Development of a Web-Based Interface for Control Motor on Mobile Robot

1st Muhammad Afif Athallah
Program Studi Teknologi Rekayasa Robotika, Jurusan Teknik Elektro
Politeknik Negeri Batam
Batam, Indonesia
m.athallah180900@gmail.com

2nd Senanjung Prayoga, S.Pd., M.T.
Program Studi Teknologi Rekayasa Robotika, Jurusan Teknik Elektro
Politeknik Negeri Batam
Batam, Indonesia
senanjung@polibatam.ac.id

3rd Rifqi Amalya Fatekha, S.ST., M.Tr.T.
Program Studi Teknologi Rekayasa Robotika, Jurusan Teknik Elektro
Politeknik Negeri Batam
Batam, Indonesia
rifqi@polibatam.ac.id

4th Ryan Satria Wijaya, S.Tr.T., M.Tr.T.
Program Studi Teknologi Rekayasa Robotika, Jurusan Teknik Elektro
Politeknik Negeri Batam
Batam, Indonesia
ryan@polibatam.ac.id

***Abstract*—Mobile robots require a control system capable of maintaining stable movement and orientation during operation. One of the most widely used control methods is the Proportional-Integral-Derivative (PID) controller. However, manually tuning PID parameters requires considerable time and user experience, making the process less efficient. This study aims to develop a web-based interface that enables real-time system monitoring and PID parameter tuning for a mobile robot. The developed system integrates a web-based interface served by a Python Flask-SocketIO backend, which bridges communication with an Arduino Mega over a serial connection. The web interface allows users to adjust PID parameters, monitor the robot's condition, and control its movement directly through a web browser. Experimental results demonstrate that the PID controller is capable of regulating the motor speed to closely match the desired setpoint. The implementation of the web-based interface also simplifies the tuning process, as the proportional (Kp), integral (Ki), and derivative (Kd) parameters can be adjusted in real time while the system response is observed immediately. The findings indicate that integrating a web-based interface with the PID control system enhances the flexibility of testing and tuning processes for an omnidirectional mobile robot. Furthermore, the developed web-based interface proved to be effective for real-time motor control and mobile robot monitoring.**

***Keywords*—**Mobile Robot, PID Control, Web Interface, Motor Control, Real-Time Monitoring

I. INTRODUCTION

Technological advancements have significantly increased the use of mobile robots across industries, households, and healthcare in recent years [1]. Once operated manually, mobile robots are now increasingly autonomous, leveraging artificial intelligence, computer vision, and various sensors for navigation and decision-making [2]. Beyond mechanical design, a robot's performance is now largely determined by its ability to navigate autonomously, plan efficient paths, avoid obstacles, and operate in dynamic environments without human intervention [3].

One robot type capable of such navigation is the omnidirectional mobile robot, which can move translationally and rotationally without changing its body orientation [4]. This capability relies on coordinated control of each wheel's speed, making motor synchronization critical for stable, smooth movement [5].

Accurate robot motion therefore requires each motor's speed to be controlled using a Proportional-Integral-Derivative (PID) controller; if motor speed cannot be properly controlled, the wheels may fail to operate or maneuver optimally [6]. PID remains one of the most widely used control methods in robotics due to its simple structure, ease of implementation, and stable response when properly tuned [7].

A key challenge in controlling an omnidirectional robot is that PID tuning is often impractical, since parameters must be modified directly in the microcontroller program, and differing motor characteristics require each motor's parameters to be individually adjusted for optimal performance [1]. A web-based interface that integrates PID adjustment, robot control, and real-time monitoring is therefore needed to simplify the tuning and testing process [8].

Manual PID tuning remains common in practice, but relies on operator experience and a time-consuming, inefficient trial-and-error approach, so control performance often depends heavily on user expertise [9]. Numerous studies have applied PID control in robotic systems: Wahab et al. [6] developed a PID-based control system that improved an omnidirectional wheeled robot's ability to regulate motor speed and maintain orientation; Zaman and Wu [10] combined PID control with a Radial Basis Function (RBF) neural network for an omnidirectional platform; and Bo et al. [11] proposed an Improved Genetic Algorithm for PID attitude-control tuning, which accelerated convergence and produced more stable control than several existing approaches.

Other studies have focused specifically on improving the PID tuning process itself. Ayurzana et al. [12] developed a graphical-user-interface-based PID tuning system for DC motor

speed control in mobile robots, simplifying parameter adjustment while supporting real-time response monitoring, while other work has proposed optimization techniques that automatically determine more optimal control parameters [13].

While prior work has developed PID-based omnidirectional robot controllers and separate web-based monitoring platforms, an integrated system combining real-time PID tuning, motor control, and web-based monitoring in one platform remains limited. This study therefore proposes a web-based interface that lets users adjust the proportional (Kp), integral (Ki), and derivative (Kd) parameters directly through a browser while issuing motion commands and monitoring motor response in real time, improving the flexibility and efficiency of the PID tuning process.

