gazebo following an out-and-back protocol. For each run, three quantities were recorded: (i) the measured pose at Position A after completion of the outbound motion from the origin; (ii) the measured final pose after the return segment, which ideally should coincide with the initial pose; (iii) the corresponding terminal errors $(\Delta x, \Delta y, \Delta\theta)$ at both stages, defined with respect to the desired reference poses. This protocol was adopted to distinguish the accuracy achieved at the intermediate waypoint from the cumulative error observed after the full mission cycle.

The simulated results reveal a clear difference between the outbound and return stages. At Position A, the mean errors reached 0.771 m in x, 0.869 m in y, and 37.200° in θ. At the final pose, the mean errors decreased to 0.265 m in x and 0.648 m in y, whereas the heading error remained relatively high at 35.700°. These values indicate that the outbound stage is more demanding in translational terms, especially in the lateral component, whereas the return stage shows smaller mean positional deviations but still exhibits a noticeable angular mismatch. This result is relevant because it suggests that, even in a controlled simulation setting, heading regulation is more difficult to stabilize than planar position. Therefore, the simulation results should be interpreted by distinguishing mission-level completion from

final-pose accuracy. The robot was able to reach the target region with bounded planar error in most runs; however, the terminal heading error remained comparatively large. Consequently, the simulated mission-level success rate reflects successful navigation toward the target region under the defined tolerance, rather than docking-level pose accuracy or precise final yaw alignment.

Consistent with these definitions, Tables 4 and 5 report representative per-trial results for the outbound and return stages, respectively. Since including all 100 runs in the main text would be impractical, Tables 6 and 7 summarize the complete simulated dataset through descriptive statistics.

Table 4. Representative navigation results in simulation for the outbound stage (origin → Position A), shown as a subset of the 100 runs. All trials started from $(x, y, \theta) = (0.0,\ 0.0,\ 0^\circ)$ and used the same target pose at Position A, $(x, y, \theta) = (5.0,\ -5.0,\ 0^\circ)$.

	Position A (real)			**Error A**		
#	x	y	θ	Δx	Δy	$\Delta\theta$
1	5.23	−5.34	40	0.23	0.34	40
2	4.1	−5.74	40	0.9	0.74	40
3	4.25	−5.91	40	0.75	0.91	40
4	4.1	−5.9	40	0.9	0.9	40
5	4.2	−5.9	40	0.8	0.9	40
6	4.24	−5.86	30	0.76	0.86	30
7	4.1	−6.05	30	0.9	1.05	30
8	4.28	−5.75	30	0.72	0.75	30
9	4.1	−5.8	40	0.9	0.8	40
10	4.18	−5.9	40	0.82	0.9	40
90	4.17	−5.96	40	0.83	0.96	40
91	5.27	−5.42	40	0.27	0.42	40
92	4.18	−5.87	30	0.82	0.87	30
93	4.27	−6.011	40	0.73	1.011	40
94	4.2	−5.85	40	0.8	0.85	40
95	4.24	−5.79	40	0.76	0.79	40
96	4.34	−6.11	40	0.66	1.11	40
97	4.13	−6.02	40	0.87	1.02	40
98	4.12	−5.89	40	0.88	0.89	40
99	4.25	−5.88	40	0.75	0.88	40
100	4.1934	−5.8	40	0.8066	0.8	40

Note: A representative subset of runs is shown for brevity. The descriptive statistics reported later were computed from the complete set of 100 simulated trials. Positions are given in meters and angles in degrees.

5.3. Comparative Analysis of Error A and Final Error

To characterize the central tendency and dispersion of the simulated navigation errors, the median, mean, mode, variance, and standard deviation were computed for x, y, and θ at both mission stages. The resulting values are reported in Tables 6 and 7. This comparison is useful because it allows the outbound waypoint-reaching performance to be distinguished from the cumulative effect observed after the robot completes the full mission and attempts to recover the initial pose.

The statistical results show that the outbound stage exhibits larger mean errors in both translational components, with 0.771 m in x and 0.869 m in y, compared with 0.265 m and 0.648 m, respectively, at the final pose. In practical terms, this indicates that the approach toward Position A generates greater positional deviation than the return segment toward the origin. The effect is especially evident in the lateral component, where both the mean and the variance are larger during the outbound motion. This behavior is consistent with the corridor geometry and with the fact that the first segment requires the platform to settle

into the commanded path while simultaneously coordinating omnidirectional motion and heading evolution.

Table 5. Simulation navigation results for the return stage (Position A → final position), reported as a chronological subset comprising the first 10 and last 10 trials out of 100. All trials started from the same nominal pose at Position A, $(x, y, \theta) = (5.0, -5.0, 0^\circ)$, and used the same final target pose, $(x, y, \theta) = (0.0, 0.0, 0^\circ)$.

