# PID-Based Position and Trajectory Control of a Four-Wheeled Omnidirectional Robot Using Robot Operating System (ROS)

Mohammad Febri Duwi Prasetiyo[1*], Parama Diptya Widayaka[2]
[1,2]Electrical Engineering Department, State University of Surabaya
[1]A5 Building Ketintang Campus, Surabaya, 60231, Indonesia
[1]mohammadprasetiyo.22093@mhs.unesa.ac.id
[2]paramawidayaka@unesa.ac.id

**Abstract – Precise position and trajectory control in omnidirectional mobile robots is essential for applications in industrial automation and competitive robotics, including real-world scenarios such as the Indonesian Robot Contest (Soccer Wheeled Middle Size League Division), while also supporting sustainable technological development aligned with the Sustainable Development Goals (SDGs), particularly in industry innovation and infrastructure. This study presents the development and evaluation of a PID-based control system for a four-wheeled omnidirectional robot using the Robot Operating System (ROS) framework, validated through both simulation and real-world implementation. The system integrates sensor fusion from an MPU6050 gyroscope, rotary encoders, and magnetic encoders to provide real-time pose feedback. PID parameters were tuned using Ziegler–Nichols and a structured trial-and-error method and evaluated across multiple trajectory scenarios. Simulation results demonstrated stable performance with low overshoot and fast settling time, while real-world experiments showed that the trial-and-error method provides more balanced and robust performance under practical disturbances compared to the more aggressive Ziegler–Nichols approach. Unlike previous studies that focus primarily on simulation or simplified robot models, this work provides a systematic experimental validation of PID tuning strategies on a full-scale holonomic robot. The main novelty lies in the reproducible tuning procedure and the quantitative analysis of the simulation-to-reality performance gap. The results confirm that the proposed system achieves accurate and consistent trajectory tracking in both simulated and real environments.**

**Keywords:** Omnidirectional robot, Odometry, PID controller, Robot Operating System (ROS), position control, sensor fusion, Soccer Wheeled Middle Size League, trajectory tracking, SDGs.

## I. INTRODUCTION

The integration of robotics into modern automation systems has necessitated the development of highly maneuverable platforms capable of operating in constrained environments. Among these, the four-wheeled omnidirectional robot has garnered significant attention due to its unique kinematic ability to translate in any direction without changing its orientation [8], [10]. This holonomic characteristic makes it exceptionally suitable for applications in logistics, healthcare, and research laboratories where space is limited and precise positioning is paramount [9], [14].

The Proportional-Integral-Derivative (PID) controller remains a cornerstone in industrial and robotic control systems due to its structural simplicity and proven effectiveness in regulating parameters such as velocity and position [1], [2], [12]. By calculating an error value as the difference between a desired setpoint and a measured process variable, the PID controller applies corrective efforts based on proportional, integral, and derivative terms. For omnidirectional platforms, the challenge lies in tuning these parameters to manage the complex coupling between the four independent wheel velocities required to execute a specific translational or rotational command [19], [21].

Recent advancements in open-source software have positioned the Robot Operating System (ROS) as a leading framework for robotic development. ROS provides a modular, distributed architecture that facilitates seamless integration of sensors, actuators, and control algorithms [6]. When coupled with Gazebo, a high-fidelity physics simulator, ROS enables the validation of control strategies in a virtual environment before deployment on physical hardware, significantly reducing development time and mitigating the risk of mechanical damage [17], [18]. This simulation-based

validation approach has been widely adopted in experimental robotic research prior to real-world implementation [18].

Although PID control has been widely applied in mobile robotics, most existing studies either focus on simulation environments or simplified kinematic platforms such as differential drive robots. Furthermore, the comparison of tuning methods is often limited to theoretical or simulation-based analysis without thorough validation on real holonomic systems [5], [16]. As a result, there is a lack of reproducible experimental studies that systematically evaluate PID tuning robustness under real-world disturbances such as wheel slip and sensor noise.

This research addresses this gap by proposing a comprehensive PID-based position control system for a four-wheeled omnidirectional robot. The primary contributions of this paper are: (1) the design and implementation of a modular ROS-based control architecture integrating fused odometry from MPU6050, rotary, and magnetic encoders, which has been shown to improve localization accuracy in omnidirectional robots [11], [15]; (2) a comparative analysis of Ziegler-Nichols and trial-and-error tuning methods based on key performance indicators such as rise time, overshoot, steady-state error, and RMSE; and (3) an evaluation of the performance fidelity between Gazebo simulation and real-world implementation across various trajectory patterns.

## II. METHOD

### A. System Architecture

The proposed system is built upon a four-wheeled omnidirectional mobile robot with a mechanical footprint of 46 cm × 46 cm and a height of 77 cm, constructed using an aluminum profile frame. The actuation system comprises four PG45 DC motors coupled with omnidirectional wheels arranged in a 90-degree configuration, as shown in Figure 1 and 2.