II. METHOD

A. Web-Based Interface

A web-based interface was developed as a Human Machine Interface (HMI) to facilitate the monitoring and control of the mobile robot. The website serves as an interactive platform that enables users to monitor the system performance and configure the PID controller without directly interacting with the hardware or modifying the microcontroller program. This approach simplifies the testing process by providing all system information and control functions through a single user interface.

The primary function of the website is to provide real-time monitoring of the robot during operation. The interface displays essential system parameters, including the setpoint, actual motor speed (RPM), error value, error percentage, PWM output, and the Proportional (Kp), Integral (Ki), and Derivative (Kd) gains currently applied to the PID controller. In addition, the website presents the individual contributions of the PID controller, namely the P Output, I Output, D Output, and the resulting PID Output. These parameters allow users to observe the behavior of the control system and evaluate the effect of PID tuning on the motor response in real time.

In addition to monitoring, the website supports PID parameter tuning: users can adjust Kp, Ki, and Kd directly through the interface, allowing different parameter combinations to be tested quickly without reprogramming the microcontroller for every change. Communication between the interface and the Arduino is handled by a Python-based bridge application (Flask-SocketIO) over a serial connection, with the dashboard charts refreshing at the same 100 ms rate as the Arduino's control loop. The round-trip communication latency between browser and Arduino was measured empirically to confirm the system's suitability for real-time tuning; the measurement procedure and results are presented in the PID Tuning subsection.

B. System Architecture

The flowchart illustrates the workflow of the mobile robot control system integrated with a web-based interface for real-time PID tuning [8]. The process begins when the user enters or modifies the PID parameters (Kp, Ki, and Kd) through the web interface. These parameters are transmitted to the backend and subsequently forwarded to the Arduino via a Python Bridge using serial communication. The Arduino processes the received parameters and reads the motor speed feedback from the encoder. Based on the PID control algorithm, PWM signals are generated and sent to the motor driver, enabling the robot to move according to the specified command. The flowchart of the proposed system is presented in Fig. 1.

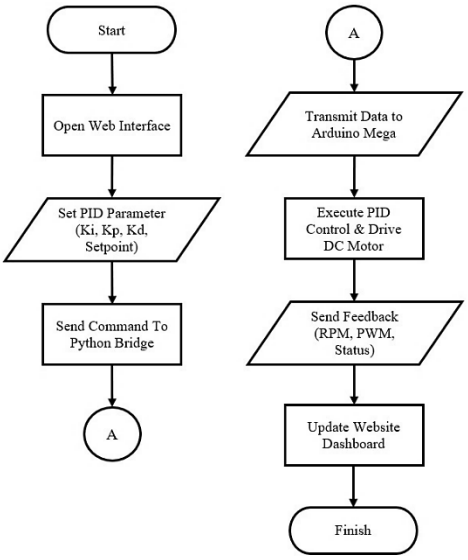

Fig. 1. Flowchart of the proposed web-based PID control system.

C. Wiring Diagram

The proposed mobile robot consists of an Arduino Mega, JGA25-370 DC motors, an L298N motor driver, and a 12 VDC power supply, as shown in Fig. 2. The Arduino Mega serves as the main controller, executing the PID control algorithm and receiving commands from the web interface. It generates PWM signals to the L298N motor driver, which controls the speed and direction of the DC motors [14]. The 12 VDC power supply provides the required electrical power for the entire system. The wiring diagram illustrates the electrical connections among the hardware components that form the proposed mobile robot control system [15].

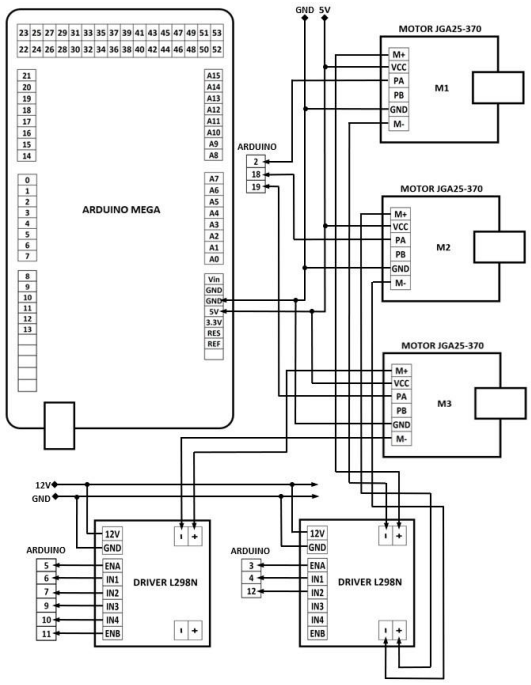

Fig. 2. Wiring diagram of the mobile robot control system.

D. Design 3D

Fig. 3 illustrates the three-dimensional (3D) design of the proposed mobile robot from the side and top views. The robot adopts a two-layer mechanical structure, where the lower layer accommodates the three DC motors and omnidirectional wheels, while the upper layer houses the electronic components, including the microcontroller, motor driver, power supply, and other supporting devices [16]. The three omnidirectional wheels are arranged in a triangular configuration at angles of 60°, 180°, and 300° with respect to the robot's center of gravity, enabling omnidirectional motion through both translational and rotational movements without changing the robot's body orientation. In addition, the component placement is designed to achieve balanced weight distribution, thereby improving the robot's stability during operation [17].

