Jurnal Sains Informatika Terapan (JSIT)
E-ISSN: 2828-1659, Volume: 5, Issue: 2, Month: July, Year: 2026
Page: 703 - 709, Available online at: https://rcf-indonesia.org/home/

ROS2-BASED MOTION CONTROL LEARNING FRAMEWORK FOR DIFFERENTIAL DRIVE MOBILE ROBOTS

Danu Jaya Saputro[1*], Adhitya Sumardi Sunarya[2], Noval Lilansa [3]
[1,2,3] Politeknik Manufaktur Bandung, Indonesia
Corresponding Author: [1]danu@polman-bandung.ac.id

Article Info

Article history:

Received: Jun 12, 2026
Revised: Jun 28, 2026
Accepted: Jul 04, 2026
Published: Jul 14, 2026

Keywords:

Motion control
Differential drive
Mobile robot
Jetbot AI
ROS2

ABSTRACT

This study aims to design and develop a motion control teaching aid for a differential wheel drive mobile robot as a learning medium in robotic courses. The system is implemented using the NVIDIA Jetbot AI platform integrated with the Robot Operating System (ROS2). The research methodology consists of planning, design, and implementation stages, covering wheel control and a preliminary autonomous control scheme. During the design stage, a differential drive control block diagram was developed by integrating trajectory reference input, robot kinematics, motor actuation, and wheel encoder feedback. The system implementation involves the development of multiple ROS2 nodes that communicate through standard topics, along with a web-based interface that supports both manual and autonomous control modes. Initial testing results indicate that the wheel control system provides stable motion response, while the ROS2-based architecture and web interface facilitate real-time monitoring and enhance students' understanding of motion control concepts. This research is expected to support effective learning of motion control and autonomous mobile robot navigation in higher education.

1. INTRODUCTION

The sophisticated technology in mobile robots has significantly helped various human sectors in addressing difficult and hazardous tasks. Several fields of mobile robots application currently exist in transportation, industrial use, and agriculture. These operations can be undertaken by improving the accuracy in navigation, control, remote sensing, and tele-operation [1]. A recent implementation of a mobile robot in the manufacturing field is significantly increasing, as it can deliver parts or materials from one workstation to another. One of its kind is the Automated Guided Vehicle (AGV), which requires precise position and orientation while simultaneously following the determined lines to perform the material handling application [2].

In recent years, the transformation of mobile robots into autonomous mobile robots (AMRs) has depicted a remarkable trend. Several sectors, such as healthcare and industry, are adopting the use of AMRs, prioritizing human-to-machine interaction over human-to-human interaction. In addition, the autonomy of AMRs features enable them to observe their surroundings, assess the outcome according to prior training, make a self-decision, and take corrective actions [3]. More specifically, the architecture of AMRs requires three fundamental aspects, including perception, localization, and navigation, to ensure smooth and adaptive mobility. Some AMRs utilize (Light detection and ranging) LiDAR, inertial measurement units (IMUs), and cameras to capture high-precision environment perception. Furthermore, while Simultaneous Localization and Mapping (SLAM) is generally used to generate environmental maps and locate specific points, the robot also employs a path planning algorithm to traverse to the destinations [4].

A controller of a mobile robot needs to work properly to achieve the predetermined trajectory and avoid tracking problems. The operation is managed by a motion control system, which can be configured by setting the wheels' speed, and the angular velocity while the robot drives from the origin point to curved trajectories. Furthermore, the motion control method relies on the construction of the robot, the environment of the robot's area, and the task that it will carry out [5]. Specifically, a differential drive robot (DDR) mobile robot can run by powering two separate driven wheels and alter the direction of the robot by changing the rotational velocity of either one of its wheels, so that it does not need a steering mechanism to change

the directions [6]. However, there are several factors that can add more complexity to maintaining smooth mobility and avoiding obstacles, such as geometric dimensions, velocity, movement direction, and acceleration. Therefore, it is imperative to organize the correct measure to attain proper functionality [7].

2. RELATED RESEARCH

Motion control is widely used in autonomous mobile robots as the algorithms and techniques to drive the robot's movement in following the planned trajectory. Recent studies have investigated a motion control technique to increase the accuracy of a mobile robot's navigation by developing PID control and tuning. The method generally aims to control the differential drive system, which manages the values of angular speed (ω) and linear speed (v) generated from DC motors on each wheel. The process of PID tuning is essential to obtain the right gain from the input of set point value, KP, KI, KD, displayed in a graph of system response among the set points with feedback. This mechanism is utilized to analyze the effect of constant and speed change to be iteratively corrected and to produce smoother robot movement [8].

