BEST
Journal of Applied Electrical & Science Technology – University of PGRI Adi Buana Surabaya
p-ISSN 2715-2871
e-ISSN 2714-5247

Design of Omniwheel Kinematics Learning Platform Using ESP32 and Microsoft Visual Studio

Sayyidul Aulia Alamsyah
Electrical Engineering Departement
State University of Surabaya
Surabaya, Indonesia
sayyidulalamsyah@unesa.ac.id

Parama Diptya Widayaka
Electrical Engineering Departement
State University of Surabaya
Surabaya, Indonesia
paramawidayaka@unesa.ac.id

Pradini Puspitaningayu
Electrical Engineering Departement
State University of Surabaya
Surabaya, Indonesia
pradinip@unesa.ac.id

Taj Hakam Ikhsan
Electrical Engineering Departement
State University of Surabaya
Surabaya, Indonesia
taj.21033@mhs.unesa.ac.id

Rifando Arya Pradana
Electrical Engineering Departement
State University of Surabaya
Surabaya, Indonesia
rifando.23234@mhs.unesa.ac.id

Ryan Rizky Herdiansyah Palma
Electrical Engineering Departement
State University of Surabaya
Surabaya, Indonesia
ryan.23262@mhs.unesa.ac.id

Ibrohim Zuhayr Achmad
Electrical Engineering Departement
State University of Surabaya
Surabaya, Indonesia
ibrohim.20030@mhs.unesa.ac.id

***Abstract*—Wheeled robots have evolved significantly, starting from simple designs that utilized a single wheel for maneuvering to the present day, where there are numerous types capable of moving in all directions. The increasing complexity of wheeled robots today has created a learning gap for students who are just entering the world of robotics and the advanced robots that currently exist. Based on this problem, a mobile robot learning platform is needed to bridge their knowledge. The design of this platform is expected to help guide students' learning of omniwheel robot kinematics in terms of robot model preparation, robot firmware design, communication design between the PC and the robot, and the implementation of kinematic formulas as robot commands. This platform is also designed to be universal, so it can be implemented for ready-made robots or robots built with any type of microcontroller. Comprehensive testing was performed for both movement modes in the application: forward kinematics and inverse kinematics. The forward kinematics test was carried out by assigning speed values to each motor individually. The inverse kinematics test was cariied out by assigning target position in cartesian plane. This platform is expected to serve as a valuable tool in the study of omniwheel robot kinematics.**

Keywords—Omniwheel, Robots, Kinematics, PID.

I. INTRODUCTION

In this modern era, robots have been widely used in our daily lives. We can easily encounter robots in industry, education, entertainment, and households. These robots have become tools that make work much easier, especially in the automation of various jobs [1]. For example, floor-cleaning robot are currently widely used for cleaning household floors [2].

Mobile robot can be classified as legged robot and wheeled robot [3]. Mobile robot is wheeled robot that using wheels to move and navigate. This type of mobile robot employs motors to drive the wheels, enabling it to move in all directions [4]. Wheeled robots are widely used as robot moving mechanisms because their movement is relatively faster than legged robots. There are various type of wheeled robots that have been studied, starting from those that use two wheels [5] to four wheels. The types of movement are also varied, some use Ackerman arcs like cars, some use omniwheels or mecanum wheels to move in all directions [6]. The floor-cleaning robot mentioned earlier use movements that utilize omniwheels or mecanum wheels which provide the advantage of being able to move in all directions.

Research on wheeled robots has been extensively conducted and has become highly complex. Wheeled robots have evolved significantly, starting from simple designs that utilized a single wheel for maneuvering to the present day, where there are numerous types capable of moving in all directions [7]. The increasing complexity of wheeled robots today has created a learning gap for students who are just entering the world of robotics and the advanced robots that currently exist. This issue requires a platform for learning about mobile robots, not only through simulations but also by observing how robots actually move and operate in real-world scenarios [8]. Based on this problem, a mobile robot learning platform is needed to bridge their knowledge.

In fact, there are already many application platforms that provide an understanding of kinematics. For Example, researcher in [9] using virtual reality, remote control and on-site laboratory for teaching 6-DOF robotics kinematics. The researcher in [10] intoduces BulletArm as a novel open-source framework for robotic manipulation learning for robot arms. Many articles discuss the CoppeliaSim-based robotics learning platforms, CoppeliaSim is one of the well-known simulation-based robotics learning platforms. In [11], the researcher discuss about the application of CoppeliaSim in robot course teaching and the advantage using it such as reduces costs and security risks. In [12] , the author introduce the new open-source

p-ISSN 2715-2871
e-ISSN 2714-5247

simulator based on CoppeliaSim and ROS for the Franka Emika Robot (FER). Besides Coppelia, the Gazebo and ROS2 pair is currently the most widely used robotic platform. Article in [13] discuss about design and implementation of 6DOF robot simulation system in ROS-Gazebo, including robot modelling, trajectory planning, and control design. Author in [14] presented a toolkit for mobile robotics learning via extending the MATLAB/Simulink using ROS and Gazebo simulator. Almost all references regarding robotics learning platforms are based on simulations, but in reality, simulation alone is not enough. Students are also required to understand the components that enable a robot to move. This is related to how robot kinematics are formulated, which involves variables such as the robot's dimensions, the selected motor speed, the robot's position, and so on. The ideal scenario would be for each student to be assisted in assembling a robot. However, this is hindered by the high cost of producing even a single type of mobile robot, which is generally expensive [15]. In addition to being constrained by costs, making an ideal robot for learning about kinematics takes a long time. This will burden students to learn about robot kinematics. In addition, none of these references have yet created a dedicated platform for learning robot kinematics, especially for omniwheel mobile robot.

The objective of this research is to develop a learning platform for kinematics, limited to the three-wheeled omniwheel robot type. The design of this platform is expected to help guide students' learning of omniwheel robot kinematics in terms of robot model preparation, robot firmware design, communication design between the PC and the robot, and the implementation of kinematic formulas as robot commands. This platform is also designed to be universal, so it can be implemented for ready-made robots or robots built with any type of microcontroller.