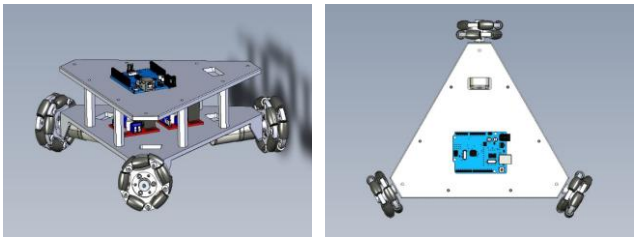

Fig. 3. 3D design of the proposed mobile robot: side view (left) and top view (right).

E. PID Control Algorithm

The Proportional-Integral-Derivative (PID) controller is a feedback control method that generates a control signal based on the difference between the desired setpoint and the measured value [18]. The continuous-time PID controller is expressed as

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

Equation (1) is to describes the PID control law, where u(t) is the control output generated by the PID controller; e(t) is the error, defined as the difference between the desired setpoint a nd the measured motor speed; Kp is the proportional gain, which determines the controller response based on the current error; Ki is the integral gain, which accumulates past errors to eliminate steady-state error; Kd is the derivative gain, which predicts future error based on the rate of change of the error and improves system stability; and t represents time, while τ is the integration variable.

Since the Arduino executes the control algorithm periodically rather than continuously, the PID controller is implemented in discrete form during each control loop as follows:

$$e(t) = SP - PV(t) \tag{2}$$

Equation (2) is to calculates the instantaneous error e(t), where SP is the desired motor speed (setpoint) and PV(t) is the process variable — the actual motor speed measured by the encoder at time t.

$$I(t) = I(t-1) + e(t)\,\Delta t \tag{3}$$

Equation (3) is to updates the accumulated integral term I(t), where I(t−1) is the integral value from the previous control cycle and Δt is the sampling interval.

$$D(t) = \frac{e(t) - e(t-1)}{\Delta t} \tag{4}$$

Equation (4) is to computes the derivative term D(t), the rate of change of the error, where e(t−1) is the error calculated in the previous control cycle.

$$u(t) = K_p\, e(t) + K_i\, I(t) + K_d\, D(t) \tag{5}$$

Equation (5) is the combine of three terms to compute the PID controller output u(t), which is then converted into a PWM signal to regulate the speed of the DC motors.

F. PID Tuning

Updated PID parameters take effect immediately, letting the robot's response be monitored in real time through the web interface [19] so users can evaluate rise time, overshoot, and settling time and iteratively adjust the gains until the desired response is achieved — simplifying optimization without interrupting robot operation [20].

PID gains for each motor were obtained through manual trial-and-error tuning via the web interface, adjusting the Kp, Ki, and Kd sliders while observing the real-time response until acceptable rise time, overshoot, and settling time were achieved, rather than using an automated method such as Ziegler–Nichols. Since Motor 1's resulting integral gain (Ki = 59.300) is relatively

large, the accumulated Integral term is clamped to a fixed range to prevent integral windup.

The percentage overshoot reported in this study is calculated as Overshoot (%) = ((Peak RPM − Setpoint) / Setpoint) × 100%, where Peak RPM is the maximum motor speed recorded during the transient response.

The Arduino's PID loop executes at a fixed sampling interval of dt = 100 ms. Serial communication with the Python Bridge uses 115200 bps, giving an effective transfer rate of 11,520 bytes/s; at this rate, a parameter-update command (20–30 bytes) takes roughly 2–3 ms and a full three-motor telemetry frame (~178 bytes) takes roughly 15 ms, both small relative to the 100 ms loop interval. To measure the actual round-trip latency, an immediate-echo command was added to the serial protocol so the Arduino replies without waiting for the next control-loop tick, isolating the delay from browser to Arduino and back. Measured client-side with performance.now() over N = 100 trials, the round-trip time was 14.85 ± 7.30 ms (median 13.20 ms, 95th percentile 31.30 ms, range 3.60–41.30 ms) — closely matching the analytical estimate and confirming that communication overhead is small relative to the 100 ms control-loop period. The time between a web-interface parameter change and its effect appearing in telemetry is therefore dominated by the 100 ms loop interval itself, supporting the system's suitability for real-time monitoring and tuning.

III. RESULT AND DISCUSSION

Fig. 4 shows the web-based PID interface used as the Human–Machine Interface (HMI) for real-time monitoring and control of the three DC motors, with independent panels for Motor 1, Motor 2, and Motor 3 and a connection-status indicator showing communication between the dashboard, Python Bridge, and Arduino Mega.