Recent literature also reported a study that combined fuzzy logic and PID dynamic model to control a nonholonomic wheeled mobile robot. A nonholonomic mobile robot is constrained by its mechanism and unable to navigate arbitrarily or move sideways. The method introduces a new parameter, A, obtained by incorporating a fuzzy PID and a PID gain-scheduling algorithm. The parameter can deal with several obstacles, like robot oscillations while driving in a curved line. This hybrid algorithm integrates system response state, knowledge base, and fuzzy inference to produce the value of the PID gains in controlling the robot's motion [9].

Moving on to the robotics experiment, [10] employed Gazebo simulation and Robot Operating System (ROS) to model robot's parts, configure collision meshes, and incorporate controllers and sensors in the simulation world. Specifically, a robot model construction needs precise data, including the robot links, weights, and default hardware contained in the Unified Robot Description Format (URDF) description file. One of the most popular URDF modelling language is Xacro (XML Macros), which displays a fine level of similarity between the robot simulation model and its real-world parameters. Another study [11], constructed two types of Simultaneous Localization and Mapping (SLAM) algorithm, which are G-mapping and the Carthographer algorithm, to create a virtual map in robotics simulation. By simulating several coordinate values from multiple measurements in RViz and the Gazebo visualization environment, the two aforementioned algorithms can establish the map in the ROS environment.

The primary objective of this study lies in the integration of a simulated environment for the motion control of a differential drive robot. By modeling a Jetbot AI robot within Foxglove Studio, this research demonstrates precise control over linear and angular velocities, alongside manual navigation via omnidirectional inputs. In autonomous mode, the system is designed to reach specific x and y coordinates while adhering to a defined maximum speed. Additionally, a LabVIEW simulation is utilized to validate line-following accuracy and velocity parameters on a square-shaped path. This work serves as a foundational tool for entry-level students to visualize and comprehend the complexities of robot kinematics and control theory.

3. METHOD

The locomotion of the wheeled mobile robot is highly popular. While the robot maintains good balance, its main concerns are traction, stability, maneuverability, and control. Specifically, in a differential-drive robot, the velocity should be exactly the same to drive both motors on the two wheels. This condition can be challenging, as there are differences among wheels, motors, and environmental situations. To address the issues, it is necessary to determine the foundation of the robot's mechanical system, namely, kinematics. This study pinpoints the mobile robot's design and control algorithm. A proper robot's controllability generates plausible paths and trajectories in its workspace. However, the robot motion is supposed to be redefined over time to acquire the robot's instant position, demanding an accurate motion estimation of the robot. Given the challenging task, robot dynamics also pose extra barriers to the workspace and trajectory due to mass and force considerations.

The proposed methodology encompasses a mobile robot motion control architecture, a differential drive tuning algorithm, and a simulation-based experimental setup to ensure navigational accuracy. Within a Docker environment, a series of shell scripts is configured to manage the communication between ROS nodes, facilitating the activation of the robot's actuators. Wheel odometry data is acquired via optical encoder sensors and processed within the ROS framework, where raw rotation measurements are logged for motion control analysis. To enhance the educational utility and monitor real-time velocity, a specialized web interface is integrated using Foxglove Studio. Furthermore, the proportional gain (Kp) tuning of the differential drive algorithm is validated through LabVIEW simulations, enabling parameter optimization to achieve precise trajectory tracking in line-following scenarios.

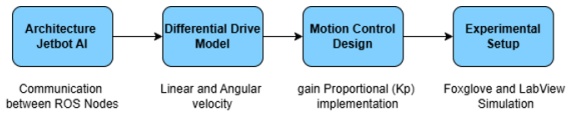

Figure 1. Research Methodology

This research utilized a control system to drive the mobile robot's position and orientation. There are several steps needed to configure the mobile robot control system and framework. The first step is to integrate the mobile robot architecture, which is conducted by employing ROS2 in a Docker container to set up the communication between ROS Nodes. Once the Docker container is set up, a Flask Web application is built as an interface to control the Jetbot in manual and automatic modes. Subsequently, a button in the web interface is connected directly to Foxglove Studio via rosbridge to display real-time visualization. Moving to the kinematics aspect, a Differential Drive Model is arranged to produce linear and angular velocity for the left and right motor. Following this, the motion control system is constructed to produce a proportional gain (Kp) implementation. Finally, the experimental setup runs through Foxglove and LabVIEW simulations to obtain the following trajectory motion.

