A Project-Based Design of a Two-Wheeled Self-Balancing Robot Using PID Controller

Amat Amir Basari , Azuha Syuhada Norman Zairi

Faculty of Electronics and Computer Technology and Engineering, Universiti Teknikal Malaysia Melaka (UTeM), 76100 Durian Tunggal, Melaka, Malaysia

DOI: https://doi.org/10.47772/IJRISS.2026.100500314

Received: 07 May 2026; Accepted: 12 May 2026; Published: 30 May 2026

ABSTRACT

This project focuses on the design and development of a two-wheel self-balancing robot prototype controlled by a PID algorithm. The hardware setup includes an ESP32 microcontroller that processes orientation data from an MPU6050 sensor to manage the movement of NEMA17 stepper motors. A key feature of this system is its connectivity; sensor data is sent to the Blynk IoT platform, which allows for remote PID tuning and system adjustments without needing to modify the code manually. At the same time, the data is exported to MATLAB for a more technical and detailed analysis of the control response. By using a closed-loop control system, the robot can maintain its balance and follow a specific trajectory more effectively, even when faced with external disturbances or changes in the PID parameters. After testing the robot's performance under various settings, the results show a successful implementation of stable control. For future development, adding a camera system could expand the robot's capabilities to include surveillance and monitoring tasks.

INTRODUCTION

The development of two-wheeled self-balancing robots (TWSBR) has emerged as a cornerstone in modern robotics research, serving as a classic example of an underactuated, inherently non-linear, and unstable system (Zhang & Nor, 2025). These robots are modeled after an inverted pendulum, where the center of mass is located above the axis of rotation, requiring continuous motor adjustments to maintain an upright position (John, 2025). While their structural simplicity allows for zero-radius turning and high mobility in tight spaces, their control remains a significant challenge due to the complex coupling between tilt angle and forward velocity (Zhang & Nor, 2025; John, 2025).

Traditionally, most research in this field has focused on systems with a single degree of freedom (1-DOF), primarily managing the pitch or "tilt" of the robot. However, 1-DOF robots often struggle in dynamic real-world environments, showing sluggish response to external disturbances and limited flexibility when navigating uneven surfaces (Nikita et al., 2021). Recent studies have highlighted that 1-DOF systems can exhibit significant oscillations in tilt response, which hinders their efficiency in motion planning and obstacle navigation (Nikita et al., 2021). To address these limitations, researchers are increasingly exploring the integration of additional degrees of freedom (2-DOF) to improve maneuverability and stability (Nikita et al., 2021).

A widely adopted strategy for stabilizing these systems is the Proportional-Integral-Derivative (PID) controller, valued for its straightforward implementation and effectiveness (Kishore, 2024). While advanced methods like Deep Reinforcement Learning (DRL) and Sliding Mode Control (SMC) are gaining traction, the PID controller remains a fundamental benchmark in both industrial and educational settings (Baltes et al., 2023; Zhang & Nor, 2025). In fact, recent educational research has demonstrated the effectiveness of using PID control as a "bridge" for students to understand more complex machine learning algorithms (Emrah et al., 2021). Modern microcontrollers, such as the ESP32, have further revolutionized this field by providing high computing power and integrated IoT connectivity, making it ideal for STEM education and real-time data processing (Hao & Sitjongsataporn, 2026; Darko, 2023).

Beyond basic stability, the focus of TWSBR research is shifting toward assistive technologies, such as providing transportation or monitoring for individuals with limited mobility (Hao & Sitjongsataporn, 2026). This shift requires robots to possess enhanced interaction capabilities, including voice control and intelligent obstacle avoidance (Hao & Sitjongsataporn, 2026). Furthermore, the quality of control is not solely dependent on algorithm tuning but also on the efficient management of hardware and software resources, such as the distribution of tasks across the dual cores of an ESP32 (Sandeep et al., 2025; Maciej & Izabela, 2025).