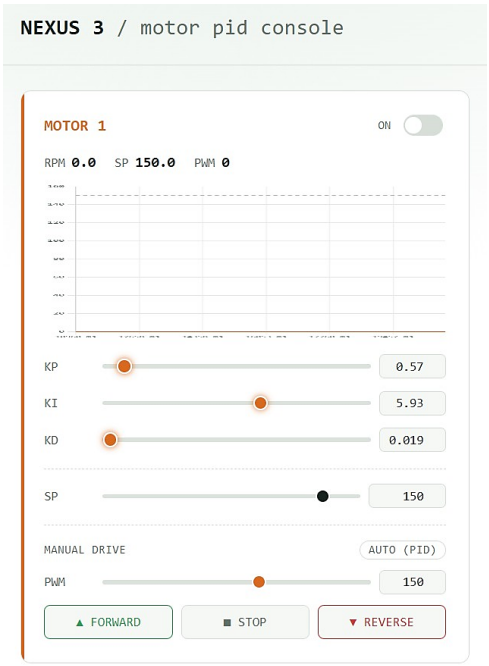

Fig. 4. Web-based PID monitoring and control interface for the three DC motors.

Each panel displays the current RPM, Setpoint (SP), and PWM, plus a real-time speed-response graph, and lets users adjust Kp, Ki, and Kd via sliders or numeric fields, modify the setpoint, and — in Manual Drive mode — set PWM directly. Forward, Stop, and Reverse buttons control motor direction, and an ON/OFF switch enables or disables each motor. Parameter updates are sent to the Python Bridge and forwarded to the Arduino Mega, which returns encoder feedback (RPM, PWM, and status) for real-time monitoring.

A. Implementation PID in Motor 1

Fig. 5. PID parameter configuration interface for Motor 1.

Fig. 5 shows the PID parameter configuration interface for Motor 1, through which Kp, Ki, and Kd can be adjusted directly in real time.

Based on the data in Table I, Motor 1 was tested using Kp = 0.57, Ki = 5.930, and Kd = 0.019 with a setpoint of 150 RPM and start running at 912 ms. The motor speed increased from 0 RPM to 147.80 RPM at 1413 ms and subsequently stabilized around the setpoint, fluctuating between 147.80 RPM and 151.32 RPM. During the steady-state condition, the error remained within −1.32 RPM to 2.20 RPM (−0.88% to 1.47%), indicating that the PID controller achieved stable speed regulation with only minor oscillations.

TABLE I
RESULTS OBTAINED BASED ON PID

Time (ms)	Setpoint	RPM Actual	Error	Error (%)	PWM
912	150	0	150	100	202
1010	150	75,66	74,34	49,56	161
1110	150	128,45	21,55	14,37	148
1209	150	144,28	5,72	3,81	149
1308	150	146,04	3,96	2,64	153
1413	150	147,8	2,2	1,47	153
1508	150	151,32	-1,32	-0,88	150
1612	150	151,32	-1,32	-0,88	150
1709	150	149,56	0,44	0,29	152
1803	150	149,56	0,44	0,29	152
1908	150	151,32	-1,32	-0,88	149
2003	150	149,56	0,44	0,29	151
2110	150	149,56	0,44	0,29	151
2211	150	151,32	-1,32	-0,88	149
2307	150	149,56	0,44	0,29	151
2414	150	149,56	0,44	0,29	151
2508	150	151,32	-1,32	-0,88	149
2604	150	149,56	0,44	0,29	151
2711	150	149,56	0,44	0,29	151
2808	150	149,56	0,44	0,29	151

Fig. 6 presents the corresponding motor speed response over time, showing the speed gradually reaching and remaining stable near the 150 RPM setpoint.

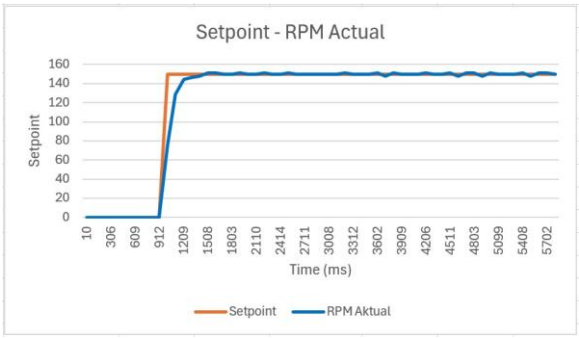

Fig. 6. Motor speed response over time for Motor 1 (data from Table I).

B. Implementation PID in Motor 2

Fig. 7. PID parameter configuration interface for Motor 2.

Fig. 7 shows the PID parameter configuration interface for Motor 2, through which Kp, Ki, and Kd can be adjusted directly in real time.

Based on the data in Table II, Motor 2 was tested using Kp = 0.33, Ki = 2.262, and Kd = 0.015 with a setpoint of 150 RPM and start running at 1709 ms. The motor speed gradually increased from 0 RPM and reached 149.56 RPM at 2414 ms before stabilizing around the setpoint. During the steady-state condition, the motor speed fluctuated between 149.56 RPM and 151.32 RPM, with an error ranging from −1.32 RPM to 0.44 RPM (−0.88% to 0.29%), indicating that the PID controller achieved stable speed regulation with only minor oscillations.