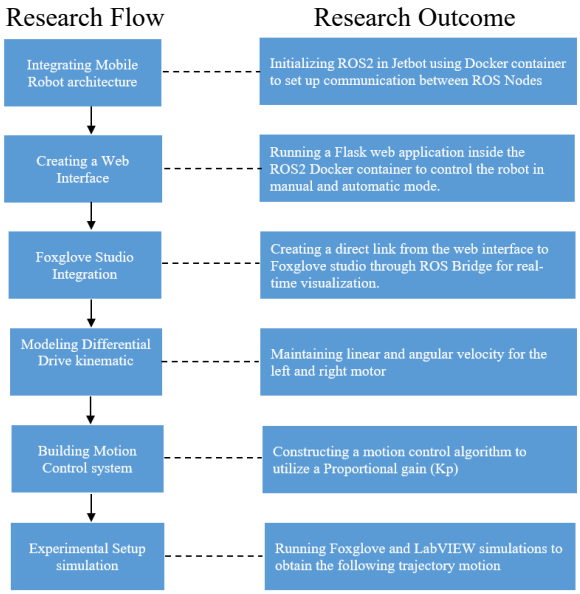

Figure 2. Research Flow

3.1. Mobile Robot Specification

The proposed system employs the NVIDIA Jetbot AI platform as a differential drive mobile robot for educational and motion control experimentation. The robot is equipped with two independently driven DC motors, optical wheel encoders, a Jetson Nano embedded computer, and a monocular camera. The Jetbot platform was selected because of its compact architecture, low-cost deployment, and compatibility with Robot Operating System 2 (ROS2) [12].

The robot operates using a differential drive mechanism in which the robot movement is generated by controlling the rotational velocities of the left and right wheels independently. The linear velocity $v$and angular velocity $\omega$of the robot are mathematically represented as:

$$v = \frac{r}{2}(\omega_r + \omega_l) \quad (1)$$

$$\omega = \frac{r}{L}(\omega_r - \omega_l) \quad (2)$$

where:

r is the wheel radius,

L is the wheelbase distance,

$\omega_r$and ω_l represent the angular velocity of the right and left wheels, respectively.

The robot kinematic model enables the system to translate wheel rotational velocity into translational and rotational robot movement. Table 1 presents the hardware and software specifications utilized in this research.

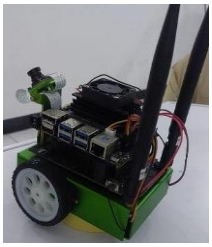

Figure 3. Jetbot

In terms of software deployment, ROS2 Humble was implemented inside a Docker container environment because the Jetbot originally operates on Ubuntu 18.04, while ROS2 Humble officially requires Ubuntu 22.04 compatibility. Docker allows the ROS2 ecosystem to run independently from the host operating system while maintaining software reproducibility and portability across multiple robot platforms.

Table 1. System specification

No	Parameter	Value
1	Jetbot model	NVIDIA Jetbot AI Jetson Nano Kit (B01)
2	Wheel diameter	±65 mm
3	Wheelbase	±120–140 mm
4	Encoder resolution	Optocoupler optical speed Encoder
5	ROS2 version	ROS2 Humble
6	Host PC	Ubuntu 18.04
7	Network Communication	Wi-Fi 802.11ac, Ethernet Gigabit, SSH/WebSocket
8	Update Rate	30 FPS camera/control 10–50 Hz

3.2. Web Interface

A web-based control interface was developed using the Flask framework to simplify robot teleoperation and autonomous navigation. The

application was deployed inside the ROS2 Docker container and connected directly to ROS topics through WebSocket communication [13]. This architecture enables browser-based robot control without requiring additional ROS installation on the client device.

The manual control interface includes:

- virtual joystick control,
- directional arrow buttons,
- linear velocity adjustment,
- angular velocity adjustment.

The joystick command is converted into ROS2 geometry_msgs/Twist messages and published through the /cmd_vel topic. The generated linear and angular velocity values are continuously transmitted to the motion control node for wheel actuation.