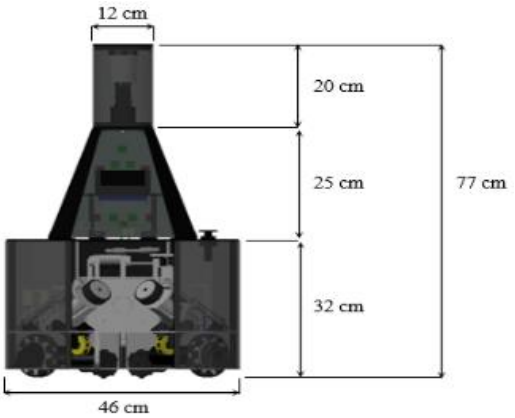


Figure 1. Robot Size Design

Figure 2. Side and Rear Views of the Robot Design

The hardware control architecture in Figure 3 employs a dual-microcontroller setup: an STM32F4-Discovery executes the main control loop, while an Arduino Nano acquires orientation data from the MPU6050 IMU. Motor feedback is obtained from integrated magnetic encoders on PG45 motors and external rotary encoders for odometry. Sensor-to-controller communication uses UART serial protocol. The STM32F4 microcontroller was selected due to its high processing capability and real-time performance required for multi-axis PID control. The Arduino Nano is utilized as a secondary controller for efficient sensor data acquisition. The MPU6050 sensor was chosen due to its low cost, compact size, and sufficient accuracy for orientation estimation in mobile robotic applications.

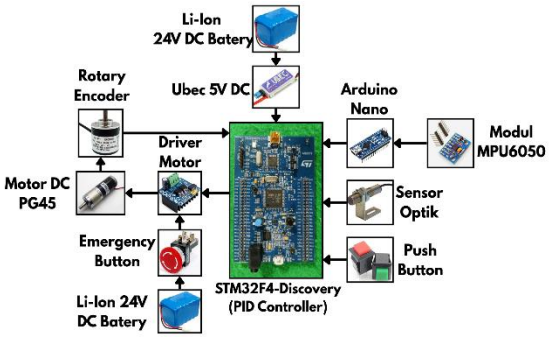


Figure 3. Robot Hardware Design

The software architecture consists of low-level embedded firmware and a high-level ROS-based control stack. The STM32F4 firmware, developed in C using STM32CubeIDE, implements PID control, sensor acquisition (encoders and MPU6050 via Arduino Nano), and PWM generation at 250 Hz. ROS Noetic manages simulation and high-level commands through nodes including gazebo_ros_control, robot_controller for inverse kinematics and PID, and odometry_publisher for fused wheel–IMU odometry. Communication via topics such as /cmd_vel, /joint_states, and /odom,

enables modular testing in simulation and deployment to hardware with minimal modification.

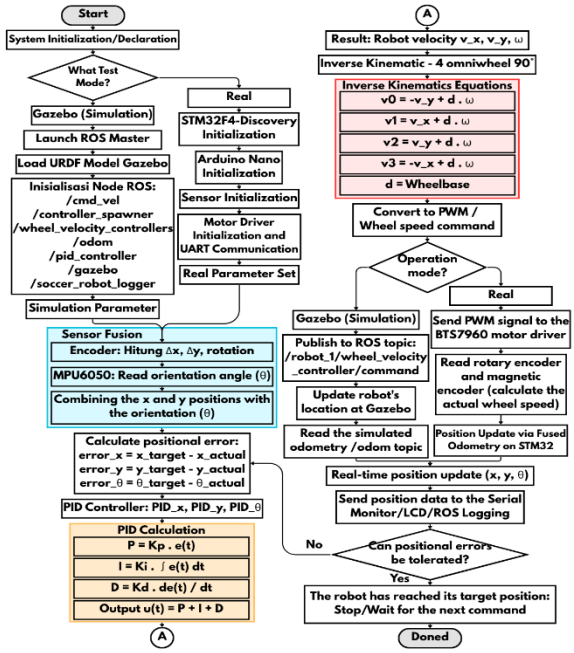


Figure 4. Robot Software Planning Flowchart

## B. Kinematics and Odometry

The motion of a four-wheeled omnidirectional robot is governed by its kinematic model, which defines the relationship between wheel velocities and the robot's body velocity. The robot's pose in the global coordinate frame is represented by $(x, y, \theta)$, and its instantaneous global velocities are given by the time derivatives of these coordinates, as expressed in (1):

$$(v_x, v_y, \omega) = \left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{d\theta}{dt}\right) \quad (1)$$

where $v_x$ and $v_y$ are the linear velocities along the global $X$ and $Y$ axes, and $\omega$ is the angular velocity about the vertical axis.