This project involves the design and development of a 2-DOF self-balancing robot prototype specifically designed for educational purposes. By utilizing an ESP32 microcontroller and an MPU6050 inertial measurement unit (IMU), the system achieves real-time balance through a tuned PID algorithm. A key innovation in this work is the integration of the Blynk IoT platform and MATLAB software. This allows for remote PID tuning and in-depth analytical visualization without the need for constant hardware tethering (Nikita et al., 2021). By moving beyond the 1-DOF constraint, this project aims to provide a robust framework for learning mechatronics, control theory, and the practical application of Internet of Things (IoT) in robotics.

METHODOLOGY

Fig. 1 shows the block diagram of the whole integration of the robot and its control system. This system integrates a feedback loop, essential for dynamically adjusting the robot's balance in response to environmental interactions. The core hardware components include an MPU6050 sensor and an ESP32 microcontroller, which work in conjunction to process tilt angle data. The sensor detects the robot's inclination, and this data is fed into the PID controller within the ESP32. Using proportional, integral, and derivative gains fine-tuned via a manual trial and error method, the controller calculates the necessary adjustments and sends precise commands to the stepper motor. The closed-loop nature of this system indicates that any disturbances affecting the robot's balance, like physical pushes or pulls, are continually monitored and counteracted.

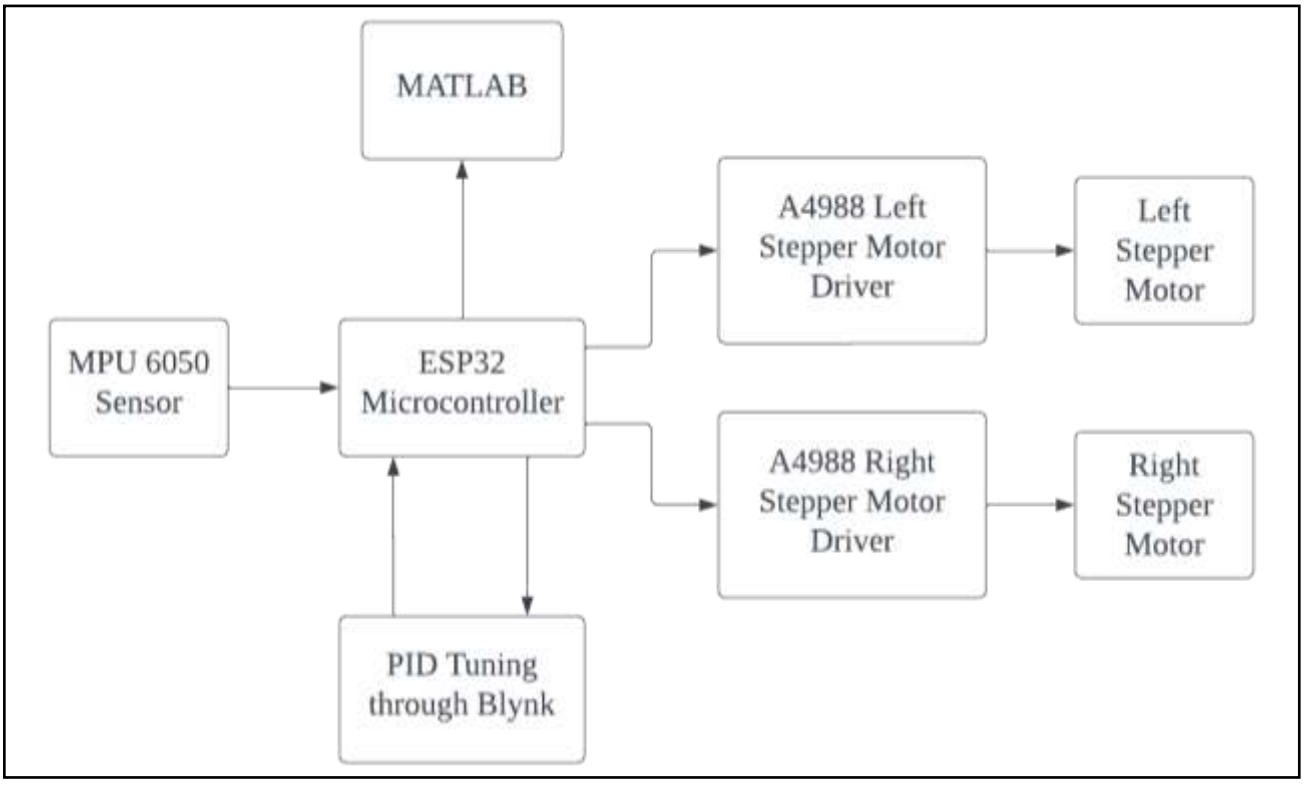

Figure 1. Block diagram of the robot

The robot's response to these disturbances is fed back into the control system, allowing the PID controller to dynamically modify motor commands to maintain stability. This continuous feedback and adjustment mechanism are crucial for the robot's ability to self-balance in a variety of conditions and ensures a high degree of responsiveness and stability (Tun et al., 2022). The integration of both hardware and software elements,

coupled with the iterative tuning of the PID controller based on real-time feedback, forms the foundation of this project's methodology. Based on the closed loop control system, the PID controller can be mathematically written as equation (1).

$$u(t) = K_P e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \quad (1)$$

Hardware Implementation