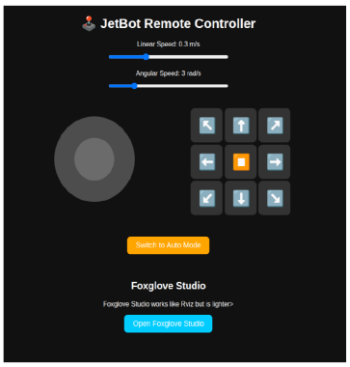

Figure 4. Robot Controller via Web Interface

In autonomous mode, the web interface allows the user to define target coordinates $(x, y)$and maximum translational speed parameters. The autonomous navigation node processes the target position and generates trajectory commands to move the robot toward the desired destination. This feature provides an introductory implementation of coordinate-based navigation suitable for educational robotics applications.

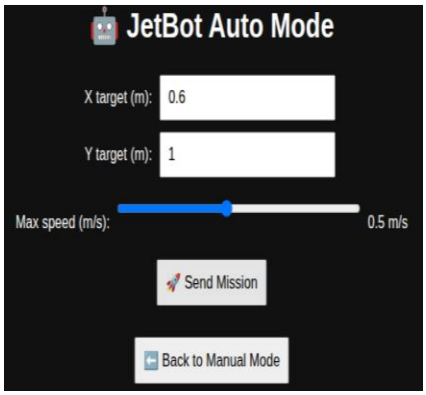

Figure 5. Robot Autonomous Controller via Web Interface

3.3. Foxglove Studio

Foxglove Studio was utilized as a real-time visualization and debugging environment for monitoring robot motion and ROS2 communication. The visualization platform was connected to the ROS2 ecosystem through ROSbridge WebSocket communication, enabling real-time observation of robot states from a web browser [14].

A robot description package containing Unified Robot Description Format (URDF) and mesh files was developed to display the three-dimensional model of the Jetbot inside Foxglove Studio. The visualization environment displays:

- robot orientation,
- wheel movement,
- ROS topic communication,
- velocity parameters,
- robot coordinate frames.

Compared to conventional RViz visualization, Foxglove Studio provides a lightweight browser-based monitoring environment that improves accessibility for beginner robotics learners. The integration also allows simultaneous monitoring between the physical robot and the virtual visualization model in real time.

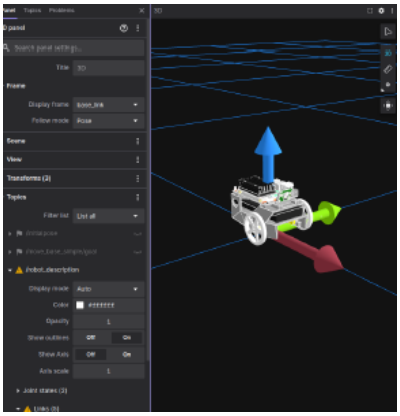

Figure 6. Jetbot Model in Foxglove Studio

3.4. Control block diagram

The motion control system was designed using a differential drive kinematic approach integrated with proportional control. The controller regulates the wheel rotational speed based on the desired linear and angular velocity generated from the web interface [15].

The wheel velocity reference for each motor is calculated using:

$$v_r = v + \frac{L}{2}\omega \qquad (3)$$

$$v_l = v - \frac{L}{2}\omega \qquad (4)$$

where:

v_r= right wheel velocity,
v_l= left wheel velocity,
v= robot linear velocity,
ω= robot angular velocity.

To minimize velocity tracking error, a proportional controller was implemented:

$$u(t) = K_p e(t) \qquad (5)$$

where:
$u(t)$ is the control signal,
K_p is the proportional gain,
$e(t)$ is the velocity error between the desired and measured wheel speed.

Encoder feedback from both wheels is continuously processed to evaluate the wheel rotational speed in revolutions per minute (RPM). The proportional controller adjusts motor throttle values to maintain stable wheel velocity and directional movement during navigation.

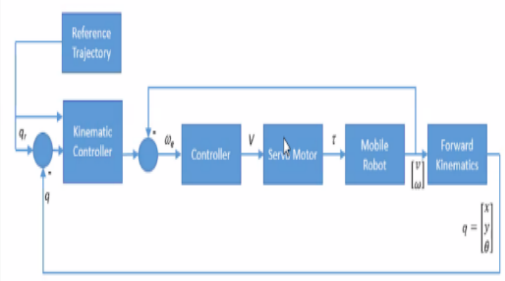

Figure 7. Control block diagram

3.5. Communication between ROS Nodes

The motion control system was designed using a differential drive kinematic approach integrated with proportional control. The controller regulates the wheel rotational speed based on the desired linear and angular velocity generated from the web interface.