The local body velocities, consisting of longitudinal velocity $v$, lateral velocity $v_n$, and angular velocity $\omega$, are related to the global velocities through a rotation matrix dependent on the robot's heading angle $\theta$. This transformation is shown in (2):

$$\begin{bmatrix} v_x(t) \\ v_y(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} cos(a(t)) & sin(a(t)) & 0 \\ -sin(a(t)) & cos(a(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} v(t) \\ v_n(t) \\ \omega(t) \end{bmatrix} \quad (2)$$

For computational clarity, (2) can be expanded into the scalar form presented in (3):

$$\begin{aligned} v_x(t) &= v(t).cos(a(t)) + v_n(t).sin(a(t)) \\ v_y(t) &= v(t).-sin(a(t)) + v_n(t).cos(a(t)) \\ \omega(t) &= \omega(t) \end{aligned} \quad (3)$$

The inverse kinematics model maps the desired body velocities to the required angular velocities of the four omniwheels $(v_0, v_1, v_2, v_3)$. For a 90° wheel arrangement with each wheel positioned at a distance $d$ from the robot's center, this relationship is defined by (4):

$$\begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & d \\ -1 & 0 & d \\ 0 & -1 & d \\ 1 & 0 & d \end{bmatrix} \begin{bmatrix} v(t) \\ v_n(t) \\ \omega(t) \end{bmatrix} \quad (4)$$

Conversely, forward kinematics enables the estimation of the robot's body velocity from measured wheel velocities, a critical component for odometry. Equation (5) provides this forward mapping:

$$\begin{aligned} v(t) &= \frac{1}{2}(v_3 - v_1) \\ v_n(t) &= \frac{1}{2}(v_0 - v_2) \\ \omega(t) &= \frac{(v_0 + v_1 + v_2 + v_3)}{(4d)} \end{aligned} \quad (5)$$

Finally, to update the robot's global pose, the local pose increments $(\Delta x_l, \Delta y_l, \Delta\theta)$ obtained by integrating the velocities are transformed into the global coordinate frame using the rotation matrix shown in (6):

$$\begin{bmatrix} x_g \\ y_g \\ \omega \end{bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_l \\ y_l \\ \theta \end{bmatrix} \quad (6)$$

This complete kinematic chain, from wheel-level velocities to global pose estimation, forms the foundation for both the PID control law and the odometry feedback employed within the ROS framework. Figure 5 illustrates the geometric arrangement of the four omniwheels and the associated velocity components for a single wheel.

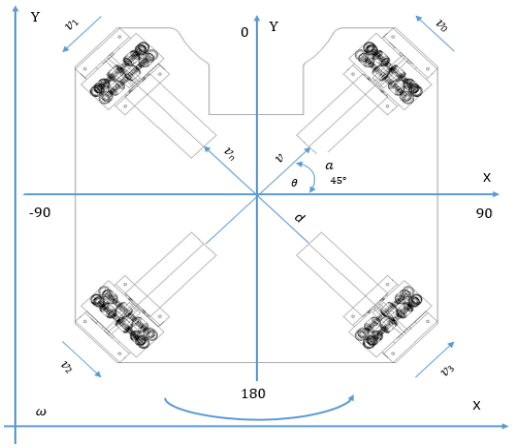


Figure 5. Robot Kinematics

## C. PID Controller Implementation and Tuning Methods

The PID controller is implemented to minimize the error between the target setpoint and the current estimated pose in three independent control axes: X (surge), Y (sway), and $\theta$ (yaw). The control law for each axis is:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)\,dt + K_d \cdot \frac{de(t)}{dt}$$

Controller performance depends on the selection of $K_p$, $K_i$, and $K_d$. Two tuning methods were evaluated: the Ziegler–Nichols closed-loop approach, which determines parameters from the ultimate gain $K_u$ and period $T_u$ and typically yields fast responses with higher overshoot, and the trial-and-error method, which iteratively adjusts $K_p$, $K_d$, and $K_i$ based on observed responses to achieve minimal overshoot and steady-state error.

The trial-and-error tuning method was implemented using a structured and iterative approach to ensure reproducibility. The procedure is defined as follows:

1. Initialize the controller with low proportional gain ($K_p$), while setting $K_i = 0$ and $K_d = 0$.
2. Gradually increase $K_p$ until a fast response is achieved with acceptable oscillation.
3. Introduce derivative gain ($K_d$) to reduce overshoot and improve system damping.
4. Add integral gain ($K_i$) to eliminate steady-state error.
5. Iteratively adjust all parameters based on performance metrics including overshoot, settling time, and RMSE.
6. Select the final parameter set based on the best trade-off between response speed, stability, and accuracy.

Table 1. PID Tuning Results Parameters

| *Tuning Method* | $K_p$ | $K_i$ | $K_d$ |
|---|---|---|---|
| Ziegler Nichols | 1.5 | 0.5 | 0.1 |
| Trial-and-error | 0.5 | 0.1 | 0.01 |

This structured tuning procedure ensures consistency and reproducibility across different trajectory scenarios and real-world conditions.