Trial	Final Position (real)			Final Error		
	x	y	θ	Δx	Δy	$\Delta\theta$
Trials 1–10						
1	−0.180	0.510	−40	0.180	0.510	40
2	−0.150	−0.650	−40	0.150	0.650	40
3	−0.290	−0.470	−40	0.290	0.470	40
4	−0.140	−0.770	−40	0.140	0.770	40
5	−0.720	−1.000	30	0.720	1.000	30
6	−0.560	−0.950	−30	0.560	0.950	30
7	−0.200	−0.570	40	0.200	0.570	40
8	−0.210	−0.620	40	0.210	0.620	40
9	−0.090	−0.070	40	0.090	0.070	40
10	−0.220	−0.850	30	0.220	0.850	30
Trials 91–100						
91	−0.150	−0.450	40	0.150	0.450	40
92	−0.130	−0.760	40	0.130	0.760	40
93	−0.250	−0.570	40	0.250	0.570	40
94	−0.160	−0.410	40	0.160	0.410	40
95	−0.140	−0.600	40	0.140	0.600	40
96	−0.410	−0.980	−30	0.410	0.980	30
97	−0.250	−0.910	−30	0.250	0.910	30
98	−0.280	−0.390	40	0.280	0.390	40
99	−0.610	−0.870	−30	0.610	0.870	30
100	−0.204	−0.490	40	0.204	0.490	40

Note: A chronological subset of 20 out of 100 simulated trials is shown for brevity. The descriptive statistics reported later were computed from the complete dataset. Positions are given in meters and angles in degrees.

By contrast, the heading component remains comparatively unstable in both stages. The mean angular error reaches 37.200° at Position A and 35.700° at the final pose, while the standard deviation remains on the order of 4° to 5°. Unlike the translational components, which improve during the return segment, the angular component does not show a comparable reduction. This suggests that orientation convergence is less robust than translational convergence, even under the controlled conditions of simulation. Therefore, the dominant residual error in the simulated campaign is not purely positional, but rotational.

Overall, the simulation results indicate that the proposed architecture can complete the prescribed navigation task with bounded translational error across repeated runs, although heading regulation remains the most sensitive state variable. This finding is important because it anticipates a pattern that may become more pronounced in physical experiments, where wheel–ground interaction, sensing uncertainty, and cumulative drift are more difficult to control than in simulation. These results also show that translational convergence and heading convergence should be interpreted as different performance dimensions. The reduction in final translational error does not imply an equivalent improvement in orientation. Therefore, the angular component is treated as a separate limitation of the current navigation configuration, rather than as a secondary effect of the planar position error.

5.4. Simulation Mission-Level Success Criteria and Rates

A run was classified under a simulation mission-level criterion when the planar error at the corresponding terminal pose, defined as $e_p = \sqrt{e_x^2 + e_y^2}$, remained below 1.5 m and the heading error satisfied $|e_\theta| \leq 40^\circ$. Under this criterion, the outbound leg achieved 98 successful runs out of 100, whereas the return leg also achieved 98 successful runs out of 100. Table 8 reports the corresponding mission-level success rates together with their 95% confidence intervals. This criterion should be interpreted as a simulation-stage mission-completion threshold rather than as a strict final-pose accuracy threshold. In particular, the angular tolerance used in simulation is wider than the one used in the physical experiments; therefore, the simulated mission-level success rates should not be directly compared with the experimental pose-accuracy rates. The relatively large simulated heading errors reported in Tables 6 and 7 indicate that, although the platform generally reached the commanded region, terminal yaw regulation remained limited. A stricter yaw tolerance would be required to evaluate docking, final alignment, or narrow-passage maneuvers.

Table 6. Statistical analysis of *Error A*.

	x (m)	y (m)	θ (°)
Median	0.790	0.860	40.000
Mean	0.771	0.869	37.200
Mode	0.800	0.800	40.000
Variance	0.015	0.031	20.160
Standard deviation	0.112	0.168	4.481

Table 7. Statistical analysis of *Final error*.

	x (m)	y (m)	θ (°)
Median	0.230	0.620	40.000
Mean	0.265	0.648	35.700
Mode	0.220	0.800	40.000
Variance	0.019	0.059	28.510
Standard deviation	0.138	0.243	5.339

Table 8. Simulation mission-level success rates under $e_p < 1.5$ m and $|e_\theta| \leq 40^\circ$.

Stage	Success/Total	Mission-Level Rate (%)	95% CI
Outbound (Origin → A)	98/100	98.0	[93.0, 99.4]
Return (A → Origin)	98/100	98.0	[93.0, 99.4]

6. Experimental Results

This section presents the physical evaluation of the proposed autonomous platform under real operating conditions. Unlike the simulation campaign, the physical trials capture the effects of wheel–ground interaction, sensing uncertainty, local trajectory corrections, and cumulative drift during repeated out-and-back missions. The analysis is organized into three parts: navigation data collection, descriptive statistics of the terminal errors at the intermediate and final poses, and pose-accuracy success rates under predefined thresholds.

6.1. Navigation Data Collection

A total of 100 autonomous navigation trials were conducted on the physical platform to evaluate its behavior under bounded campus-like conditions, including a polished

floor, visual references, and taped corridor constraints. All outbound legs started from the nominal origin $(x, y, \theta) = (0.0, 0.0, 0^\circ)$ and were commanded toward the nominal waypoint Position A, $(x, y, \theta) = (5.0, -5.0, 0^\circ)$. For each trial, the measured pose reached at Position A and the corresponding terminal errors (e_x, e_y, e_θ) were recorded.

After the outbound leg, the return command was executed from the *actual* pose reached by the robot at the end of the first segment, rather than from a reset at the nominal waypoint. Therefore, the final error reflects the cumulative deviation over the complete out-and-back mission, including the residual error carried from the outbound stage. The experimental test track and platform instrumentation are shown in Figure 6.

Consistent with this protocol, Tables 9 and 10 report a chronological subset of the experimental dataset, specifically the first 10 and last 10 trials, for the outbound and return stages, respectively. This deterministic selection keeps the presentation concise while preserving traceability to the full 100-trial campaign. The complete dataset was used to compute the descriptive statistics reported in Tables 11 and 12.