TABLE II
RESULTS OBTAINED BASED ON PID

Time (ms)	Setpoint	RPM Actual	Error	Error (%)	PWM
1612	150	0	150	100	0
1709	150	0	150	100	105
1803	150	52,79	97,21	64,81	80
1908	150	103,81	46,19	30,79	73
2003	150	117,89	32,11	21,41	82
2110	150	128,45	21,55	14,37	84
2211	150	139	11	7,33	83
2307	150	144,28	5,72	3,81	83
2414	150	149,56	0,44	0,29	81
2508	150	151,32	-1,32	-0,88	81
2604	150	149,56	0,44	0,29	82
2711	150	151,32	-1,32	-0,88	81
2808	150	151,32	-1,32	-0,88	81
2904	150	151,32	-1,32	-0,88	80
3008	150	151,32	-1,32	-0,88	80
3103	150	149,56	0,44	0,29	81
3206	150	149,56	0,44	0,29	81
3312	150	151,32	-1,32	-0,88	80
3404	150	149,56	0,44	0,29	81
3504	150	151,32	-1,32	-0,88	79

Fig. 8 presents the corresponding motor speed response over time, showing the speed gradually reaching and remaining stable near the 150 RPM setpoint.

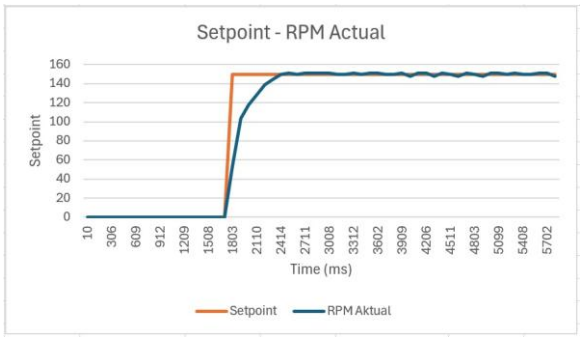

Fig. 8. Motor speed response over time for Motor 2 (data from Table II).

C. Implementation PID in Motor 3

Fig. 9. PID parameter configuration interface for Motor 3.

Fig. 9 shows the PID parameter configuration interface for Motor 3, through which Kp, Ki, and Kd can be adjusted directly in real time.

Based on the data in Table III, Motor 3 was tested using Kp = 0.36, Ki = 2.800, and Kd = 0.015 with a setpoint of 150 RPM and start running at 2711 ms. The motor speed gradually increased from 0 RPM and reached 147.80 RPM at 3206 ms, followed by a maximum overshoot of 154.84 RPM (3.23%). After the transient response, the motor speed stabilized around the setpoint, fluctuating between 147.80 RPM and 151.32 RPM. During the steady-state condition, the error remained within −1.32 RPM to 2.20 RPM (−0.88% to 1.47%), indicating that the PID controller effectively regulated the motor speed despite the initial overshoot. This range coincides numerically with the steady-state error range obtained for Motor 1, which is expected because Motor 1 and Motor 3 use the same JGA25-370 gear-motor and encoder combination sampled at the same 100 ms loop interval, so both motors' speed readings are quantized to the same discrete RPM steps regardless of their individually tuned PID gains.

TABLE III
RESULTS OBTAINED BASED ON PID

Time (ms)	Setpoint	RPM Actual	Error	Error (%)	PWM
2508	150	0	150	100	0
2604	150	0	150	100	0
2711	150	0	150	100	118
2808	150	59,82	90,18	60,12	90
2904	150	116,13	33,87	22,58	80
3008	150	133,72	16,28	10,85	84
3103	150	140,76	9,24	6,16	86
3206	150	147,8	2,2	1,47	84
3312	150	153,08	-3,08	-2,05	81
3404	150	151,32	-1,32	-0,88	83
3504	150	154,84	-4,84	-3,23	79
3602	150	154,84	-4,84	-3,23	78
3698	150	151,32	-1,32	-0,88	80
3808	150	151,32	-1,32	-0,88	79
3909	150	151,32	-1,32	-0,88	78
4005	150	149,56	0,44	0,29	79
4100	150	149,56	0,44	0,29	79
4206	150	151,32	-1,32	-0,88	78
4300	150	147,8	2,2	1,47	81
4408	150	149,56	0,44	0,29	79

Fig. 10 presents the corresponding motor speed response over time, showing the speed gradually reaching and remaining stable near the 150 RPM setpoint.

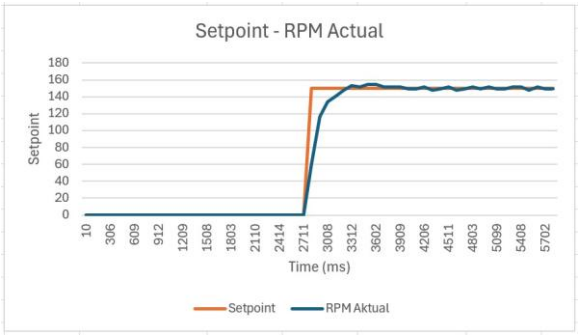

Fig. 10. Motor speed response over time for Motor 3 (data from Table III).

D. Comparative Analysis

The experiments were conducted on three DC motors at a 150 RPM setpoint; all three achieved speeds close to the setpoint after the transient response, with small steady-state deviations. Motor 1 (Kp = 0.57, Ki = 5.930, Kd = 0.019) is shown in Fig. 11, where the RPM curve rises rapidly and settles around 150 RPM with only minor oscillations — a fast transient response with good steady-state accuracy.