### D. Simulation Environment using ROS-Gazebo

Prior to hardware deployment, the control system was validated in a virtual environment using ROS Noetic and the Gazebo simulator. The robot model was defined using the Unified Robot Description Format (URDF), incorporating accurate mass (44.4 kg), inertial properties, and friction coefficients ($\mu_{wheel} = 0.8$, $\mu_{roller} = 0.7$). The friction coefficients used in the Gazebo simulation were obtained from literature values for rubber-based wheels operating on flat surfaces and further adjusted empirically to approximate real-world robot behavior.

The simulation environment, shown in Figure 6, includes a flat ground plane and the robot model with articulated omniwheels allowed for safe and repeatable testing of the PID tuning parameters on various trajectory patterns (straight line, L-shape, square, triangle, and complex maneuver). The ROS node architecture facilitates the direct porting of the control algorithm from simulation to the physical STM32F4 firmware.

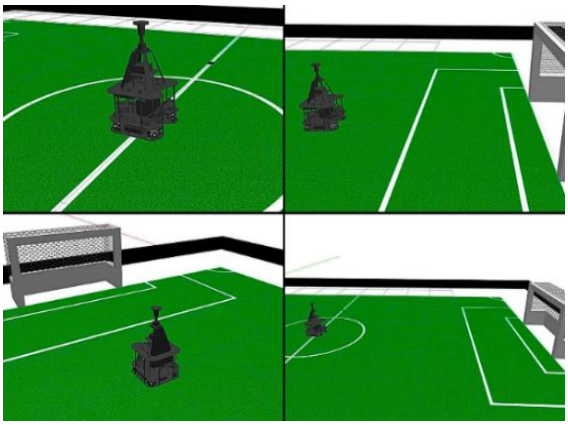

Figure 6. Simulation Environment in Gazebo

## III. RESULTS AND DISCUSSION

### A. Step Response and PID Tuning Performance

The dynamic response of the PID controller was evaluated by applying a step input target velocity of 2.00 m/s. Figure 7 presents the velocity response curves for the three test conditions: simulation (Gazebo), real robot with trial-and-error tuning, and real robot with Ziegler-Nichols tuning.

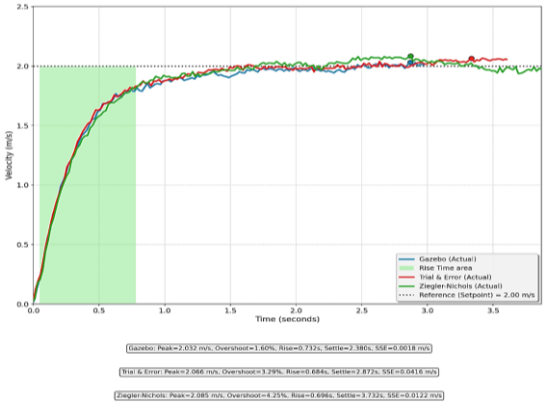


Figure 7. System Response Graphs for the Simulation Using Two Tuning Methods

Table 1 below summarizes the quantitative performance metrics derived from the curves in Figure 7 above.

Table 2. Comparison of Test Results for Gazebo and Two Tuning Methods

| *Testing* | *Peak* | *Over shoot* | *Rise time* | *Settle time* | *SSE* | *RMSE* |
|---|---|---|---|---|---|---|
| Gazebo | 2.032 m/s | 1.60% | 0.732 s | 2.380 s | 0.0018 m/s | 0.4724 m/s |
| Trial and Error | 2.066 m/s | 3.29% | 0.684 s | 2.872 s | 0.0416 m/s | 0.4279 m/s |
| Ziegler–Nichols | 2.085 m/s | 4.25% | 0.696 s | 3.732 s | 0.0122 m/s | 0.4274 m/s |

The superior performance of the trial-and-error method can be attributed to its ability to adapt the PID parameters to the nonlinear characteristics of the real system, including friction, actuator saturation, and sensor noise. In contrast, the Ziegler–Nichols method produces more aggressive gains, leading to higher overshoot due to its reliance on idealized system assumptions. The PID parameters directly influence system dynamics: increasing Kp reduces rise time but may increase overshoot, Ki eliminates steady-state error but may reduce stability, and Kd improves damping and reduces oscillation. These effects are consistent with the observed experimental results.

In real-world implementation, unmodeled dynamics such as wheel slip and communication delay introduce nonlinear disturbances. The trial-and-error method implicitly compensates for these effects through iterative tuning, whereas Ziegler–Nichols assumes a linear time-invariant system, resulting in suboptimal performance.

## B. Trajectory Tracking Accuracy

The robot's ability to follow a predetermined path was tested on five different track configurations: a straight line, an L-shape, a square, a triangle, and a complex maneuvering path. Figure 8 shows examples of the paths the robot will traverse.