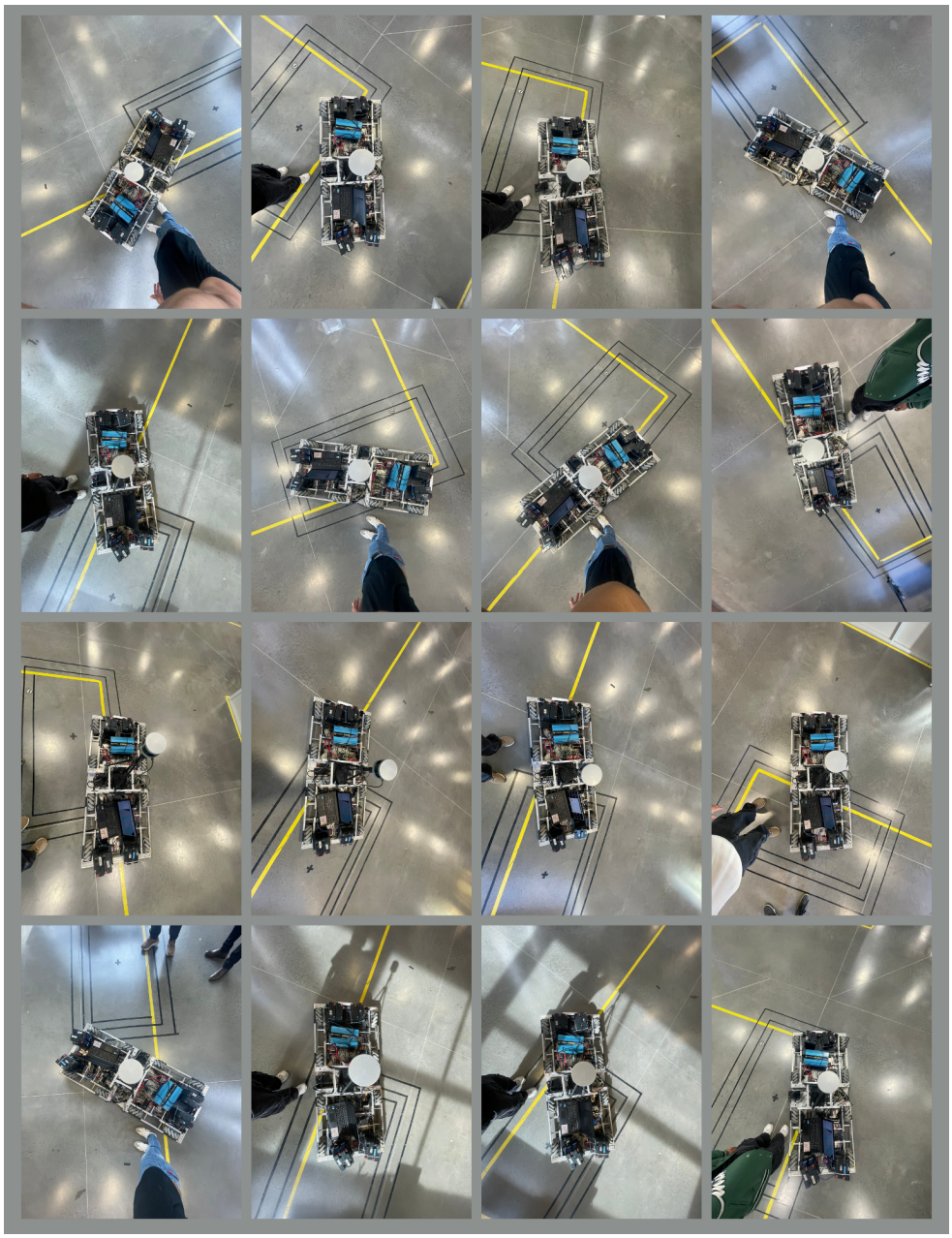

Figure 6. Experimental tests: platform during data acquisition on a marked track.

Table 9. Experimental navigation results for the outbound stage (origin → Position A), reported as a chronological subset comprising the first 10 and last 10 trials out of 100. All trials started from the nominal origin $(x, y, \theta) = (0.0, 0.0, 0^\circ)$ and used the same target pose at Position A, $(x, y, \theta) = (5.0, -5.0, 0^\circ)$.

Trial	Measured Position at A			Error at A		
	x	y	θ	e_x	e_y	e_θ
Trials 1–10						
1	6.71	−4.76	8	1.71	0.24	8
2	5.94	−4.90	6	0.94	0.10	6
3	6.05	−4.85	8	1.05	0.15	8
4	5.89	−4.92	5	0.89	0.08	5
5	5.92	−4.90	6	0.92	0.10	6
6	5.69	−5.76	−48	0.69	0.76	48
7	6.27	−4.49	22	1.27	0.51	22
8	5.64	−5.09	−8	0.64	0.09	8
9	5.30	−4.33	66	0.30	0.67	66
10	5.62	−5.19	−17	0.62	0.19	17
Trials 91–100						
91	5.99	−5.03	−2	0.99	0.03	2
92	6.14	−5.35	−17	1.14	0.35	17
93	5.95	−5.50	−28	0.95	0.50	28
94	6.05	−5.20	−11	1.05	0.20	11
95	5.38	−5.07	−10	0.38	0.07	10
96	5.84	−5.18	−12	0.84	0.18	12
97	5.82	−5.44	−28	0.82	0.44	28
98	5.50	−5.54	−47	0.50	0.54	47
99	5.54	−4.87	13	0.54	0.13	13
100	5.60	−4.89	10	0.60	0.11	10

Note: A chronological subset of 20 out of 100 trials is shown for brevity. The descriptive statistics reported later were computed from the complete dataset. Positions are given in meters and angles in degrees.