II. METHOD

The proposed omniwheel kinematics learning platform design consists of several stages: robot model design, robot firmware design, robot-PC communication design, and finally, kinematics implementation as robot commands. All stages of this omniwheel robot kinematics learning are created as GUI application using Microsoft Visual Studio 2022 and run according to the flowchart in the figure 1.

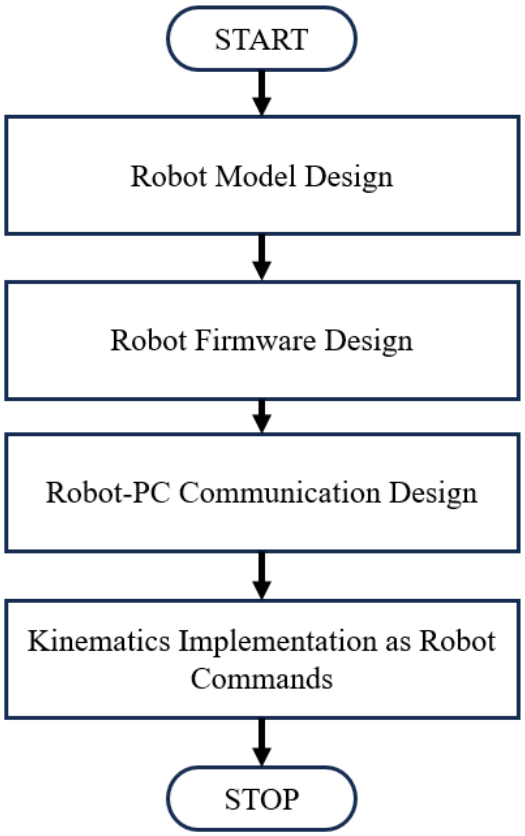

Fig. 1. Proposed Omniwheel Kinematics Learning Platform Stages

A. Omniwheel Robot Model Design

Omniwheels, or omnidirectional wheels, refer to wheels utilized in robotic systems that allow movement in all directions. A robot equipped with such wheels can move along three motion dimensions: x, y, and ω. The x and y dimensions represent translational movement on a planar surface, while ω denotes the robot's rotational orientation (heading). Kinematic modeling is essential to coordinate the simultaneous movement of multiple omniwheels, enabling motion along the x, y, and ω axes. The position and physical dimensions of the robot significantly influence the kinematic model of an omniwheel robot. In this study, the model used is a three-wheeled omniwheel robot, as illustrated in Figure 1, where each wheel arm is separated by an angle of 120°.

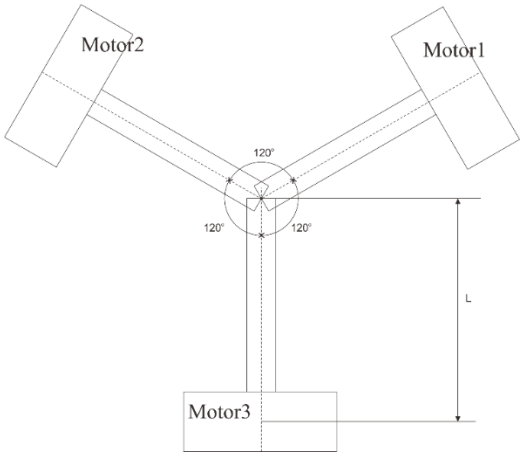

Fig. 2. Omniwheel Model

The kinematics of an omniwheel robot can be derived by breaking down the direction of movement generated by each motor. Once the influence of each motor on the robot's body motion is determined, the overall motion of the robot along the x-axis (v_x), y-axis (v_γ), and its angular movement (ω) can be formulated as follows:

p-ISSN 2715-2871
e-ISSN 2714-5247

$$v_x = -v_{1x} - v_{2x} + v_{3x} = -\frac{v_1}{2} - \frac{v_2}{2} + v_3 \dots\dots\dots (1)$$

$$v_y = v_{1y} - v_{2y} = \frac{v_1\sqrt{3}}{2} - \frac{v_2\sqrt{3}}{2} \dots\dots\dots (2)$$

$$\omega = \omega_1 + \omega_2 + \omega_3 = -v_1 \cdot l - v_2 \cdot l - v_3 \cdot l \dots\dots\dots (3)$$

Alternatively, this can be represented in matrix form:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & 1 \\ \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 \\ l & l & l \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Equations (1), (2), and (3) are referred to as the forward kinematics equations. These equations provide the body velocity of the robot in the *x, y*, and *ω* dimensions as the output, with the input being the individual motor velocities. Another form of kinematic modeling is the inverse kinematics. In this context, the inverse kinematics equations are defined as equations that yield the required individual motor velocities based on the desired robot body movement along the *x, y,* and *ω* axes.

The inverse kinematics can be derived through matrix operations on the forward kinematics equations. By denoting the forward kinematics in matrix form as follows:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} & 1 \\ \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 \\ l & l & l \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\boldsymbol{A_{3\times1} = B_{3\times3} \cdot C_{3\times1}}$$

Multiplying both sides by the inverse of matrix B(3×3), we obtain:

$$\boldsymbol{C_{3\times1} = B^{-1}_{3\times3} \cdot A_{3\times1}}$$

By inverting matrix $\boldsymbol{B_{3\times3}}$ using the known arm length l = 10.5158 cm, the inverse kinematics equations are derived as follows:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -0.333 & 0.5774 & 0.0317 \\ -0.333 & 0.5774 & 0.0317 \\ 0.6667 & 0 & 0.0317 \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

These forward and inverse kinematics equations are subsequently implemented in the graphical user interface (GUI) to issue motion commands to the robot body.