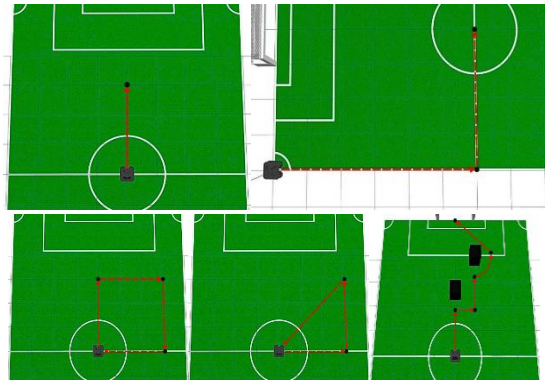

Figure 8. The 5 Test Trajectory Scenarios

The following is a trajectory plot representing the robot's movement and position based on test log data from the Gazebo simulator:

### a. Straight Trajectory

Simulation results in Figure 9 show linear motion from A (160.0, 400.0) to B (900.0, 400.0) with an average velocity of 0.3714 m/s in 8.1 s and high tracking accuracy, indicated by an average deviation of 0.25 cm and a maximum of 1.22 cm.

Real-robot testing in Figure 9 shows that Ziegler–Nichols is faster (0.4325 m/s, 6.8 s) but less accurate (2.72 cm average deviation; 9.07 cm maximum), whereas trial-and-error is slower (0.2021 m/s, 16.1 s) yet more precise (0.88 cm average; 1.41 cm maximum).

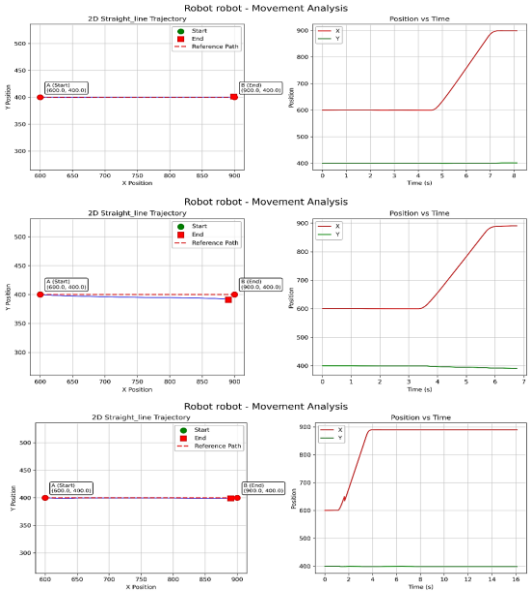


Figure 9. Comparison of Straight-Line Trajectories from Simulations and Two Tuning Methods

### b. L Pattern Trajectory

Simulation results in Figure 10 show L-shaped motion in the 2D plane (x, y) starting from (0.0, 0.0), covering 10.6929 m with an average velocity of 0.5646 m/s in 18.9 s. The final position ranges were X 6.18–667.69 and Y −6.98–468.11, with an average deviation of 5.93 cm, indicating acceptable tracking accuracy despite deviations at turning segments.

Real-robot testing in Figure 10 shows that trial-and-error achieves higher precision and speed (0.7134 m/s, 13.9 s; 71.0% within 10 cm) compared to Ziegler–Nichols (0.5853 m/s, 17.0 s; 45.8% within 10 cm), indicating better performance for the L-shaped trajectory.

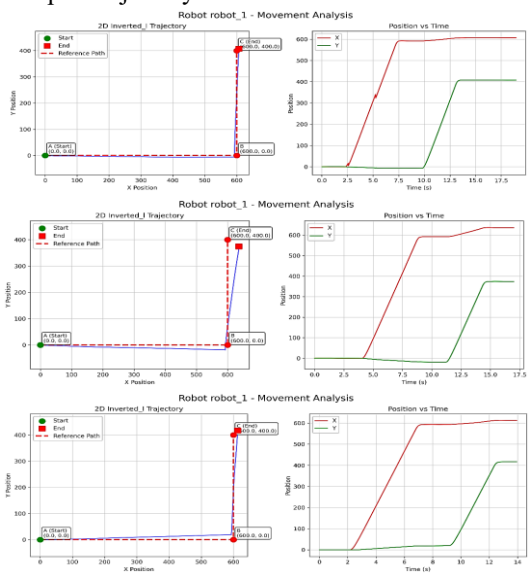


Figure 10. Comparison of L-Pattern Trajectories from Simulations and Two Tuning Methods

**c. Square Trajectory**

Figure 11 shows a 3 m × 3 m square trajectory centered at (758.6, 250.9), with a 12.0703 m path completed in 18.1 s at 0.8345 m/s. The average deviation was 1.41 cm (max 3.96 cm), indicating good tracking accuracy.

Real-robot results indicate Ziegler–Nichols achieves faster motion (0.6940 m/s) but lower accuracy (51.6% within 10 cm), while trial-and-error is slightly slower (0.6545 m/s) yet more precise (97.0% within 10 cm), demonstrating superior square-path tracking.