Fig. 11. Real-time response of Motor 1 on the web-based monitoring interface.

Motor 2 (Kp = 0.33, Ki = 2.262, Kd = 0.015), shown in Fig. 12, also converges to 150 RPM but with a smoother rise before settling, reflecting dynamic characteristics that differ from Motor 1.

Fig. 12. Real-time response of Motor 2 on the web-based monitoring interface.

Motor 3 (Kp = 0.36, Ki = 2.800, Kd = 0.015), shown in Fig. 13, exhibits a response pattern similar to Motor 2, reaching 150 RPM with small steady-state deviations.

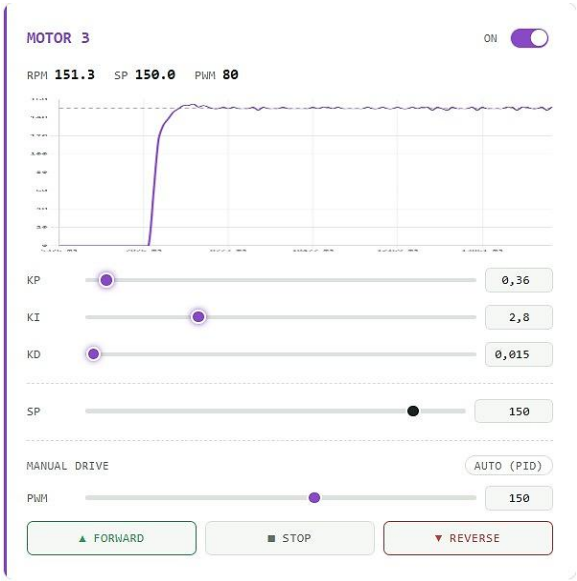

Fig. 13. Real-time response of Motor 3 on the web-based monitoring interface.

As shown in Figs. 11–13, the variation in PID gains among the three motors reflects their differing dynamic characteristics, requiring individual tuning for satisfactory control — a process the proposed web-based interface enables efficiently by allowing each motor's parameters to be adjusted independently in real time.

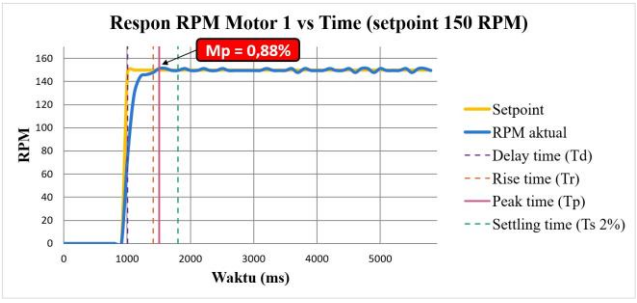

Fig. 14. Transient response test results for Motor 1 (setpoint 150 RPM).

As shown in Fig. 14, Motor 1's transient response test (start at 912 ms, 150 RPM setpoint) reached a peak speed of 151.32 RPM at 1,508 ms — a maximum overshoot of 0.88%. The system reached 50% of the setpoint at 1,010 ms and 90% at 1,413 ms (rise time 501 ms), then settled within ±2% by 891 ms (settling time). These results indicate a fast, low-overshoot, and stable speed response.

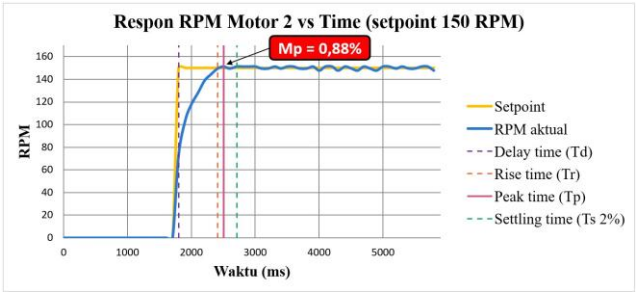

Fig. 15. Transient response test results for Motor 2 (setpoint 150 RPM).

As shown in Fig. 15, Motor 2's transient response test (start at 1709 ms, 150 RPM setpoint) reached a peak speed of 151.32 RPM at 2,508 ms — a maximum overshoot of 0.88%. The system reached 50% of the setpoint at 1,803 ms and 90% at 2,414 ms (rise time 705 ms), then settled within ±2% at a settling time of 908 ms since the motor was started. These results demonstrate a relatively fast response with minimal overshoot and stable performance.

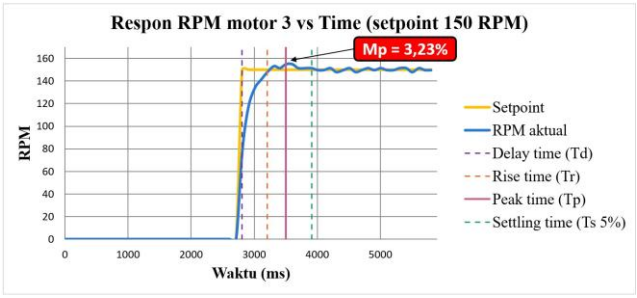

Fig. 16. Transient response test results for Motor 3 (setpoint 150 RPM).