The kinematic breakdown process is intended to provide students with a comprehensive understanding of the key parameters that influence robot kinematics. In the case of a three-wheeled omniwheel robot, critical parameters include the angle between the wheel arms and the length of the wheel arms, both of which exert a significant impact on the robot's kinematic behavior.

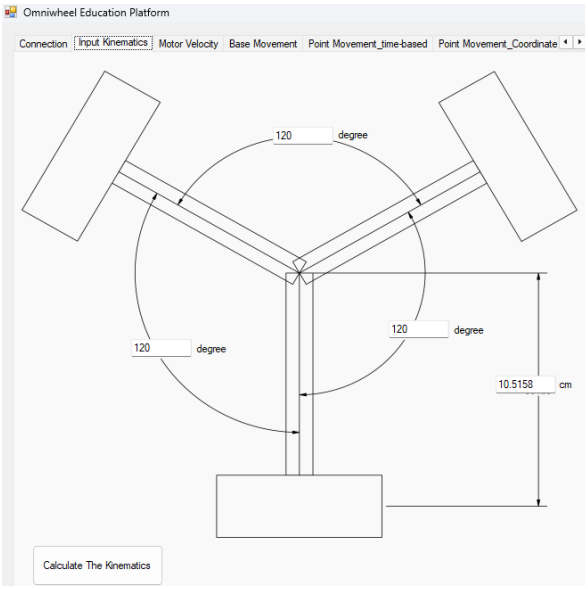

Fig. 3. GUI Initization View

These parameters are subsequently integrated into the application to assist students in deriving the omniwheel kinematic equations. Figure 3 shows the Initialization interface of the application, which contains settings related to the physical configuration of the omniwheel robot. Once the values are provided, the user can click "Calculate the Kinematics" to generate the matrices for both forward kinematics and inverse kinematics.

B. Omniwheel Robot Firmware Design

The omniwheel robot in this study was build using ESP32 as the main microcontroller, L293N motor driver, and motor equipped with encoders. Based on the kinematic modelling, the motion of the omniwheel robot is highly dependent on the the speed of each wheels. This makes the design of reading the motor speed and controlling the motor speed the most prioritzed. Figure 4 show the ilustration of the robot motion that relies on three wheels controlled by microcontroller through the L293N motor driver. The motor placement is adjusted to the omniwheel model in figure 2.

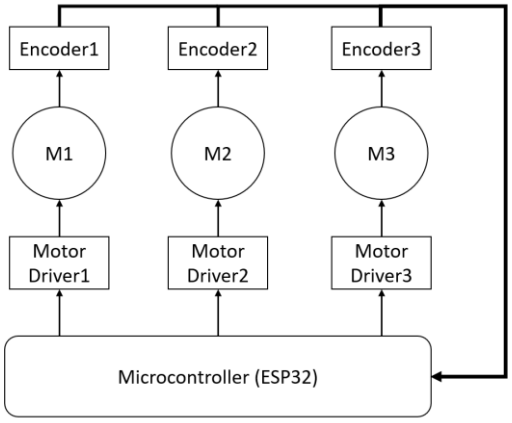

Fig. 4. Motor Controlling Block Diagram

p-ISSN 2715-2871
e-ISSN 2714-5247

Accurate measurement of motor speed is closely linked to the encoder sensors used. The encoder readings must be calibrated to the wheel size in order to calculate the correct linear speed. By utilizing interrupts on the ESP32, changes in encoder signals can be handled as a priority process in the program algorithm. To determine the linear velocity, the angular velocity of the motor must first be computed. The angular velocity is obtained using the following equations.

$$RPM = \frac{n_{1s}}{1s} \times \frac{1}{p} \times 60s = \frac{n_{1s}}{p} \times 60 \ldots\ldots\ldots (1)$$

$$\omega\left(\frac{rad}{s}\right) = \frac{RPM \times 2\pi}{60} \ldots\ldots\ldots (2)$$

$$V(m/s) = \omega \times r \ldots\ldots\ldots (3)$$

Equation (1): Calculates revolutions per minute (RPM) based on the number of pulses detected by the encoder per second (n_{1S}) and the number of pulses per revolution (p). Equation (2): Converts RPM to angular velocity in radians per second (rad/s). Equation (3): Converts angular velocity to linear speed, where r represents the wheel radius in meters.

In applying Equation (1) to the microcontroller, a modification was introduced to improve responsiveness. The traditional method of sampling pulses every 1 second is considered too slow, especially given the robot's relatively small diameter (5.8 cm). Such a delay would reduce the robot's responsiveness to velocity changes. To address this, encoder pulses are sampled every 50 milliseconds, resulting in a modified equation:

$$RPM = \frac{n_{50ms}}{50ms} \times \frac{1}{p} \times 60s = \frac{n_{50ms}}{p} \times 1200 \ldots\ldots\ldots (4)$$

Equation (4): Uses n_{50ms}, the number of pulses recorded within 50ms, and adjusts the calculation accordingly so the RPM output remains normalized to one minute.

An additional adjustment is made to the units of linear velocity, which are converted to centimeters per second (cm/s) to simplify command inputs. Accordingly, the wheel radius r in Equation (3) is also expressed in centimeters.

Motor speeds are controlled via Pulse Width Modulation (PWM) signals generated by the ESP32. A PID control system is implemented to adjust the PWM output and achieve the desired linear speed setpoints for each motor. The digital PID control system used is defined as:

$$u = K_p e + K_i . \sum e + K_d \Delta e$$

The setpoint is the target speed, the feedback is the encoder-measured speed, and the PID output is the PWM signal sent to the motor.