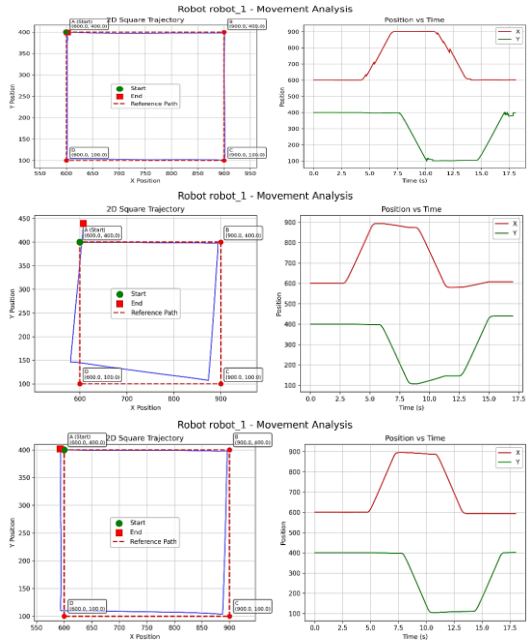


Figure 11. Comparison of Square Trajectories from Simulations and Two Tuning Methods

**d. Triangle Trajectory**

Simulation results in Figure 12 show a right-triangle trajectory (3 m × 3 m) with a total distance of 11.2732 m and an average velocity of 0.7576 m/s in 14.9 s. The average deviation was 1.30 cm with all deviations below the threshold, indicating very high tracking accuracy.

Real-robot testing in Figure 12 shows trial-and-error achieving higher speed and accuracy (0.9691 m/s; 90.3% within 10 cm) compared to Ziegler–Nichols (0.7724 m/s; 73.1%), indicating superior triangular-path tracking.

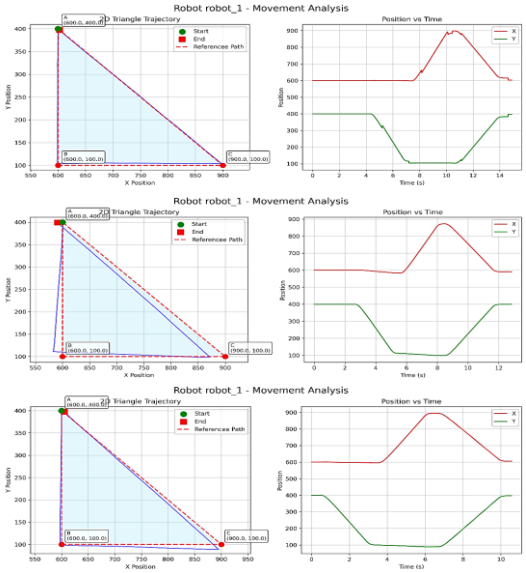


Figure 12. Comparison of Triangle Trajectories from Simulations and Two Tuning Methods

**e. Maneuver Trajectory**

Simulation results in Figure 13 show a complex maneuver trajectory combining linear and circular segments, with a total distance of 8.8218 m and an average velocity of 0.6734 m/s in 13.1 s. The average deviation was 8.06 cm (maximum 25.0 cm), indicating acceptable tracking accuracy for complex motion.

Real-robot testing in Figure 13 shows trial-and-error providing better performance than Ziegler–Nichols, with higher speed (0.4571 m/s vs. 0.3873 m/s) and improved tracking accuracy, indicating more stable control for complex maneuver trajectories.

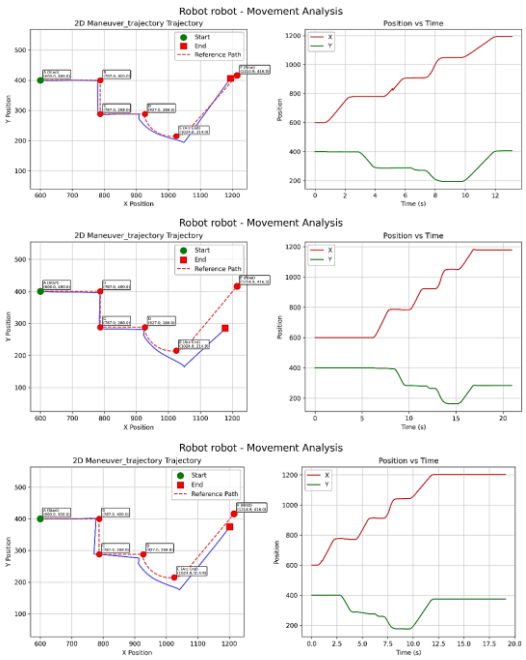


Figure 13. Comparison of Maneuver Trajectories from Simulations and Two Tuning Methods

These results indicate that the trial-and-error tuning method provides more stable trajectory tracking performance across varying motion patterns, particularly in scenarios involving sharp directional changes.

### C. Simulation-to-Reality Gap Analysis

Comparison between the Gazebo simulation and the physical implementation using trial-and-error parameters shows a consistent dynamic trend with measurable deviations. The settling time increased from 2.380 s (simulation) to 2.872 s (actual), while the steady-state error rose from 0.0018 m/s to 0.0416 m/s.