As shown in Fig. 16, Motor 3's transient response test (start at 2711 ms, 150 RPM setpoint) reached a peak speed of 154.84 RPM at 3,504 ms — a maximum overshoot of 3.23%. The system reached 50% of the setpoint at 2,808 ms and 90% at 3,206 ms (rise time 497 ms), then settled within ±5% by a settling time of 1,198 ms. Despite the larger overshoot, the system returned to a stable state and maintained the desired speed, indicating effective control performance.

IV. CONCLUSION

Based on the transient response tests for the three motors at a 150 RPM setpoint, the implemented PID control system regulated motor speed effectively, with all overshoot values remaining within an acceptable ±5% range (Motors 1 and 2: 0.88%; Motor 3: 3.23%). Motor 3 achieved the fastest rise time (497 ms), followed by Motor 1 (501 ms) and Motor 2 (705 ms); for settling time, Motor 1 was fastest (891 ms), followed by Motor 2 (908 ms) and Motor 3 (1,198 ms), the latter reflecting its higher overshoot. Overall, the system demonstrated fast, stable, and accurate speed regulation for all three motors.

The proposed web-based PID control system for a mobile robot was thus successfully developed and validated. The interface facilitated real-time PID tuning and performance monitoring, with a measured round-trip communication latency of 14.85 ± 7.30 ms (N = 100 trials) — small relative to the 100 ms control-loop period — confirming that it supports real-time monitoring and control without significant delay, and providing a flexible, efficient platform for PID tuning and real-time mobile-robot control.

This study has limitations worth noting: each motor was evaluated in a single test run rather than repeated trials, so repeatability and statistical measures (e.g., standard deviation, IAE, ISE, ITAE) could not be reported, and the interface's benefits were assessed qualitatively rather than through direct comparison with manual reprogramming or benchmarking against related studies. Future work will address these limitations through repeated trials with statistical analysis,

quantitative timing comparisons, literature benchmarking, and PID-based heading/orientation control using the onboard IMU sensor.

REFERENCES

[1] M. Huba, P. Mižák, and P. Bisták, "PID Tuning for DIPDT System by Web Application," *IFAC-PapersOnLine*, vol. 55, no. 4, pp. 201–206, 2022, doi: 10.1016/j.ifacol.2022.06.033.

[2] S. Cebollada, L. Payá, M. Flores, A. Peidró, and O. Reinoso, *A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data*, vol. 167. Elsevier Ltd., 2021. doi: 10.1016/j.eswa.2020.114195.

[3] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path Planning for Autonomous Mobile Robots: A Review," *Sensors*, vol. 21, no. 23, 2021, doi: 10.3390/s21237898.

[4] G. Erdemir, A. E. Kuzucuoglu, E. Kaplanoglu, and Y. El-Kahlout, "Design and Implementation of Web Based Mobile Robot Control Platform for Robotics Education," *Appl. Mech. Mater.*, vol. 704, pp. 283–287, 2015, doi: 10.4028/www.scientific.net/AMM.704.283.

[5] H. Taheri and C. X. Zhao, "Omnidirectional mobile robots, mechanisms and navigation approaches," *Mech. Mach. Theory*, vol. 153, p. 103958, 2020, doi: https://doi.org/10.1016/j.mechmachtheory.2020.103958.

[6] F. Wahab, C. Zaskie, and B. M. Arthaya, "Kendali Kecepatan Robot Beroda Omni dengan Kemampuan Menuju Posisi dan Orientasi yang Diinginkan Berbasis Pengendali PID," *JTERA (Jurnal Teknol. Rekayasa)*, vol. 6, no. 2, p. 273, 2021, doi: 10.31544/jtera.v6.i2.2021.273-284.

[7] Q. L. Dinh, T. M. Nguyen, and T. D. Nguyen, "Metaheuristic Algorithms to Determine PID Controller Parameters for Mobile Robots," *J. Electr. Electron. Eng.*, vol. 18, no. 1, pp. 27–32, 2025.

[8] P. Fernandez, F. Hadary, and S. D. Panjaitan, "Interactive Learning for Website-Based PID Controller," *J. Otomasi Kontrol dan Instrumentasi*, vol. 17, no. 2, pp. 113–121, 2025, doi: 10.5614/joki.2025.17.2.6.

[9] O. A. Somefun, K. Akingbade, and F. Dahunsi, "The dilemma of PID tuning," *Annu. Rev. Control*, vol. 52, pp. 65–74, 2021, doi: https://doi.org/10.1016/j.arcontrol.2021.05.002.

[10] M. Q. Zaman and H.-M. Wu, "Genetic Algorithm Based Takagi-Sugeno Fuzzy Modelling of a Mecanum Wheeled Mobile Robot," *Int. J. Control. Autom. Syst.*, vol. 23, no. 3, pp. 907–919, 2025, doi: 10.1007/s12555-024-0506-z.