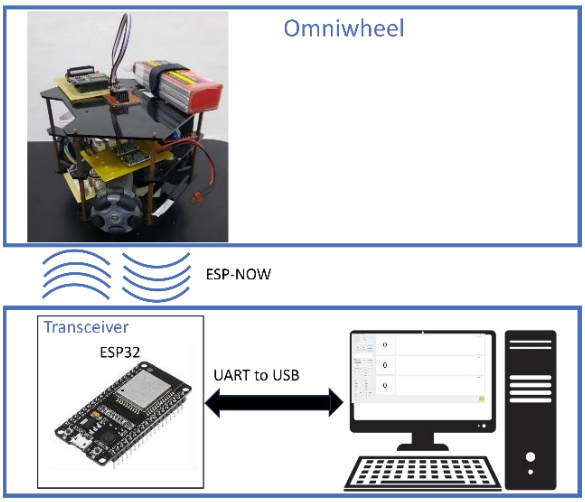

Fig. 5. Robot-PC Communication

C. *Robot-PC Communication Design*

Omniwheel robots utilize the ESP-NOW wireless communication protocol system that is already integrated with ESP32. ESP-NOW is a communication protocol between Espressif products that utilizes a wireless-fidelity network. This protocol requires the MAC-Address of each device to be able to communicate with each other. This communication is used to send linear speed data for each motor, and also receive commands for various omniwheel operating modes. Figure 5 illustrates the communication between Robot and PC.

In this article, the term transceiver refers to the device connected to the computer. Its main function is to relay data between the omniwheel robot and the computer. The transceiver was built using an ESP32 microcontroller, which allows it to communicate wirelessly using the same ESP-NOW protocol as the robot. Data received from the robot is forwarded to the computer through serial communication, which is natively supported by the ESP32 and facilitated via a USB to Serial CH304 interface.

By default, the omniwheel robot is configured to transmit motor speed data every 200 ms. The target setpoints for each motor, however, are determined by data sent from the computer via the transceiver. This data is formatted as a string containing 10 comma-separated values, including specific command codes used to activate various robot operation modes. The robot supports two primary modes: (1) Configuration Mode (Setting Mode) and (2) Operation Mode (Running Mode).

D. *Kinematics Implementation as Robot Commands*

The formulation of the kinematic equations in the initial stage is subsequently implemented as the foundation for issuing robot commands from the PC. Two modes of command are provided in accordance with the kinematic models: forward kinematics, which is executed through the Base Movement menu, and inverse kinematics, which is executed through the Point Movement menu.

p-ISSN 2715-2871
e-ISSN 2714-5247

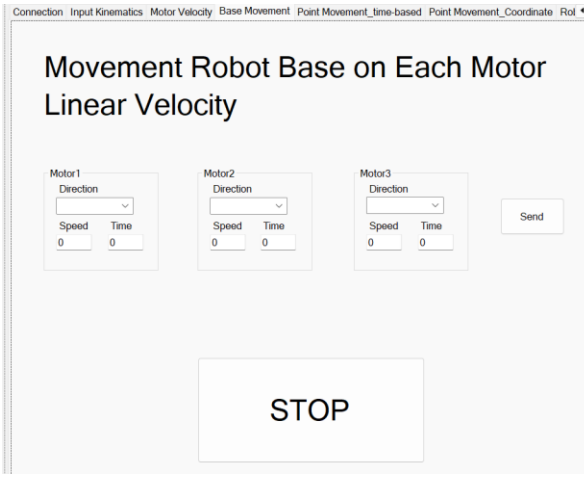

Fig. 6. Base Movement View

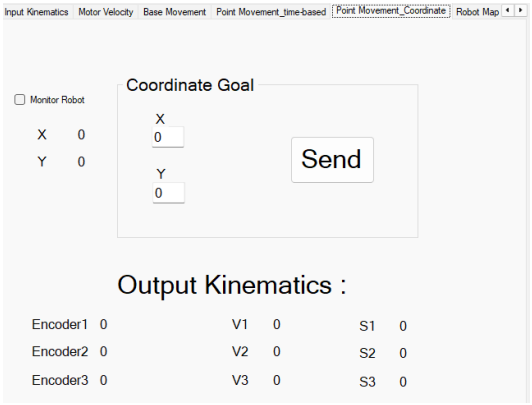

Fig. 8. Point Movement Coordinate

Base Movement

The movement of the omniwheel robot is governed by the velocity of each individual motor. The Base Movement page provides a simplified interface for sending manual velocity commands to each motor, effectively implementing the forward kinematics model. Within this page, users can specify the desired speed, rotation direction, and duration for each motor. Figure 6 shows the Base Movement menu.

Point Movement

This interface implements the inverse kinematics commands for the omniwheel robot, where users input the desired body velocity in Cartesian coordinates. Using the inverse kinematics formulas based on the parameter values provided during the Initialization, the application converts the robot's body velocity into individual motor velocities. These motor velocities are then transmitted to the robot via the transceiver.The Point Movement module consists of two sub-modes: Time-Based and Coordinate-Based. Figure 7 illustrates the Time-Based mode, which allows users to input body velocity components (x, y, and ω) along with the desired movement duration. Figure 8 shows the Coordinate-Based mode, where users can input target coordinates in centimeters, and the robot will attempt to move toward that point.