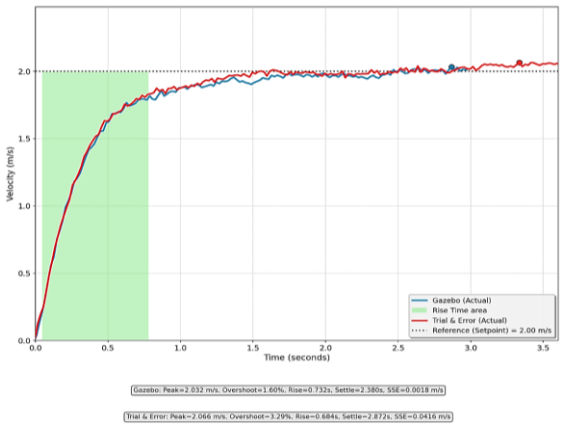


Figure 14. Comparison Chart of Simulation System Responses vs. Actual Robot Testing

Figure 15 compares the simulated and actual trajectories for the square path. The simulation closely follows the reference, whereas the real robot exhibits slight corner drift, mainly due to wheel slip, sensor communication latency, and motor torque variations absent in the ideal model.

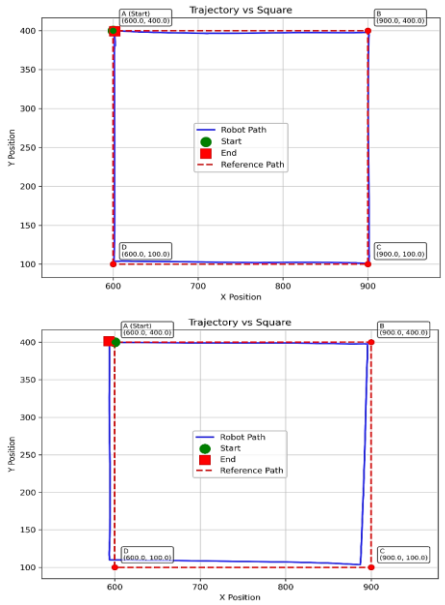


Figure 15. Comparison of Simulated and Actual Test Trajectories

The performance gap between simulation and real-world implementation is primarily caused by unmodeled nonlinearities. This gap can be quantified by the increase in settling time (20.7%) and steady-state error, indicating the limitations of ideal simulation models. This confirms the presence of unmodeled nonlinear dynamics in real-world conditions.

Compared to advanced tuning methods such as Particle Swarm Optimization (PSO) or adaptive PID control, the proposed approach offers lower computational complexity and easier implementation. However, optimization-based methods have the potential to further improve performance by systematically minimizing error metrics, which will be considered in future work.

## IV. CONCLUSION

This paper has presented the design and evaluation of a PID-based position control system for a four-wheeled omnidirectional robot utilizing a ROS-based simulation workflow. The system successfully integrates sensor fusion from MPU6050 and encoders to provide closed-loop feedback. The experimental investigation into tuning methodologies revealed that while Ziegler-Nichols offers a quick analytical starting point, the trial-and-error method yields superior real-world performance due to its adaptability to physical system idiosyncrasies. The PID controller demonstrated a marked improvement over open-loop control, reducing overshoot from 8.85% to 3.29% and lowering the RMSE from 0.7025 m/s to 0.4279 m/s. The robot successfully executed various trajectories with an average deviation of 4.03 cm on square paths. The comparison between simulation and reality highlighted a moderate performance gap driven by environmental friction and sensor noise, yet the consistency in dynamic behavior validates the proposed development pipeline.

Future improvements may include the implementation of Kalman filtering for enhanced sensor fusion and disturbance observers to compensate for wheel slip and external perturbations. Additionally, future work will explore the integration of adaptive tuning algorithms, such as Particle Swarm Optimization (PSO), and the inclusion of LiDAR-based obstacle avoidance to enhance autonomous navigation capabilities. This study contributes not only as an implementation but also as a reproducible experimental framework for evaluating PID tuning strategies on real holonomic robotic systems under practical conditions.

## REFERENCES


[1] Khan, H., Khatoon, S., & Gaur, P. (2024). Stabilization of wheeled mobile robot by social spider algorithm based PID controller. *International Journal of Information Technology (Singapore)*, *16*(3), 1437–1447. https://doi.org/10.1007/s41870-023-01438-w.

[2] Sagita, M. R., Ma'arif, A., Furizal, F., Rekik, C., Caesarendra, W., & Majdoubi, R. (2024). Motion System of a Four-Wheeled Robot Using a PID Controller Based on MPU and Rotary Encoder Sensors. *Control Systems and Optimization Letters*, *2*(2), 257–265. https://doi.org/10.59247/csol.v2i2.150.

[3] Pérez-Juárez, J. G., García-Martínez, J. R., Medina Santiago, A., Cruz-Miguel, E. E., Olmedo-García, L. F., Barra-Vázquez, O. A., & Rojas-Hernández, M. A. (2025). Kinematic Fuzzy Logic-Based Controller for Trajectory Tracking of Wheeled Mobile Robots in Virtual Environments. Symmetry, 17(2). https://doi.org/10.3390/sym17020301.