Initially, the ESP32 Microcontroller is programmed using the Arduino IDE, ensuring all necessary libraries are installed. This stage includes coding, verification, and uploading, with the focus on precise PID parameter settings. The software phase involves calibrating and monitoring the MPU6050 sensor, and adjusting the A4988 stepper motor driver for optimal performance. The final stage integrates hardware and software, facilitating data communication with the IoT platform and enabling a comprehensive performance analysis of the two-wheel self-balancing robot. Thus, the systematic approach ensures the successful development of a PID controller-based robotic prototype.

Software and Hardware Integration

The integration process in the proposed system involves a collaborative interaction between the ESP32 microcontroller, Blynk platform, and MATLAB. The ESP32 sends sensor data to Blynk, which manages and adjusts PID tuning and system parameters based on real-time feedback. This ensures optimal control and system responsiveness. In parallel, the ESP32 transmits data to MATLAB for detailed analysis. This creates a robust feedback loop, with MATLAB processing hardware data and providing insights for system improvement. The system's success hinges on this multidirectional communication, ensuring effective control, monitoring, and continuous refinement between the hardware and the IoT platform.

RESULTS AND DISCUSSION

Parameter Analysis

For $K_p = 4.5$, the robot exhibited a rise time of 0.194 seconds but experienced a fall due to a slow response, eventually stabilizing at 0.928 seconds without disturbances. With $K_p = 9.0$, the rise time was 0.21 seconds, and the robot reached stability at 1.127 seconds without any disturbances. Setting $K_p = 13.5$ resulted in a rise time of 0.1 seconds, with the robot stabilizing at 2.25 seconds, albeit slightly slower than $K_p = 9.0$. $K_p = 18.0$ yielded a remarkably short rise time of 0.097 seconds, with stability achieved after 10.17 seconds, even in the presence of disturbances. For $K_p = 22.5$, the rise time was 0.097 seconds, and stability was reached at 1.627 seconds without disturbances.

Comparing these values in Table 1, it is evident that $K_p = 18.0$ stands out as the best performer. It combines a very short rise time with a long stable time, making it ideal for the control system. This combination allows for rapid responses while maintaining stability even in the face of disturbances. Thus, it emerges as the optimal choice due to its exceptional balance of a fast rise time (0.097 seconds) and a prolonged stable time (10.17 seconds), showcasing its ability to respond quickly and maintain long-term stability in the control system.