The primary ROS2 nodes include:

- web_interface,
- cmd_vel_publisher,
- wheel_encoder_node,
- motor_control_node,
- autonomous_navigation_node,
- robot_visualization_node.

Velocity commands generated from the web interface are published through the /cmd_vel topic and processed by the motion control node. Encoder sensors continuously publish wheel RPM data to feedback topics, which are subsequently used by the proportional controller to adjust motor throttle commands.

The ROS2 communication architecture allows independent subsystem operation while maintaining synchronized robot movement. This modular framework also simplifies debugging and educational observation because students can inspect topic communication directly using Foxglove Studio.

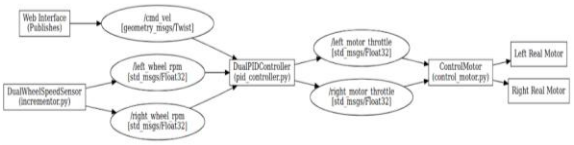

Figure 8. Communication between ROS2 Nodes

4. RESULTS AND DISCUSSION

The experimental implementation successfully demonstrated the integration of ROS2-based motion control, web-based teleoperation, and real-time visualization using the NVIDIA Jetbot AI platform. The robot was capable of performing manual and semi-autonomous navigation while maintaining stable communication among ROS2 nodes inside the Docker environment.

4.1. Experimental Setup and Simulation

The experimental setup combines physical robot implementation and simulation-based validation to evaluate the proposed motion control framework. A LabVIEW simulation was developed to reproduce line-following trajectory behavior similar to the physical differential drive robot.

The simulation evaluates:

- proportional gain tuning,
- directional stability,
- turning response,
- trajectory tracking performance.

A square-shaped trajectory was selected as the testing path because it represents combined straight-line and angular turning motion. During the experiment, the robot velocity response and directional correction behavior were observed under several proportional gain values.

The physical robot movement was simultaneously monitored using Foxglove Studio to compare the consistency between simulated behavior and real-world implementation. The synchronization between simulation and physical response demonstrates that the proposed framework can bridge theoretical robotics learning and practical motion control experimentation effectively.

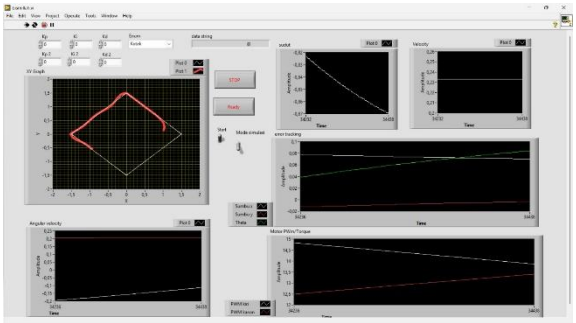

Figure 8. Line following tracking in LabView

The differential drive control system maintained stable wheel rotational velocity through proportional gain adjustment. Encoder feedback successfully generated wheel RPM measurements that were utilized by the controller to regulate motor throttle values. Experimental observations indicated that low proportional gain values produced slower turning

response, while excessively high gain values caused trajectory overshoot during corner navigation.

The LabVIEW simulation successfully reproduced line-following trajectory behavior corresponding to the physical robot movement. The simulation demonstrated that proportional gain tuning significantly affects trajectory smoothness and directional correction stability. The consistency between simulation results and physical robot behavior indicates that the proposed framework can effectively support robotics education through combined virtual and real-world experimentation.

4.2. Web-based system

The web-based teleoperation system successfully generated real-time linear and angular velocity commands through the /cmd_vel topic. During manual operation, the robot responded smoothly to joystick and directional button commands with minimal delay. The generated motion enabled forward, backward, left-turn, and right-turn maneuvers consistently during indoor navigation experiments.

In autonomous mode, the robot successfully navigated toward predefined coordinate targets (x, y)entered through the web interface. The navigation system generated stable directional movement while maintaining the specified maximum translational speed parameter. Although the navigation method did not employ SLAM-based localization, the robot demonstrated acceptable coordinate-reaching capability for educational robotics applications.