6.2. Statistical Analysis of the Error at Position A

Table 11 summarizes the descriptive statistics of the terminal errors measured at Position A. The results show that the translational components remain comparatively bounded, with median errors of 0.64 m in x and 0.19 m in y, whereas the heading component exhibits substantially greater variability, with a median error of 17° and a standard deviation of 16.07°. This indicates that the platform generally reaches the intermediate waypoint with moderate positional consistency, but orientation regulation remains more sensitive to disturbances, local corrections, and small mismatches accumulated during the outbound maneuver.

These results indicate that translational and angular convergence should be interpreted separately. Although the platform reached the intermediate waypoint with moderate positional consistency, the heading statistics show that final yaw regulation was substantially less stable than planar positioning. This behavior is consistent with a navigation protocol in which reaching the target region was prioritized over docking-level terminal orientation. Consequently, the outbound results support waypoint-reaching capability, but they do not imply precise final-pose alignment.

Table 10. Experimental navigation results for the return stage (actual Position A → final position), reported as a chronological subset comprising the first 10 and last 10 trials out of 100. In each trial, the return command was issued from the actual pose reached at the end of the outbound leg; therefore, the final error reflects cumulative deviation over the complete mission rather than a reset from the nominal waypoint.

Trial	Final Position (Real)			Final Error		
	x	y	θ	e_x	e_y	e_θ
Trials 1–10						
1	1.35	0.19	8	1.35	0.19	8
2	1.26	0.09	4	1.26	0.09	4
3	1.71	−0.27	−9	1.71	0.27	9
4	1.11	−0.24	−12	1.11	0.24	12
5	0.54	−1.06	−63	0.54	1.06	63
6	1.25	0.20	9	1.25	0.20	9
7	1.12	−0.36	−18	1.12	0.36	18
8	1.18	0.73	32	1.18	0.73	32
9	1.36	0.17	7	1.36	0.17	7
10	1.50	−0.70	−25	1.50	0.70	25
Trials 91–100						
91	0.57	−1.16	−64	0.57	1.16	64
92	0.63	−0.98	−57	0.63	0.98	57
93	0.62	−0.80	−52	0.62	0.80	52
94	1.19	−0.60	−27	1.19	0.60	27
95	0.70	−0.53	−37	0.70	0.53	37
96	0.81	−0.75	−43	0.81	0.75	43
97	0.61	−0.90	−56	0.61	0.90	56
98	0.27	−1.26	−78	0.27	1.26	78
99	0.64	−1.11	−60	0.64	1.11	60
100	0.94	0.44	25	0.94	0.44	25

Note: A chronological subset of 20 out of 100 trials is shown for brevity. Positions are given in meters and angles in degrees.

Table 11. Statistics of the Error at Position A (in m and °).

	x	y	θ
Median	0.64	0.191	17.00
Mean	0.7013	0.2526	20.5556
Mode	0.60	0.20	17.00
Variance	0.15239	0.05310	259.72166
Standard deviation	0.37655	0.23044	16.06590

Note: x, y in m, θ in °.

6.3. Statistical Analysis of the Final Error

Table 12 reports the descriptive statistics of the terminal errors after the complete out-and-back cycle. Compared with the results at Position A, the final pose exhibits larger central values in both translation and heading, which indicates cumulative error propagation over the full mission. The effect is especially pronounced in the angular component, where both the mean and the dispersion increase substantially. This behavior suggests that the platform preserves acceptable waypoint-reaching capability during the outbound segment, but experiences a measurable degradation in pose recovery once the full closed route is completed. The final-pose statistics further show that heading error

accumulates over the complete out-and-back mission. Since the return command was issued from the actual pose reached at Position A, any residual yaw mismatch from the outbound segment could propagate into the return stage. This helps explain why the final angular dispersion is substantially larger than the translational dispersion. In practical terms, the platform demonstrated the ability to complete the route and return toward the origin, but the current implementation did not achieve docking-level final-pose accuracy.

Table 12. Statistics of the Final Error (in m and °).

	x	y	θ
Median	1.0157	0.5733	28.50
Mean	0.9881	0.6032	32.4255
Mode	0.00	0.20	10.00
Variance	0.14642	0.13478	945.79765
Standard deviation	0.38265	0.36713	30.75382

Note: x, y in m, θ in °.

6.4. Experimental Pose-Accuracy Success Criteria and Rates

A run was classified as successful under the experimental pose-accuracy criterion when the planar error at the target, defined as $e_p = \sqrt{e_x^2 + e_y^2}$, remained below 1.5 m and the heading error satisfied $|e_\theta| < 15°$. Under this criterion, the outbound leg achieved 96 successful runs out of 100, whereas the return leg achieved 81 successful runs out of 100. Table 13 reports the corresponding pose-accuracy success rates together with their 95% confidence intervals. The difference between outbound and return performance is consistent with the statistical results discussed above: the platform shows greater reliability in reaching the intermediate waypoint than in recovering the initial pose after completing the full mission, indicating that cumulative drift and orientation mismatch remain the dominant sources of degradation in the experimental setting. Unlike the simulation-stage mission-level criterion, this experimental criterion applies a stricter angular threshold and should therefore be interpreted as a pose-accuracy metric. This distinction is important because completing the route and reaching the target region do not necessarily imply docking-level final-pose accuracy.