Table 1. Proportional Gain K_p Variation Results

K_p Value	Stable tilt angle (°)	Rise time (s)	Stable time (s)
4.5	-0.54	0.194	0.928
9.0	-0.44	0.210	1.127
13.5	0.86	0.100	2.250
18.0	-2.19	0.097	10.51 (with disturbance)
22.5	-0.65	0.097	1.627

Furthermore, different K_d values' also effects on the robot's behaviour. Overshoot, which is closely related to the system's damping ratio, can indicate underdamped or oscillatory behaviour when it is high, potentially leading to instability (Maheshbhai et al., 2020). Settling time is crucial for assessing how quickly a dynamic system stabilizes after a disturbance, reflecting the system's responsiveness and efficiency in reaching a steady state.

For $K_d = 0.3$, the robot exhibited an overshoot of 2425%, with a settling time of 0.716 seconds. With $K_d = 0.6$, the overshoot reduced to 314%, and the settling time was 1.641 seconds. However, for $K_d = 0.9$, the overshoot dramatically increased to 10881%, and there was no discernible settling time, indicating continuous oscillation. Comparing these values in Table 2, it is clear that $K_d = 0.6$ performs the best among the three values.

It exhibits a controlled response with a relatively low overshoot and a reasonable settling time, indicating a balanced and stable performance for the self-balancing robot. Moreover, the influence of different K_i values also effects on the robot's behaviour. For $K_i = 35$, the robot exhibits no overshoot or settling time as it doesn't recover from imbalances.

Table 2. Derivative Gain K_d Variation Results

K_d Value	Overshoot (%)	Settling time (s)
0.3	2425	0.716
0.6	**314**	**1.641**
0.9	10881	-

The average steady state is calculated as 1.67º, which is similar to the steady-state error as the setpoint is 0º. With $K_i = 70$, there is an overshoot of 1092% and a settling time of 2.041 seconds. The input vs. time graph shows the system's resilience, maintaining stability despite disturbances.

The average steady state is calculated as 1.92º. $K_i = 105$ results in a high overshoot of 14516%, with no discernible settling time. The aggressive K_i value leads to instability, and the system doesn't reach a stable state. The average steady state is calculated as -1.33º.

Comparing these values in Table 3, $K_i = 70$ exhibits a balanced and effective performance. It has a moderate overshoot, a reasonable settling time, and maintains stability despite disturbances. The presence of a steady state and an average steady state of 1.92º suggests effective stabilization.

Table 3. Integral Gain K_i Variation Results

K_i value	Overshoot (%)	Settling time (s)	Average steady-state (º)	Steady-state error
35	Not Recover	Not Recover	1.670	1.670
70	1092	2.246	1.920	1.920
105	14516	Not Recover	-1.330	1.330

Therefore, the choice of K_i values significantly impacts the self-balancing robot's performance. $K_i = 70$ strikes a good balance between overshoot, settling time, and stability, demonstrating well-tuned and stable behaviour. Additionally, the self-balancing robot is subjected to multiple disturbances to assess the performance of the control system under various conditions.

Fig. 2 displays the pitch angle (tilt) vs. time of the self-balancing robot when subjected to multiple disturbances. The pitch angle values are predominantly close to 0º, indicating that the robot consistently maintains an upright position. In the context of a self-balancing robot, a pitch angle of 0º represents a perfectly upright position. Table 4 presents the calculated parameters based on the data in Fig. 2 to assess the robot's performance and stability.

Table 4. Overall Performance

K_p	K_i	K_d	Rise time (s)	Overshoot (%)	Settling time (s)	Average steady-state	Steady-state error
18	70	0.6	0.391	225.9	1.932	24.96	24.96

The robot's exhibits efficient control with a fast rise time of 0.391 seconds and controlled overshoot of 225.9%, providing a quick response to disturbances while maintaining stability. Its settling time of 1.932 seconds and consistent average steady-state position of 24.96º highlight its ability to recover and maintain the desired position accurately. The PID controller, with tuned gains ($K_p = 18$, $K_i = 70$, $K_d = 0.6$), plays a crucial role in maintaining stability. The proportional, integral, and derivative components work together to respond effectively to changes in the pitch angle, allowing the robot to correct for disturbances swiftly and accurately.