[11] G. A. O. Bo, W. U. Minghui, H. U. Heping, and Y. Chen, "Attitude PID control parameter tuning of curtain wall cleaning robot based on improved genetic algorithm," *J. Shanghai Univ. Eng. Sci.*, vol. 38, no. 4, pp. 429–436, 2024.

[12] M. E. Ayurzana, E. Gankhuyag, D. Batbaatar, and N. Erdenesuren, "PID Parameter Tuning of a Low-Cost DC Motor Speed Control for Mobile Robot Application," *Proc. Int. Conf. Artif. Life Robot.*, pp. 1004–1008, 2024, doi: 10.5954/icarob.2024.gs4-5.

[13] F. Hasan, B. B. Murti, Fahmizal, and M. Arrofiq, "Kendali Heading pada Trajectory Tracking Miniatur Robot Mobil," *J. List. Instrumentasi dan Elektron. Terap.*, vol. 1, no. 2, pp. 3–7, 2021, doi: 10.22146/juliet.v1i2.59347.

[14] T. Taufiq and A. Aswardi, "Self Balancing Robot Menggunakan Metode PID Berbasis Arduino," *JTEIN J. Tek. Elektro Indones.*, vol. 3, no. 1 SE-Articles, pp. 15–24, Jan. 2022, doi: 10.24036/jtein.v3i1.192.

[15] R. Euldji *et al.*, "Improved Path Tracking Control in Mobile Robots Using a Hybrid FOPID Controller with Backstepping Technique: An Experimental Study," *J. Eur. des Syst. Autom.*, vol. 56, no. 2, pp. 173–186, 2023, doi: 10.18280/jesa.560201.

[16] S. Supriadi, A. Wajiansyah, M. Zainuddin, and A. B. W. Putra, "Optimization of Proportional Integral Derivative Controller for Omni Robot Wheel Drive by Using Integrator Wind-up Reduction Based on Arduino Nano," *J. Robot. Control*, vol. 5, no. 6 SE-Articles, pp. 1690–1701, Sep. 2024, doi: 10.18196/jrc.v5i6.21807.

[17] R. T. Yunardi, D. Arifianto, F. Bachtiar, and J. Intan Prananingrum, "Holonomic Implementation of Three Wheels Omnidirectional Mobile Robot using DC Motors," *J. Robot. Control*, vol. 2, no. 2 SE-Articles, pp. 65–71, Mar. 2021, doi: 10.18196/jrc.2254.

[18] R. Solekha and U. Latifa, "Sistem Kendali Proportional Integral Derivative (PID) Menggunakan Mikrokontroler Arduino Pada Thinkercad," *Electron J. Ilm. Tek. Elektro*, vol. 5, no. 1, pp. 89–96, 2024.

[19] K. Hamıdı and S. Abdulkerim, "Design and Validation of an Online PID Tuning Tool for Mechatronic Systems : Application to a Dual-BLDC Balancing Platform," vol. 10, no. 3, pp. 748–757, 2026.

[20] Y. Wu, S. Li, and K. Li, "Enhanced receding horizon optimal performance for online tuning of PID controller parameters," *Int. J. Model.*, vol. 10, no. 10, pp. 1–9, 2018.

[21] L. Abdikarso, P. Ramadhan, and D. Irawan, "Design and Implementation of a Web-Based SCADA for Closed-Loop PID Control Optimized by AI with PLC Integration: A Multi-Layer Architecture Integrating Real-Time Monitoring, Intelligent PID Tuning, and Industrial Automation," *J. Edukasi Elektro*, vol. 9, no. 2, pp. 159–169, 2025, [Online]. Available: https://journal.uny.ac.id/index.php/jee/article/view/88868

[22] M. B. Emara, A. W. Youssef, M. Mashaly, J. Kiefer, L. A. Shihata, and E. Azab, "Digital Twinning for Closed-Loop Control of a Three-Wheeled Omnidirectional Mobile Robot," *Procedia CIRP*, vol. 107, no. March, pp. 1245–1250, 2022, doi: 10.1016/j.procir.2022.05.139.

[23] S. Sagiroglu and N. Yilmaz, "Web-based mobile robot platform for real-time exercises," *Expert Syst. Appl.*, vol. 36, no. 2, Part 2, pp. 3153–3166, 2009, doi: https://doi.org/10.1016/j.eswa.2008.01.046.

[24] H. Perier, E. Matheson, and M. Di Castro, "Web based User Interface Solution for Remote Robotic Control and Monitoring Autonomous System," *Proc. Int. Conf. Informatics Control. Autom. Robot.*, vol. 1, no. Icinco, pp. 680–687, 2022, doi: 10.5220/0011318800003271.

[25] S. Kirci, U. Birgul, and G. Atali, "A Multi-Functional Web Control Interface for Industrial Autonomous Mobile Robot Fleets," *Eur. J. Res. Dev.*, vol. 4, no. 4, pp. 101–109, 2024, doi: 10.56038/ejrnd.v4i4.517.

[26] U. U. S. Rajapaksha, C. Jayawardena, and B. A. Macdonald, "Design, Implementation, and Performance Evaluation of a Web-Based Multiple Robot Control System," *J. Robot.*, vol. 2022, 2022, doi: 10.1155/2022/9289625.