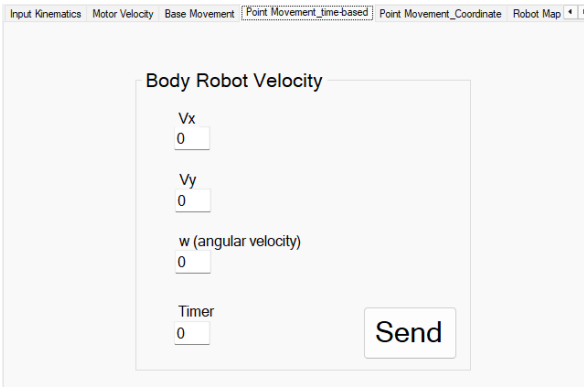

Fig. 7. Point Movement Time Based

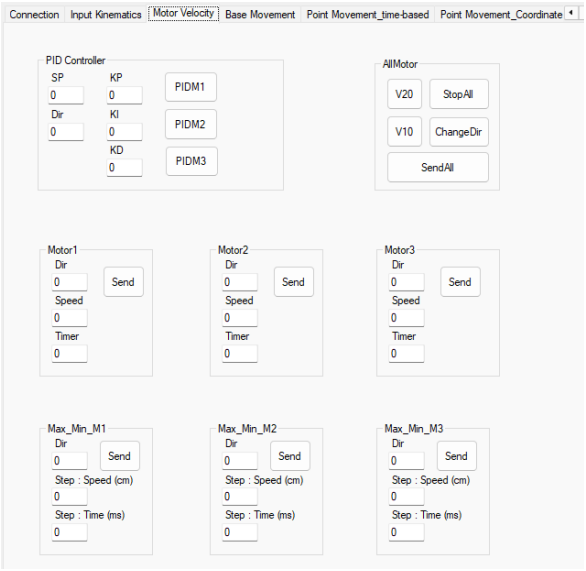

Fig. 9. PID Setting View

III. Result and Discussion

The results presented in this section are the outcomes of constructing the omniwheel robot based on the proposed stages. PID testing was conducted initially to obtain optimal speed control for each motor. Comprehensive testing was performed for both movement modes in the application: forward kinematics and inverse kinematics. The forward kinematics test was carried out by assigning speed values to each motor individually. The inverse kinematics was evaluated by assigning target position commands in Cartesian space, after which the robot's movements were recorded and subsequently analyzed through background subtraction applied to the recorded video.

Each motor was operated with a specific speed for a limited time duration. These tests generated speed response graphs for each motor over time. The results helped validate whether the motors could maintain the given speed consistently under open-loop conditions. Inverse kinematics testing involved commanding the robot's body to move in various directions in the Cartesian plane, represented by specific values for x and y velocity

p-ISSN 2715-2871
e-ISSN 2714-5247

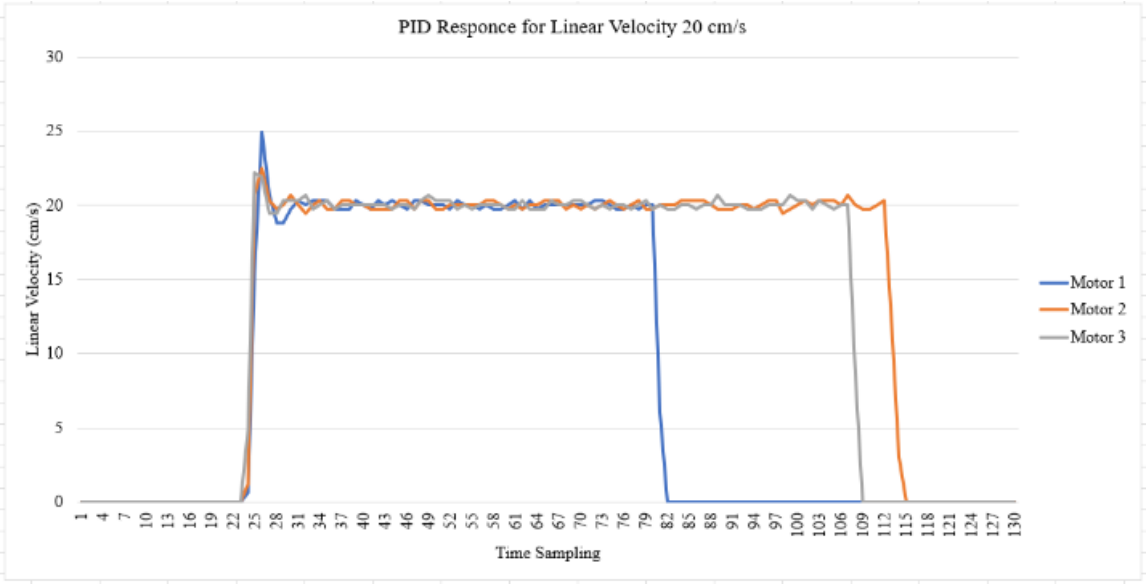

Fig. 10. Motor Response Set Point 20cm/s

inputs. The system then converted these body velocity inputs into individual motor speed set points using the inverse kinematics equations.

Initial PID tests were performed with the motor operating under minimal load—only the wheel was attached. Figure 9 shows the display for PID testing in the application. After multiple iterations, the optimal PID parameters were determined to be: Kp = 2 ; Ki = 0.1 ; Kd = 0.01. These values were applied uniformly across all three motors. Figure 10 illustrates the recorded motor responses to a 20 cm/s set point. Table 1 summarizes the performance metrics of each motor including Overshoot, Rise Time, Settling Time, and Steady-State Error. The overshoot values among the three motors showed slight variation but were relatively close. The rise time for Motor 1 was approximately 200 ms slower than Motors 2 and 3, primarily due to the 200 ms data transmission interval, not motor behavior itself. Therefore, the rise times are still considered equivalent. A similar pattern was observed for the settling times. The steady-state errors were also similaracross all three motors. From these observations, it can be concluded that all three motors behave consistently, allowing the same PID parameters to be effectively applied to each.

Table 1. PID Response for Each Motors

	Overshoot	Rise Time	Settling Time	Steady-State Error
Motor1	24.4 %	<600ms	1.2 s	0.395 cm/s
Motor2	12.08 %	<400ms	1 s	0.340 cm/s
Motor3	10.67 %	<400ms	1 s	0.250 cm/s