Table 13. Experimental pose-accuracy success rates under $e_p < 1.5$ m and $|e_\theta| < 15°$.

Stage	Pose-Accuracy Success/Total	Pose-Accuracy Rate (%)	95% CI
Outbound (Origin → A)	96/100	96.0	[90.1, 98.9]
Return (A → Origin)	81/100	81.0	[72.6, 87.6]

7. Discussion

The results indicate that the proposed direct and inverse kinematic formulations are consistent with the observed behavior of the eight-wheel Mecanum platform in both simulation and physical experiments. In this sense, the study does not seek to establish superiority over another wheel configuration, but rather to verify that the derived kinematic model provides an operational mapping between body motion and wheel velocities for the implemented architecture.

The comparison between simulation and experimental trials shows that the overall motion trends are preserved across both environments, which supports the use of the simulation framework as a development and verification stage. At the same time, the larger dispersion observed in the physical experiments, especially in heading, indicates that real operation remains affected by factors that are only partially represented in the current

simulator, such as wheel–ground interaction variability, slip, small asymmetries in contact conditions, sensing uncertainty, and accumulated odometric error. The success criteria used in simulation and physical experiments should be interpreted as different performance levels. The simulation criterion is reported as a mission-level success metric because it uses a wider heading tolerance and evaluates whether the platform reaches the commanded region under controlled conditions. By contrast, the physical experimental criterion is reported as a pose-accuracy metric because it uses a stricter angular tolerance. Therefore, the corresponding success rates should not be interpreted as directly interchangeable measures of the same performance level. This distinction explains why the system can show high mission-completion performance while still exhibiting limited terminal yaw accuracy.

The relatively large heading errors observed in both simulation and physical experiments require a separate interpretation from the translational mission-completion results. In the present evaluation, successful navigation was primarily associated with reaching the target region without safety interruption and with bounded planar error, rather than with docking-level final-pose accuracy. This distinction is important because a heading error of this magnitude may be tolerable when the vehicle only needs to reach a broad stopping area in a low-speed campus-navigation scenario, but it would not be sufficient for docking, narrow-corridor alignment, passenger boarding alignment, or other tasks requiring accurate terminal yaw regulation.

Several factors may explain why the closed-loop system did not suppress the yaw error more effectively. First, the evaluation protocol and goal acceptance behavior emphasized arrival at the target region more strongly than terminal yaw alignment; consequently, the robot could complete the mission-level objective even when its final orientation differed from the nominal reference. Second, Mecanum odometry is sensitive to lateral slip and roller–ground interaction, which can introduce yaw drift during combined lateral and rotational maneuvers. Third, the eight-wheel layout increases the number of wheel–ground contact points, making the platform more sensitive to non-uniform normal loading, small wheel-radius differences, floor irregularities, and unequal traction among wheels. These effects are only partially represented in the current simulation model and can accumulate during the out-and-back trajectory, especially during the return segment.

Accordingly, the angular error is considered one of the main limitations of the current implementation. Future improvements should include stricter terminal yaw tolerances, explicit heading regulation in the local controller, improved IMU–wheel odometry fusion, friction/contact parameter identification, and additional validation under maneuvers where final orientation is a primary performance requirement.

From this perspective, the main significance of the observed error is not that the kinematic model fails, but that the complete autonomous system is subject to disturbances and uncertainties beyond the idealized geometric mapping. The results therefore support the validity of the proposed kinematic model at the system level, while also showing that dynamic effects, sensing imperfections, and environmental variability play an important role in final pose accuracy.

The eight-wheel redundant topology should also be interpreted carefully. In this work, redundancy is treated primarily as a structural characteristic of the platform that enables the proposed velocity allocation and omnidirectional motion, rather than as a claimed advantage over alternative configurations. Accordingly, the present contribution lies in the formulation, implementation, and validation of the corresponding kinematic and control framework for this specific vehicle architecture.

Some limitations of the present study should nevertheless be stated explicitly. The validation was conducted on bounded campus-like routes and under a limited set of operating conditions. In addition, the current simulation model does not yet include

full parameter identification of friction, contact effects, actuator nonlinearities, or richer disturbance models. Therefore, while the results demonstrate the feasibility and consistency of the proposed framework, they should not yet be interpreted as a complete predictive representation of all real-world operating conditions.

A further limitation of the present validation is that the experimental campaign did not include a systematic evaluation under dynamic obstacles, multiple surface conditions, longer routes, or controlled pedestrian-density variations. Although the physical trials were conducted in a campus-like environment and the navigation stack incorporated obstacle-detection capabilities, the reported results should be interpreted as a baseline evaluation under bounded and relatively structured operating conditions. Therefore, the present study does not yet quantify the robustness of the platform under highly variable pedestrian motion, surface-dependent wheel–ground interaction, extended route lengths, or repeated exposure to dynamic obstacle fields.

Future validation campaigns should explicitly include these conditions through a staged protocol. First, dynamic-obstacle experiments should be conducted with controlled moving objects and supervised pedestrian crossings to evaluate local-planner response, stopping behavior, and recovery after obstacle clearance. Second, surface-dependent tests should compare polished indoor flooring, rougher concrete, outdoor pavement, and transition zones, since Mecanum wheel motion is sensitive to friction, roller–ground contact, and slip. Third, longer routes should be used to quantify accumulated odometric drift, heading degradation, and localization stability over extended campus-scale trajectories. Finally, pedestrian-shared trials should be performed only under a formal safety protocol, with speed limits, emergency-stop supervision, and predefined exclusion zones. These additional experiments are required before the system can be generalized from structured campus trials to broader real-world deployment scenarios.