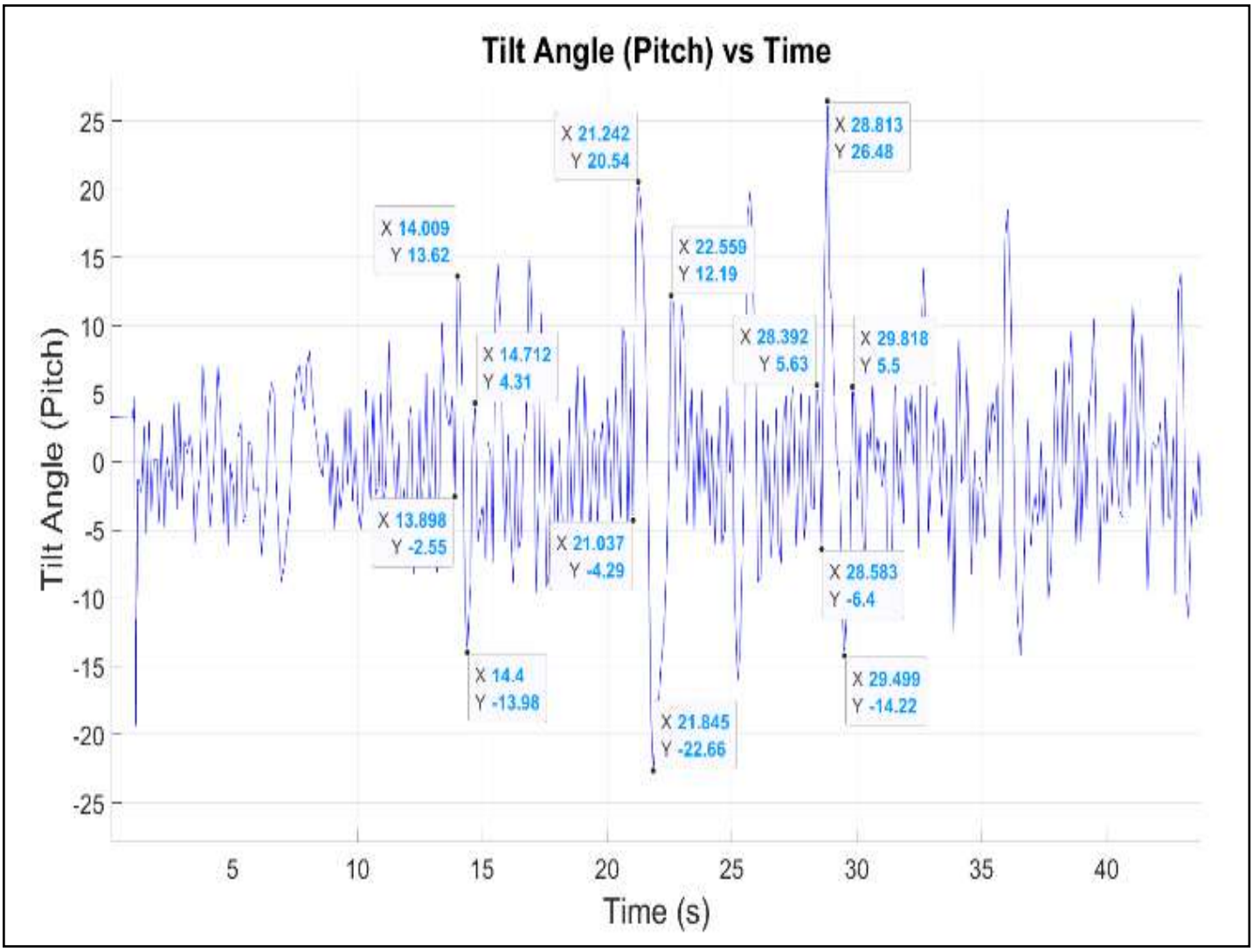

Figure 2. Tilt angle response

Prototype Model Testing

The tests were conducted on various surfaces, including wooden, rubber, and rough surfaces. The robot consistently maintained balance without tipping over on these diverse surfaces, demonstrating its effectiveness in self-balancing with the same tuning parameters. Additionally, tests were performed on different terrains, including uneven surface as shown in Fig. 3, where the robot exhibited a remarkable stability.

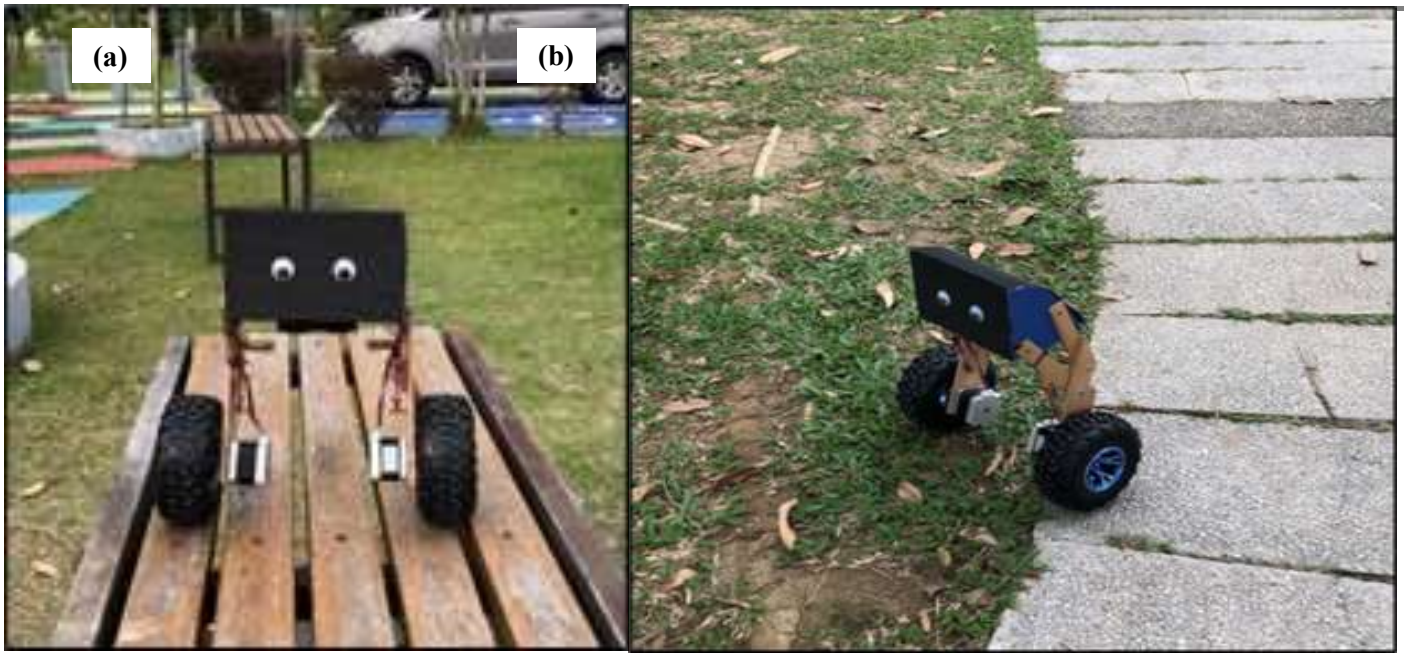

Figure 3. Robot on (a) Uneven surface, (b) Inclined surface

CONCLUSION

This project successfully developed a two-wheel self-balancing robot using a PID controller as a proof of concept for research in PID controller implementation. The project encompassed hardware development with components like the ESP32 microcontroller, MPU6050 sensor, NEMA17 stepper motor, and a4988 stepper motor driver. The use of the Blynk application as an IoT platform allowed for real-time PID gain control, facilitating the tuning of parameters for optimal performance. MATLAB was employed for in-depth analysis, calculating key parameters such as rise time, settling time, overshoot, and steady-state error for varying PID gains.