[4] Fahmizal, F., Pratikno, M. S., Isnianto, H. N., Mayub, A., Maghfiroh, H., & Anugrah, P. (2024). Control and Navigation of Differential Drive Mobile Robot with PID and Hector SLAM: Simulation and Implementation. *Jurnal Ilmiah Teknik Elektro Komputer Dan Informatika*, *10*(3), 594–607. https://doi.org/10.26555/jiteki.v10i3.29428.

[5] Popovici, A. T., Dosoftei, C. C., & Budaciu, C. (2022). Kinematics Calibration and Validation Approach Using Indoor Positioning System for an Omnidirectional Mobile Robot. Sensors, 22(22), 2–22. https://doi.org/10.3390/s22228590.

[6] Iswanto, Ma'arif, A., Raharja, N. M., Supangkat, G., Arofiati, F., Sekhar, R., & Rijalusalam, D. U. (2021). Pid-based with odometry for trajectory tracking control on four-wheel omnidirectional COVID-19 aromatherapy robot. Emerging Science Journal, 5(Special issue), 157–181. https://doi.org/10.28991/ESJ-2021-SPER-13.

[7] Hernández, J. C. O., & Almeida, D. I. R. (2024). Kinematic control in a four-wheeled Mecanum mobile robot for trajectory tracking. The Journal of Engineering, 9, 1–18. https://doi.org/10.1049/tje2.70006.

[8] Marwanto, S. B., & Puriyanto, R. D. (2023). IMU Sensor Based Omnidirectional Robot Localization and Rotary Encoder. Control Systems and Optimization Letters, 1(2), 103–110. https://doi.org/10.59247/csol.v1i2.39.

[9] Purnata, H., Ramadan, S., Hidayat, M. A., & Maulana, I. (2022). PID Control Schematic Design for Omni-directional Wheel Mobile Robot Cilacap State of Polytechnic. Journal of Telecommunication Network, 12(2), 89–94. https://doi.org/10.33795/jartel.v12i2.322.

[10] Tan, X., Zhang, S., & Wu, Q. (2021). Research on Omnidirectional Indoor Mobile Robot System Based on Multi-sensor Fusion. Proceedings - 2021 5th International Conference on Vision, Image and Signal Processing, ICVISP 2021, 111–117. https://doi.org/10.1109/ICVISP54630.2021.00028

[11] Hafidz, C. M., Khairudin, M., Mustakim, W., Andrasto, T., & Yohana, Y. (2025). Four-Wheeled Omni-drive Odometry System on Two-Rotary Encoder and Gyroscope Sensor. AIP Conference Proceedings, 3281(1). https://doi.org/10.1063/5.0261160.

[12] Fuadi, M. M., Zuhrie, M. S., & Anifah, L. (2024). Perancangan Sistem Kontrol Tendangan Berdasarkan Posisi Robot Pada Four Wheeled Omnidirectional Mobile Robot Berbasis Fuzzy Logic Controller. JURNAL TEKNIK ELEKTRO, 13(2), 111–121. https://doi.org/10.26740/jte.v13n2.p111-121.

[13] Gómez-Peñate, S., López Cancino, L. D., Luna, M., Santos Ruiz, I., & Molina-Domínguez, S. D. J. (2024). Trajectory Tracking Control of a Mecanum-Wheeled Omnidirectional Mobile Robot Implemented in ROS-Gazebo. Memorias Del Congreso Nacional de Control Automático, 7(1), 481–486. https://doi.org/10.58571/CNCA.AMCA.2024.082

[14] Apritasari, Y. D., Sudrajat, I., & Wonorahardjo, S. (2023). Experimental Research with Computer Simulation (Case Study of Urban Cool Island). International Journal of Built Environment and Scientific Research, 7(1), 41–50. https://doi.org/10.24853/ijbesr.7.1.41-50.

[15] Ghazal, F., Al-Fatlawi, R., Ahmed, A. M., & Kadhim, R. A. (2025). Framework-Driven PID Auto-Tuning on Mobile Robots Using Bayesian Optimization and Differential Evolution. International Journal of Advanced Robotic Systems, 19(5). https://doi.org/10.1177/17298814241234567.

[16] Afandi, M. A., & Rusimamto, P. W. (2026). Design and implementation of a depth stability control system using a water pressure sensor on an underwater remotely operated vehicle (ROV). Indonesian Journal of Electrical and Electronics Engineering (INAJEEE), 9(1), 14–22. https://doi.org/10.26740/inajeee.v9n1.p14-22

[17] Mauludin, P. B., Rinanton(2025). Simulation of Braitenberg method of differential wheeled robot using Gazebo ROS features. Indonesian Journal of Electrical and Electronics Engineering (INAJEEE), 8(2), 121–126. https://doi.org/10.26740/inajeee.v8n2.p121-126