A further limitation concerns communication timing. Although the ROS 2/micro-ROS architecture was functionally validated during simulation and physical navigation trials, the present study did not include a dedicated communication-timing benchmark. The reported timing values correspond to configured controller and embedded sampling periods, including the 10.000 Hz ROS 2 wheel-control loop and the 100.000 ms encoder update period, but they do not represent measured end-to-end communication latency, jitter, packet loss, callback execution time, or worst-case round-trip delay between the ROS 2 computer and the ESP32 micro-ROS nodes. Consequently, the current results should be interpreted as evidence of functional distributed control integration rather than as a formal real-time communication characterization. Future work will include timestamped round-trip measurements, synchronized logging between the ROS 2 computer and ESP32 nodes, and communication-load tests to quantify mean latency, maximum latency, jitter, packet loss, and execution-time variability under nominal and stressed operating conditions.

8. Conclusions

This paper presented the modeling, implementation, and validation of an eight-wheel omnidirectional autonomous vehicle integrated natively in ROS 2 and micro-ROS. The proposed framework combined direct and inverse kinematic development, URDF-based modeling, Gazebo simulation, distributed control, sensor integration, and campus-scale experimental evaluation.

The obtained results show that the formulated kinematic model is consistent with the implemented platform and supports omnidirectional motion in both simulation and physical operation. In this sense, the work demonstrates that the proposed mapping between the body twist and wheel angular velocities is suitable for the eight-wheel Mecanum archi-

tecture considered here and can be successfully integrated into a complete autonomous navigation stack.

A direct comparison between simulation and real experiments further shows that simulation is effective for architecture verification, controller debugging, and lower-risk system integration, whereas physical trials reveal greater variability, particularly in heading regulation and final-pose repeatability. Therefore, the main contribution of this work is not a claim of superiority over other wheel configurations, but the successful formulation and system-level validation of the kinematic, control, and simulation framework for the proposed eight-wheel platform.

The reported implementation parameters improve the reproducibility of the distributed ROS 2/micro-ROS control stack. However, formal end-to-end timing characterization, including latency, jitter, packet loss, callback execution time, and worst-case execution time, remains outside the scope of the present validation and is identified as future work.

The experimental scope should also be interpreted with caution. The present validation was conducted under bounded campus-like conditions and did not include systematic testing with dynamic obstacles, different surface materials, longer routes, or controlled pedestrian-density variations. Accordingly, the reported results are best understood as a baseline validation of the proposed architecture rather than as evidence of deployment-level robustness under all real-world operating conditions.

Future work may focus on reducing the simulation-to-reality gap through improved parameter identification, more realistic disturbance and sensor models, tighter state estimation, stricter terminal yaw regulation, improved IMU–wheel odometry fusion, and hardware-in-the-loop validation using the same micro-ROS communication structure adopted in the physical prototype. The validation protocol can also be extended to include controlled dynamic-obstacle trials, surface-dependent experiments, longer campus routes, and supervised scenarios involving pedestrians under a formal safety protocol.

In addition, a matched 4W–8W Mecanum benchmark is identified as a relevant future direction to quantify the effect of redundant actuation under controlled conditions. Such an evaluation could keep the chassis geometry, total mass, perception stack, Nav2 configuration, route geometry, maximum body velocities, and success criteria as equivalent as possible, while reporting final planar error, heading error, mission-level and pose-accuracy success rates, RMS tracking error, wheel-speed saturation, control effort, and, where instrumentation permits, load distribution across the wheel contact points.

This work demonstrates that an eight-wheel Mecanum vehicle can be modeled, controlled, and experimentally evaluated within a ROS 2/micro-ROS framework for campus-scale autonomous navigation while also identifying simulation-to-real consistency, heading robustness, communication-timing characterization, and broader environmental validation as the main areas requiring further refinement.