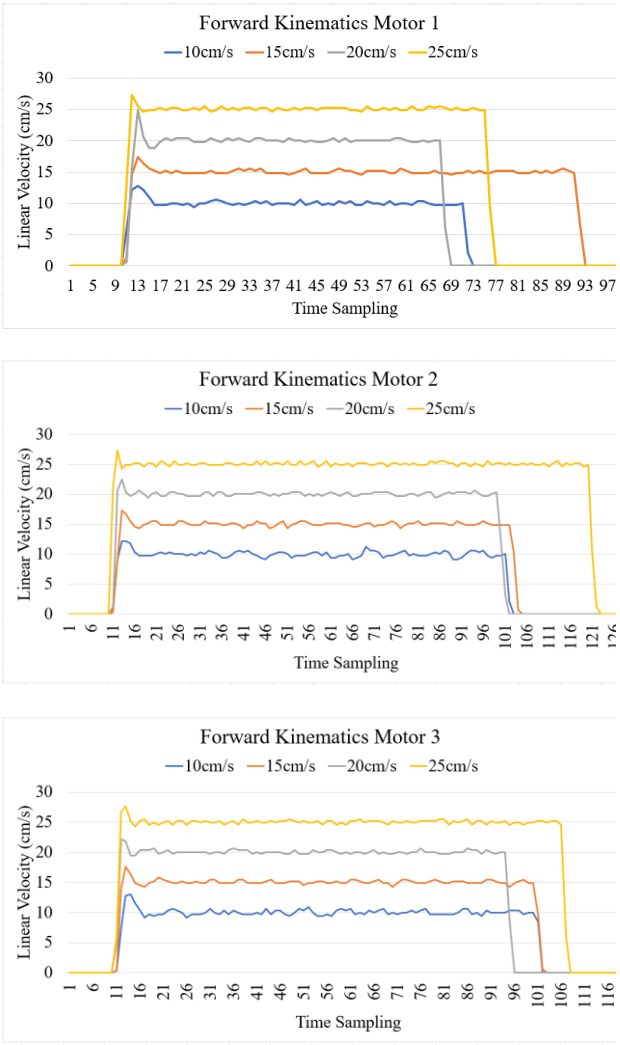

Fig. 11. Forwad Kinematics Through All 3 Motors

p-ISSN 2715-2871
e-ISSN 2714-5247

The next stage of testing evaluated the entire system under real-world conditions. The robot was placed on the floor, introducing additional load from the robot body. Tests were conducted by assigning speed set points individually to each motor with predefined durations. Four different speeds were tested for each motor: 10 cm/s, 15 cm/s, 20 cm/s, and 25 cm/s. Figure 11 presents the speed responses of all three wheels for each set point. The results indicate that, even under load, the motors were able to maintain the target speeds accurately, validating the effectiveness of both the PID controller and the kinematic model.

The final set of tests involved inverse kinematics-based movement. The input to the application was the desired body velocity in the x and y directions. These inputs were then translated into a speed set point for each motor using the inverse kinematics formula. The outcome of these tests was recorded on video. The video footage was then processed using background subtraction

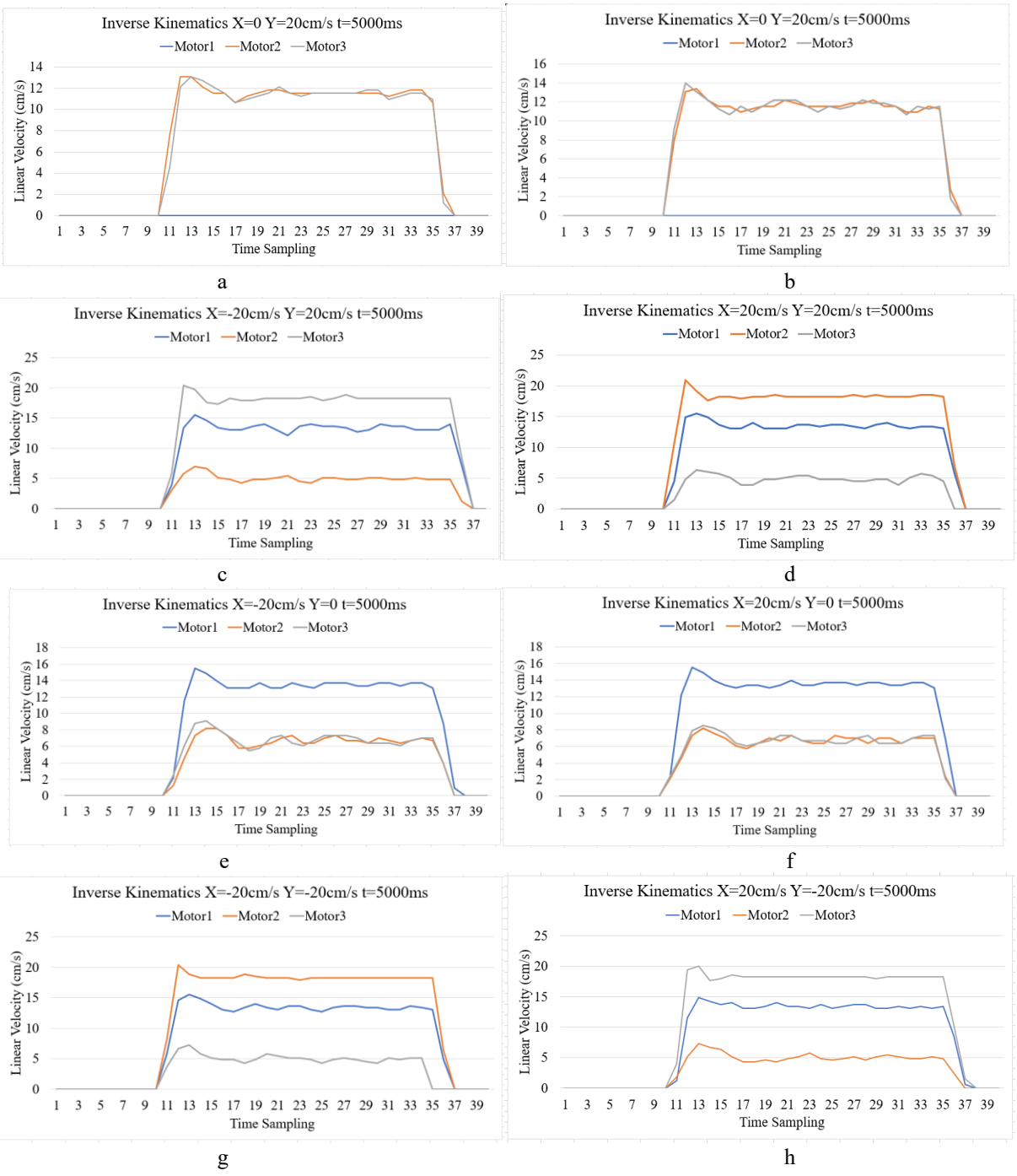

Fig. 12. Motor Velocity Response Based GUI

p-ISSN 2715-2871
e-ISSN 2714-5247

techniques implemented in OpenCV and Python to extract the robot's movement trajectory.