The Foxglove Studio integration successfully visualized the Jetbot URDF model, wheel motion, coordinate frames, and ROS2 topic communication in real time. The visualization environment enabled direct observation of wheel RPM feedback, velocity commands, and robot orientation simultaneously. Compared to conventional local visualization tools, the browser-based monitoring system improved accessibility and simplified debugging processes during experimentation.

4.3. Discussion

The results demonstrate that the integration of ROS2, Docker, Foxglove Studio, and web-based teleoperation can establish an effective educational robotics ecosystem for learning motion control systems. Unlike conventional robotics educational platforms that primarily focus on hardware implementation, the proposed framework emphasizes observability, modularity, and reproducibility of robot communication and control processes.

One of the primary contributions of this study is the implementation of a browser-based ROS2 motion control framework. Previous robotics learning environments generally require local ROS installation and Linux-based configuration, which often create technical barriers for beginner students. In contrast, the proposed web interface enables robot control and monitoring directly through a browser, significantly simplifying robotics experimentation and improving accessibility in educational environments.

The use of Docker containers also contributes significantly to system reproducibility. In many robotics implementations, software dependency conflicts frequently prevent consistent deployment across different devices. By encapsulating the ROS2 ecosystem inside Docker containers, this study demonstrates a portable and reproducible robotics environment that can be replicated efficiently on multiple Jetbot platforms without modifying the host operating system.

Another important contribution lies in the integration between physical robot experimentation and simulation-based validation. The LabVIEW simulation reproduces line-following trajectory behavior that resembles the physical Jetbot motion response. This synchronization enables students to compare theoretical control behavior with real-world robot dynamics directly, which improves conceptual understanding of motion control systems and differential drive kinematics.

Furthermore, the modular ROS2 node communication architecture improves educational transparency in robotics learning. Students are able to observe how velocity commands propagate through ROS topics, are processed by motion control algorithms, and eventually generate wheel rotational motion. This differs from black-box educational robots that hide internal communication mechanisms from users.

Despite these promising results, several limitations remain. The autonomous navigation system still relies on predefined coordinates and does not implement advanced localization techniques such as SLAM or sensor fusion. Additionally, the control system currently employs only proportional gain tuning without implementing full PID optimization or adaptive control strategies. Future research should integrate LiDAR-based localization, obstacle avoidance, and quantitative trajectory error analysis to improve autonomous navigation performance and scientific research contributions in mobile robot.

5. CONCLUSION

This study successfully developed a ROS2-based motion control learning framework for a differential drive mobile robot using the NVIDIA Jetbot AI platform. The proposed system integrates differential drive kinematics, ROS2 node communication, browser-based teleoperation, and real-time visualization into a unified educational robotics environment. Experimental implementation demonstrated stable robot movement during manual and autonomous navigation while maintaining

synchronized communication between wheel encoders, motion controllers, and motor actuators.

The integration of Docker, Flask web applications, and Foxglove Studio improved the accessibility, reproducibility, and observability of robotics experimentation. In addition, the LabVIEW simulation produced trajectory behavior consistent with the physical robot implementation, supporting the use of the proposed framework as a bridge between theoretical robotics learning and real-world experimentation.

The main findings of this study can be summarized as follows:

- A ROS2-based modular motion control architecture was successfully implemented on the Jetbot AI platform.
- The web-based interface enabled real-time manual and autonomous robot control through a lightweight browser environment.
- Foxglove Studio successfully visualized robot motion, ROS2 topic communication, and wheel feedback in real time.
- Docker containers improved portability and reproducibility of the ROS2 environment across multiple devices.
- The LabVIEW simulation demonstrated trajectory behavior consistent with the physical robot movement.
- The proposed framework supports beginner-level robotics education by exposing the internal communication and control mechanisms of the robot system.

Although the proposed framework demonstrated promising educational and technical capabilities, the current implementation is still limited to proportional control and coordinate-based navigation. Future work will focus on integrating SLAM, obstacle avoidance, adaptive control algorithms, and quantitative trajectory analysis to improve autonomous navigation performance and scientific contribution.

ACKNOWLEDGEMENTS

Author thanks the Center for Research and Community Service (P3M) of the Bandung Manufacturing Polytechnic for funding the Beginner Lecturer Research Program so that this activity can be carried out properly and smoothly. Gratitude was also expressed to the Department of Manufacturing Automation and Mechatronics for facilitating the equipment used in this study.