Author Contributions: Conceptualization, L.D.O.-L. and L.C.B.-P.; methodology, L.D.O.-L. and L.C.B.-P.; software, L.D.O.-L.; validation, L.D.O.-L. and L.C.B.-P.; formal analysis, L.D.O.-L. and L.C.B.-P.; investigation, L.D.O.-L.; prototype design and development, L.D.O.-L. and L.C.B.-P.; prototype construction, L.D.O.-L. and L.C.B.-P.; experimental testing, L.D.O.-L. and L.C.B.-P.; resources, L.C.B.-P.; data curation, L.D.O.-L.; writing—original draft preparation, L.D.O.-L.; writing—review and editing, L.C.B.-P., U.O.-R., J.D.C.-T. and M.A.P.-C.; visualization, L.D.O.-L., L.C.B.-P., U.O.-R., J.D.C.-T. and M.A.P.-C.; supervision, L.C.B.-P. and U.O.-R.; project administration, L.C.B.-P.; funding acquisition, L.C.B.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CETYS Universidad through the Comisión Institucional de Investigación (CII).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors gratefully acknowledge CETYS Universidad and the Centro de Innovación y Diseño (CEID; Center for Innovation and Design) for providing the facilities, equipment, and institutional support necessary for this work. The authors also thank Ismael Ayala, Ever Alcaráz, Isaí Contreras, and Erick Salazar for their valuable support during the experimental testing of the prototype.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Xiang, W.; Honarvar Shakibaei Asli, B.; Ji, A. A Vision–Locomotion Framework Toward Obstacle Avoidance for a Bio-Inspired Gecko Robot. *Electronics* **2026**, *15*, 882. [CrossRef]
2. Orozco-Rosas, U.; Picos, K.; Montiel, O.; Castillo, O. Environment Recognition for Path Generation in Autonomous Mobile Robots. In *Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine*; Castillo, O., Melin, P., Eds.; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2020; Volume 827, pp. 273–288. [CrossRef]
3. Bengler, K.; Dietmayer, K.; Färber, B.; Maurer, M.; Stiller, C.; Winner, H. Three Decades of Driver Assistance Systems: Review and Future Perspectives. *IEEE Intell. Transp. Syst. Mag.* **2014**, *6*, 6–22. [CrossRef]
4. Broggi, A.; Cerri, P.; Debattisti, S.; Laghi, M.C.; Medici, P.; Panciroli, M.; Versari, P. TerraMax Vision at the Urban Challenge 2007. *IEEE Trans. Intell. Transp. Syst.* **2010**, *11*, 194–205. [CrossRef]
5. Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W. Robot Operating System 2: Design, architecture, and uses in the wild. *Sci. Robot.* **2022**, *7*, eabm6074. [CrossRef]
6. Gambo, M.L.; Danasabe, A.; Almadani, B.; Aliyu, F.; Aliyu, A.; Al-Nahari, E. A Systematic Literature Review of DDS Middleware in Robotic Systems. *Robotics* **2025**, *14*, 63. [CrossRef]
7. Belsare, K.; Rodriguez, A.C.; Sánchez, P.G.; Hierro, J.; Kołcon, T.; Lange, R.; Lütkebohle, I.; Malki, A.; Losa, J.M.; Melendez, F.; et al. Micro-ROS. In *Robot Operating System (ROS)*; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2023; Volume 1051, pp. 3–55. [CrossRef]
8. Wang, Z.; Liu, S.; Ji, D.; Yi, W. Improving Real-Time Performance of Micro-ROS with Priority-Driven Chain-Aware Scheduling. *Electronics* **2024**, *13*, 1658. [CrossRef]
9. Mamani-Saico, A.; Yanyachi, P.R. Implementation and Performance Study of the Micro-ROS/ROS2 Framework to Algorithm Design for Attitude Determination and Control System. *IEEE Access* **2023**, *11*, 128451–128460. [CrossRef]
10. Flores González, J.M.; Coronado, E.; Yamanobe, N. ROS-Compatible Robotics Simulators for Industry 4.0 and Industry 5.0: A Systematic Review of Trends and Technologies. *Appl. Sci.* **2025**, *15*, 8637. [CrossRef]
11. Singh, M.; Kapukotuwa, J.; Gouveia, E.L.S.; Fuenmayor, E.; Qiao, Y.; Murray, N.; Devine, D. Comparative Study of Digital Twin Developed in Unity and Gazebo. *Electronics* **2025**, *14*, 276. [CrossRef]
12. Jeon, S.; Park, J.; Seo, D. Implementation of Re-Simulation-Based Integrated Analysis System to Evaluate and Improve Autonomous Driving Algorithms. *Vehicles* **2024**, *6*, 2209–2227. [CrossRef]
13. Ranft, B.; Stiller, C. The role of machine vision for intelligent vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 8–19. [CrossRef]
14. Liu, S.; Tang, J.; Zhang, Z.; Gaudiot, J.L. Computer Architectures for Autonomous Driving. *Computer* **2017**, *50*, 18–25. [CrossRef]
15. Zhen, W.; Hu, Y.; Liu, J.; Scherer, S. A Joint Optimization Approach of LiDAR–Camera Fusion for Accurate Dense 3-D Reconstructions. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3585–3592. [CrossRef]
16. Ye, C.; Pan, H.; Gao, H. Keypoint-Based LiDAR–Camera Online Calibration with Robust Geometric Network. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 2503011. [CrossRef]
17. Seven, S.; Dikmen, I.C.; Karadag, T.; Bakir, H.G.; Abbasov, T. Design and implementation of ultrasonic sonar system for autonomous vehicles. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018; pp. 1–4. [CrossRef]
18. Jeong, H.S.; Kim, J.H. Empirical Validation of a Multidirectional Ultrasonic Pedestrian Detection System for Heavy-Duty Vehicles Under Adverse Weather Conditions. *Sensors* **2025**, *25*, 5287. [CrossRef]
19. Macenski, S.; Martin, F.; White, R.; Clavero, J.G. The Marathon 2: A Navigation System. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 2718–2725. [CrossRef]
20. aheri, H.; Qiao, B.; Ghaeminezhad, N. Kinematic Model of a Four Mecanum Wheeled Mobile Robot. *Int. J. Comput. Appl.* **2015**, *113*, 6–9. [CrossRef]
21. Jara Ten Kathen, M.; Benitez, N.; Arzamendia, M.; Gutiérrez Reina, D. ACO-Path: ACO-Based Informative Path Planning with Gaussian Processes for Water Monitoring with a Fleet of ASVs. *Electronics* **2026**, *15*, 676. [CrossRef]