Testing was conducted eight times based on table 2. Figure 12 shows the speed change of each motor for each movement command. Figure 13 displays the processed trajectory output using the background subtraction method. The robot's path is marked with red lines, starting from a small red circle and progressing toward a larger one. The frame numbers are annotated on the trajectory to indicate progress over time. The results demonstrate that the robot moved accurately in accordance with the commands outlined in Table 2, confirming the correctness of the inverse kinematics implementation.

Compared to other robotics learning platforms, the proposed kinematics learning platform is more technical in nature. The platform provides a series of technical steps

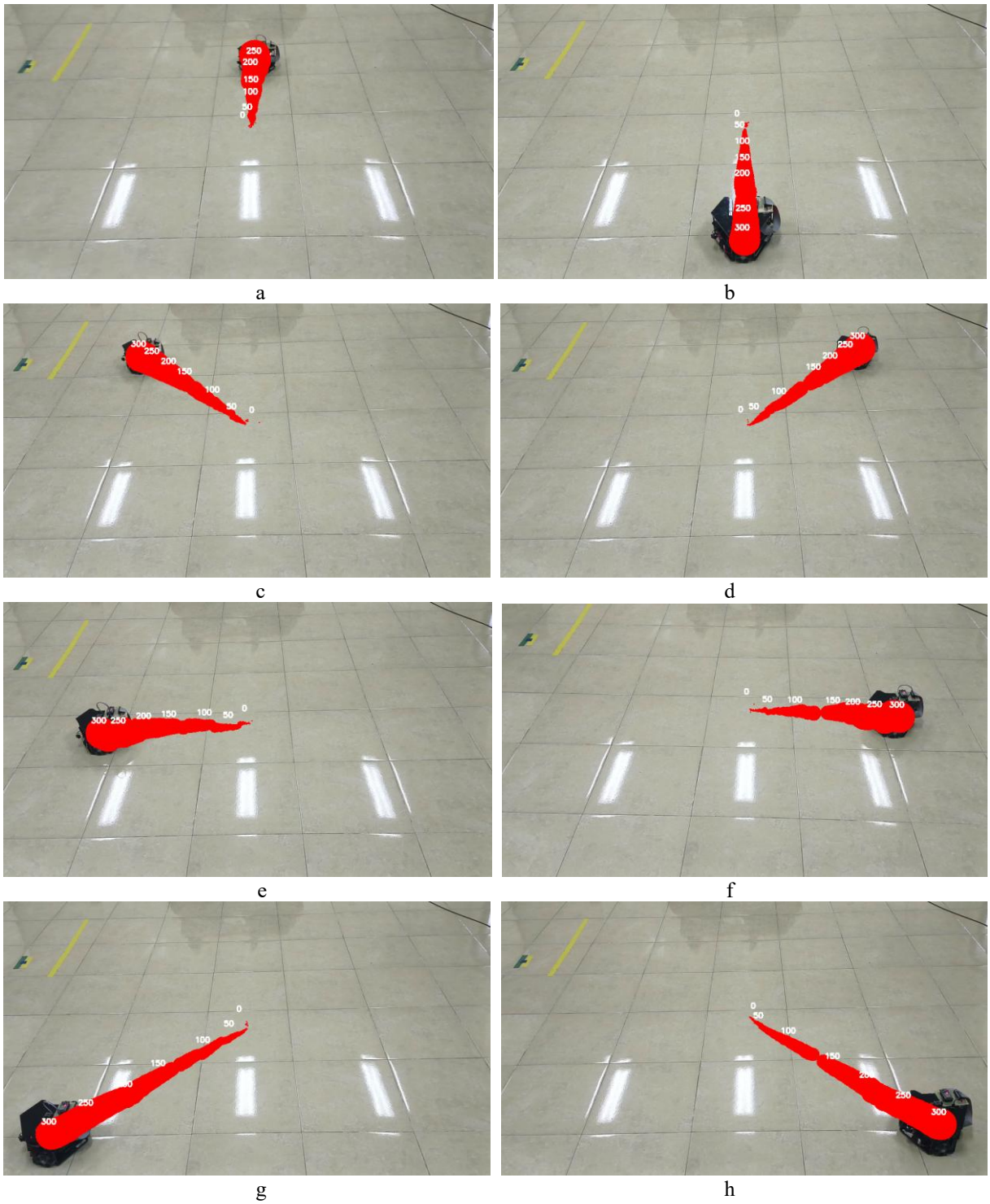

Fig. 13. Response Motor Velocity Base on GUI Command

p-ISSN 2715-2871
e-ISSN 2714-5247

Table 2. Inverse Kinematics Command

CMD	Vx	Vy	Graph	Movement
1	0	20cm/s	Figure 16.a	Figure 17.a
2	0	-20cm/s	Figure 16.b	Figure 17.b
3	-20cm/s	20cm/s	Figure 16.c	Figure 17.c
4	20cm/s	20cm/s	Figure 16.d	Figure 17.d
5	-20cm/s	0	Figure 16.e	Figure 17.e
6	20cm/s	0	Figure 16.f	Figure 17.f
7	-20cm/s	-20cm/s	Figure 16.g	Figure 17.g
8	20cm/s	-20cm/s	Figure 16.h	Figure 17.h

for learning robot kinematics in a universal manner, enabling students to independently apply them to self-built robots. Unlike ready-made commercial robots that must be purchased, this platform allows students to design and develop their own robots from scratch. In addition, the platform includes software that facilitates communication between the PC and the robot. However, the proposed system is limited in that it can only be implemented on the Windows operating system and is restricted to three-wheeled omniwheel robots.