REFERENCES

[1] Tamakloe, E., Kommey, B., Addo, E. O., Opoku, D., Tamakloe, E., Kommey, B., Addo, E. O., & Opoku, D., "Mobile Robots and Autonomous Vehicle Control", *A Comprehensive Review of the Advancements and Challenges*, *29*(1). 2025, doi: 10.7454/mst.v29i1.1635.

[2] Saputro, D. J., & Aisyah, S., "Fuzzy Steering Control For Wall-following Behavior Of A Mobile Robot In Webots Simulation", *Jurnal Sains Informatika Terapan (JSIT)*, 347–353, Feb. 2026, doi: 10.62357/jsit.v5i1.1001.

[3] Loganathan, A., & Ahmad, N. S., "Engineering Science and Technology, an International Journal A systematic review on recent advances in autonomous mobile robot navigation" *Engineering Science and Technology, an International Journal*, *40*, 2023, doi: 10.1016/j.jestch.2023.101343.

[4] Gargouri, A., Karray, M., & Zalila, B., "Design and Development of an Autonomous Mobile Robot for Unstructured Indoor Environments". 1–27. *MDPI: machines*, Nov. 2025, doi: 10.3390/machines13111044.

[5] Jadlovský, J., & Kopcik, M., "Basic Motion Control of Differential-Wheeled Mobile Robot ALFRED". *Advances in Intelligent Systems and Computing*, 73–74., 2015, doi: 10.1007/978-3-319-10783-7.

[6] Hilal, W., & Bone, G. M. "Motion Control of a Differential Drive Mobile Robot Considering Voltage and Current Limits". *Advances in Science and Engineering Technology International Conferences (ASET)*, 4, 1–9. 2023, doi: 10.1109/ASET56582.2023.10180654.

[7] Alwan, H. M., Nikolavich, V. A., Shbani, A., & Vladmirovna, K. O., "Motion Control and Obstacle Avoidance of Mobile Robot with Mecanum Wheels". *International Journal of Technology*, 82–96, Jan. 2025, doi: 10.14716/ijtech.v16i1.7254.

[8] Pratikno, M. S., Isnianto, H. N., Mayub, A., & Maghfiroh, H., "Control and Navigation of Differential Drive Mobile Robot with PID and Hector SLAM : Simulation and Implementation. *10*(3), 594–607., 2024, doi: 10.26555/jiteki.v10i3.29428.

[9] Allagui, N. Y., Salem, F. A., & Aljuaid, A. M. "Artificial Fuzzy-PID Gain Scheduling Algorithm Design for Motion Control in Differential Drive Mobile Robotic Platforms", 2021, doi: 10.1155/2021/5542888.

[10] Dobrokvashina, A., Lavrenov, R., Magid, E., Bai, Y., & Svinin, M., "How to Create a New Model of a Mobile Robot in ROS / Gazebo Environment : An Extended Tutorial". 12(4), 192–199., 2023, doi: 10.18178/ijmerr.12.4.192-199.

[11] Cui, Y., & Niu, Z., "Simulation and Implementation of SLAM Drawing Based on ROS Wheeled Mobile Robot". *Journal of Physics: Conference Series*, 8–9, 2021, doi: 10.1088/1742-6596/1865/4/042068.

[12] Nambiar, A. R., Satheesh, A., Anil, G., Anil, D., & Megalingam, R. K., "Design and Implementation of the Platform for Testing a Differential Drive Robot Using ROS2". *Proceedings of the 2025 International Conference on Robotics and Mechatronics (ICRM), IEEE.* doi: 10.1109/ICRM66809.2025.11349105.

[13] Lim, J. M., Kim, K. W., Choi, B. W., & Delgado, R., "A Docker-Enabled Real-Time Framework for Robotic Applications in Heterogeneous ROS 2 Environments". *Processes*, 14(5), 804, 2026, doi: 10.3390/pr14050804.

[14] Cruz, F. R, Walton C, Frye, M, T. "Implementing an Autonomous Navigation Stack in ROS2: Simulation on the Bumperbot Platform", *IFAC-PapersOnLine*, 59(30), 1018–1023, 2024, doi: 10.1016/j.ifacol.2025.12.373.

[15] Champatiray, C., Rout, A., George, A., & George, P. "An experiential learning framework for teaching non-holonomic mobile robot kinematics and control using a ROS 2-enabled differential drive platform". ,2026, doi: 10.1177/03064190261462.