22. Orozco-Rosas, U.; Picos, K.; Montiel, O. Acceleration of Path Planning Computation Based on Evolutionary Artificial Potential Field for Non-static Environments. In *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications*; Castillo, O., Melin, P., Kacprzyk, J., Eds.; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2020; Volume 862, pp. 271–297. [CrossRef]
23. Orozco-Rosas, U.; Montiel, O.; Sepúlveda, R. An Optimized GPU Implementation for a Path Planning Algorithm Based on Parallel Pseudo-bacterial Potential Field. In *Nature-Inspired Design of Hybrid Intelligent Systems*; Melin, P., Castillo, O., Kacprzyk, J., Eds.; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2017; Volume 667, pp. 477–492. [CrossRef]
24. Xu, H.; Yu, G.; Wang, Y.; Zhao, X.; Chen, Y.; Liu, J. Path Planning of Mecanum Wheel Chassis Based on Improved A* Algorithm. *Electronics* **2023**, *12*, 1754. [CrossRef]
25. Li, H.; Liu, J.; Lyu, C.; Liu, D.; Liu, Y. Design and Implementation of Omnidirectional Mobile Robot for Materials Handling among Multiple Workstations in Manufacturing Factories. *Electronics* **2023**, *12*, 4693. [CrossRef]
26. Chen, A.; Khalil, E.; Huang, H.P.; Du, G. Slip Estimation and Compensation Control of Omnidirectional Wheeled Automated Guided Vehicle. *Electronics* **2021**, *10*, 840. [CrossRef]
27. Pizá, R.; Carbonell, R.; Casanova, V.; Cuenca, Á.; Llobregat, J.J.S. Nonuniform Dual-Rate Extended Kalman-Filter-Based Sensor Fusion for Path-Following Control of a Holonomic Mobile Robot with Four Mecanum Wheels. *Appl. Sci.* **2022**, *12*, 3560. [CrossRef]
28. Szeremeta, M.; Szuster, M. Neural Tracking Control of a Four-Wheeled Mobile Robot with Mecanum Wheels. *Appl. Sci.* **2022**, *12*, 5322. [CrossRef]
29. Tian, Y.; Zhang, S.; Liu, J.; Chen, F.; Li, L.; Xia, B. Research on a New Omnidirectional Mobile Platform with Heavy Loading and Flexible Motion. *Adv. Mech. Eng.* **2017**, *9*, 1–15. [CrossRef]
30. Typiak, A.; Lopatka, M.J.; Rykala, L.; Kijek, M. Dynamics of Omnidirectional Unmanned Rescue Vehicle with Mecanum Wheels. *AIP Conf. Proc.* **2018**, *1922*, 120005. [CrossRef]
31. Palacín, J.; Rubies, E.; Bitriá, R.; Clotet, E. Phasor-Like Interpretation of the Angular Velocity of the Wheels of Omnidirectional Mobile Robots. *Machines* **2023**, *11*, 698. [CrossRef]
32. Huang, Y.; Meng, R.; Yu, J.; Zhao, Z.; Zhang, X. Practical Obstacle-Overcoming Robot with a Heterogeneous Sensing System: Design and Experiments. *Machines* **2022**, *10*, 289. [CrossRef]
33. Li, Y.; Dai, S.; Zhao, L.; Yan, X.; Shi, Y. Topological Design Methods for Mecanum Wheel Configurations of an Omnidirectional Mobile Robot. *Symmetry* **2019**, *11*, 1268. [CrossRef]
34. Li, Y.; Ge, S.; Dai, S.; Zhao, L.; Yan, X.; Zheng, Y.; Shi, Y. Kinematic Modeling of a Combined System of Multiple Mecanum-Wheeled Robots with Velocity Compensation. *Sensors* **2020**, *20*, 75. [CrossRef] [PubMed]
35. Cao, G.; Zhao, X.; Ye, C.; Yu, S.; Li, B.; Jiang, C. Fuzzy Adaptive PID Control Method for Multi-Mecanum-Wheeled Mobile Robot. *J. Mech. Sci. Technol.* **2022**, *36*, 2019–2029. [CrossRef]
36. Song, H.; Wu, Y.; Wu, Y.; Zhou, G.; Luo, C. Two-Vehicle Coordination System for Omnidirectional Transportation Based on Image Processing and Deviation Prediction. *J. Control Autom. Electr. Syst.* **2021**, *32*, 875–883. [CrossRef]
37. Ottensmeier, M.; Prokop, G. Development of a Motion Control for a highly dynamic, self-propelled driving simulator. *Automot. Engine Technol.* **2023**, *8*, 17–42. [CrossRef]
38. Ye, Y.; Nie, Z.; Liu, X.; Xie, F.; Li, Z.; Li, P. ROS2 Real-time Performance Optimization and Evaluation. *Chin. J. Mech. Eng.* **2023**, *36*, 144. [CrossRef]
39. Jalil, A.; Kobayashi, J.; Saitoh, T. Performance Improvement of Multi-Robot Data Transmission in Aggregated Robot Processing Architecture with Caches and QoS Balancing Optimization. *Robotics* **2023**, *12*, 87. [CrossRef]
40. Kim, S.; Choi, H.; Lee, S.; Kim, M.; Shin, H.; Moon, C. A Dynamic Bridge Architecture for Efficient Interoperability Between AUTOSAR Adaptive and ROS2. *Electronics* **2025**, *14*, 3635. [CrossRef]
41. Hong, D.; Moon, C. Autonomous Driving System Architecture with Integrated ROS2 and Adaptive AUTOSAR. *Electronics* **2024**, *13*, 1303. [CrossRef]
42. Nichols, K.M.; Roembke, R.A.; Adamczyk, P.G. Real-Time Motor Control Using a Raspberry Pi, ROS, and CANopen over EtherCAT, with Application to a Semi-Active Prosthetic Ankle. *Actuators* **2025**, *14*, 84. [CrossRef]