IV. CONCLUSION

A learning platform for an omniwheel robot has been successfully developed using the ESP32 microcontroller and a custom application built with Microsoft Visual Studio 2022. Experimental results validate that the system functions effectively, establishing reliable communication between the GUI application and the omniwheel robot. Both forward kinematics and inverse kinematics were successfully implemented within the application and transmitted to the robot via the ESP-NOW wireless protocol. These results demonstrate the feasibility of the system for real-time robotic control and kinematic visualization. In contrast to other robotics learning platforms, the proposed kinematics platform adopts a more technical approach. Rather than relying on pre-assembled commercial robots, it enables students to design and construct their own robots from the ground up. This platform is expected to serve as a valuable tool in the study of omniwheel robot kinematics.

Future development of the system may include the integration of path planning and local positioning systems, which would enable a more comprehensive and advanced robotics learning platform.

ACKNOWLEDGMENT

We would like to say thank you for the support to Lembaga Penelitian dan Pengabdian Kepada Masyarakat Universitas Negeri Surabaya for the research funding.

REFERENCES

[1] O. Ikpeze, T. Ejidokun, and M. Onibonoje, “Smartphone Control Mobile Robot for Education and Research,” *J. Robot.*, vol. 2022, pp. 1–10, Oct. 2022, doi: 10.1155/2022/5178629.

[2] Y. Irawan, M. Muhardi, R. Ordila, and R. Diandra, “Automatic Floor Cleaning Robot Using Arduino and Ultrasonic Sensor,” *J. Robot. Control JRC*, vol. 2, no. 4, 2021, doi: 10.18196/jrc.2485.

[3] R. T. Yunardi, D. Arifianto, F. Bachtiar, and J. I. Prananingrum, “Holonomic Implementation of Three Wheels Omnidirectional Mobile Robot using DC Motors,” *J. Robot. Control JRC*, vol. 2, no. 2, 2021, doi: 10.18196/jrc.2254.

[4] “Making a Mobile Educational Robot with a Practical Approach Using Arduino,” in *Nusantara Science and Technology Proceedings*, Galaxy Science, May 2022. doi: 10.11594/nstp.2022.2439.

[5] I. A. Hassan, I. A. Abed, and W. A. Al-Hussaibi, “Path Planning and Trajectory Tracking Control for Two-Wheel Mobile Robot,” *J. Robot. Control JRC*.

[6] F. Fahmizal, A. Priyatmoko, and A. Mayub, “Implementasi Kinematika Trajectory Lingkaran pada Robot Roda Mecanum,” *J. List. Instrumentasi Dan Elektron. Terap. JuLIET*, vol. 3, no. 1, May 2022, doi: 10.22146/juliet.v3i1.74760.

[7] D. U. Rijalusalam and I. Iswanto, “Implementation Kinematics Modeling and Odometry of Four Omni Wheel Mobile Robot on The Trajectory Planning and Motion Control Based Microcontroller,” *J. Robot. Control JRC*, vol. 2, no. 5, 2021, doi: 10.18196/jrc.25121.

[8] A. Suarez, D. García-Costa, J. Perez, E. López-Iñesta, F. Grimaldo, and J. Torres, “Hands-on Learning: Assessing the Impact of a Mobile Robot Platform in Engineering Learning Environments,” *Sustainability*, vol. 15, no. 18, p. 13717, Sept. 2023, doi: 10.3390/su151813717.

[9] X. Xu, P. Guo, J. Zhai, and X. Zeng, “Robotic kinematics teaching system with virtual reality, remote control and an on–site laboratory,” *Int. J. Mech. Eng. Educ.*, vol. 48, no. 3, pp. 197–220, July 2020, doi: 10.1177/0306419018807376.

[10] D. Wang, C. Kohler, X. Zhu, M. Jia, and R. Platt, “BulletArm: An Open-Source Robotic Manipulation Benchmark and Learning Framework,” Oct. 17, 2022, *arXiv*: arXiv:2205.14292. doi: 10.48550/arXiv.2205.14292.

[11] Y. Zhang, B. Feng, G. Zhang, and X. Bai, “The Application of CoppeliaSim EDU in Robot Course Teaching,” *J. Educ. Educ. Res.*, vol. 7, no. 1, pp. 167–170, Jan. 2024, doi: 10.54097/as82qn67.

[12] A. A. Oliva, F. Spindler, P. R. Giordano, and F. Chaumette, “FrankaSim: A Dynamic Simulator for the Franka Emika Robot with Visual-Servoing Enabled Capabilities,” in *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2022, pp. 868–874. doi: 10.1109/ICARCV57592.2022.10004274.

[13] N. Zivkovic, A. Devic, J. Vidakovic, I. Lazarevic, and M. Lazarević, “Design of a 6DOF Robot Simulation System in ROS-Gazebo with a Brief Reference to Modern Robot Simulation Software,” in *New Trends in Engineering Research*, N. Mitrovic, G. Mladenovic, and A. Mitrovic, Eds., Cham: Springer Nature Switzerland, 2024, pp. 246–252.

[14] Y. Niu, H. Qazi, and Y. Liang, “Building a Flexible Mobile Robotics Teaching Toolkit by Extending MATLAB/Simulink with ROS and Gazebo,” in *2021 7th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, 2021, pp. 10–16. doi: 10.1109/ICMRE51691.2021.9384836.

[15] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox, “Mona: an Affordable Open-Source Mobile Robot for Education and Research,” *J. Intell. Robot. Syst.*, vol. 94, no. 3–4, pp. 761–775, June 2019, doi: 10.1007/s10846-018-0866-9.