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to Universiti Teknikal Malaysia Melaka (UTeM) for providing an excellent academic environment, and we are especially grateful to the Faculty of Electronics and Computer Technology and Engineering (FTKEK) for the invaluable guidance and technical support provided by its members throughout this project.

REFERENCES

1. Baltes, J., Christmann, G., & Saeedvand, S. (2023). A deep reinforcement learning algorithm to control a two-wheeled scooter with a humanoid robot. Engineering Applications of Artificial Intelligence, 126(4), 106941. https://doi.org/10.1016/j.engappai.2023.106941
2. Darko, H., Tone, L., Mitja, T., & Oto, T. (2023). Design and Implementation of ESP32-Based IoT Devices. Sensors 2023, 23(15), 6739. https://doi.org/10.3390/s23156739
3. Emrah, A., Kazim, Y., & Eyup, E. (2021). Use of PID control during education in reinforcement learning on Two Wheel balance robot. Journal of Science PART C: DESIGN AND TECHNOLOGY, 9(4), 597-607. https://doi.org/10.29109/gujsc.955562
4. Hao, L., & Sitjongsataporn, S. (2026). Design and implement of smart voice controlled two-wheeled self-balancer for following and avoidance. International Electrical Engineering Transactions, 11(2). https://ph04.tci-thaijo.org/index.php/IEET/article/view/11634
5. John, A.M. (2023). Modeling and control strategies for a two-wheel balancing mobile robot. Master Thesis, University of Arkansas. https://scholarworks.uark.edu/cgi/viewcontent.cgi?article=6514&context=etd

6. Kishore, R., & Rohit, L. (2025). Two wheeled path following self balancing robot. Indian Institute of Technology Delhi. https://web.iitd.ac.in/~subashish/ELP7100/Biwheeled_Robot_Rohit_Kishore_Report.pdf
7. Maciej, S., & Izabela, K. (2025). Influence of control system architecture on mobile robot stability and performance. Sensors 2025, 25, 7353. https://doi.org/10.3390/s25237353
8. Maheshbhai, V.A., Kumar, D., & Sinha, R. (2020). Development of Two Wheeled Robot (TWR) by Single Stepper Driver using PID controller. *Research Square*, 1-33. https://doi.org/10.21203/rs.3.rs-56271/v1
9. Nikita, T., & Prajwal, K.T. (2021). PID Controller Based Two Wheeled Self Balancing Robot. 2021 5th International Conference on Trends in Electronics and Informatics. https://doi.org/10.1109/ICOEI51242.2021.9453091
10. Sandeep, G., Kanad R., & Shamim, K. (2025). Self-balancing mobile robot with Bluetooth control: Design, implementation, and performance analysis. Automation, 6(3), 42. https://doi.org/10.3390/automation6030042
11. Tun, H.M., Nwe, M.S., Naing, Z.M., Latt, M.M., Pradhan, D., & Sahu, P.K. (2022). Research on Self-balancing Two Wheels Mobile Robot Control System Analysis. *Electrical Science & Engineering*, 4(1), 1-7. https://doi.org/10.30564/ese.v4i1.4398
12. Zhang, H., & Nor, N. M. (2025). Control strategies for two-wheeled self-balancing robotic systems: A comprehensive review. Robotics, 14(8), 101. https://doi.org/10.3390/robotics14080101