Design and Implementation of an ANN-Based Adaptive PID Controller for a Self-Balancing Robot

1st Zahidan Akhmad Nuradheli
dept. Eletrical and Electronics Engineering
Universitas Negeri Yogyakarta
Special Region of Yogyakarta, Indonesia
zahidanakhmad.2023@student.uny.ac.id

***Abstract*—The two-wheeled self-balancing robot is a nonlinear and inherently unstable system, posing a significant challenge in the field of control systems. Although the Proportional-Integral-Derivative (PID) controller is commonly applied, its optimal performance is often hindered by fixed gain parameters, which make it less robust to parameter variations and external disturbances. To overcome these limitations, this paper proposes the design and implementation of an adaptive PID controller based on an Artificial Neural Network (ANN), aimed at improving the robot's stability and robustness. The proposed system utilizes a multi-layer perceptron ANN to dynamically and in real-time adjust the PID parameters (Kp, Ki, Kd) based on the robot's current state, including its angular error and angular velocity. This control algorithm is implemented on an embedded platform, a Raspberry Pi 3, with an MPU6050 Inertial Measurement Unit (IMU) as the primary sensor. The trained ANN model is then deployed using the ONNX runtime to ensure efficient inference. Experimental test results show that the proposed ANN-Adaptive PID controller exhibits a more stable response and superior disturbance rejection capabilities compared to a conventional, manually tuned PID controller. This indicates that the proposed approach is a viable solution for enhancing the performance of self-balancing robot control systems.**

Keywords—Self-Balancing Robot, Artificial Neural Network (ANN), PID Controller, Adaptive Control, ONNX

I. INTRODUCTION

In recent years, the two-wheeled self-balancing robot has become a popular research subject in the field of control systems and robotics [1]. As a system analogous to an inverted pendulum, it is characterized by its inherently unstable, multi-variable, and nonlinear dynamics [2], making it an excellent platform for testing and validating advanced control theories. Its maneuverability and low power consumption also make it a promising platform for various applications, from personal transportation to surveillance. The system is often modeled as a mobile inverted pendulum, which serves as the basis for mathematical analysis [3].

The most common approach for controlling a self-balancing robot is the Proportional-Integral-Derivative (PID) controller, mainly due to its simple structure and ease of implementation [1]. However, the main drawback of a conventional PID controller lies in its fixed parameters (Kp, Ki, Kd) These parameters are typically tuned for specific operating conditions and may not perform optimally when the system faces external disturbances or variations in its physical parameters [1]. This limitation often results in degraded performance, such as large oscillations or even instability, when the robot operates in dynamic real-world environments.

To overcome the limitations of conventional PID, various intelligent control strategies have been explored. These include methods based on Fuzzy Logic , Sliding Mode Control (SMC) to improve robustness , and hybrid approaches like Adaptive Neuro-Fuzzy Inference Systems (ANFIS). Among these, Artificial Neural Networks (ANNs) offer a powerful solution due to their ability to learn complex nonlinear relationships and adapt to changing conditions [4]. ANNs can be used to model, identify, and control dynamic systems, including creating adaptive controllers that tune their own parameters online [5].

This paper presents the design and implementation of an ANN-based adaptive PID controller for a two-wheeled self-balancing robot. The proposed control system utilizes a multilayer perceptron (MLP) neural network to continuously adjust the PID gains based on the robot's real-time state variables. The goal is to improve the system's stability and robustness compared to a fixed-gain PID controller. The entire system is implemented on a low-cost embedded platform consisting of a Raspberry Pi and an MPU6050 sensor, with the trained ANN model deployed using the ONNX (Open Neural Network Exchange) format for efficient execution [6]. Experimental results are presented to validate the effectiveness of the proposed controller.

The remainder of this paper is organized as follows. Chapter II provides a review of related works in the field. Chapter III details the hardware design, data collection process, and the design and training of the proposed ANN-Adaptive PID controller. Chapter IV presents and discusses the experimental results, comparing the performance of the proposed controller with a conventional PID. Finally, Chapter V concludes the paper and suggests directions for future work

II. RELATED WORK

The development of control systems for two-wheeled self-balancing robots has been an active area of research. Various approaches have been explored to address the system's inherent instability and nonlinear nature. Based on the literature, these approaches can be broadly categorized as follows:

A. Conventional Control Methods

1) PID Controller

This method serves as the most common foundational method due to its simple structure [1]. However, its primary drawback is the use of fixed gain parameters (Kp, Ki, Kd), which leads to performance degradation when facing external disturbances. Experimental studies have even shown that

conventional PID controllers can fail to maintain the robot's balance under external forces [7].

2) PID Backstepping

As an enhancement to the standard PID, the backstepping method has been proposed as a powerful design tool for nonlinear systems [8]. One implementation utilized three control loops: a backstepping controller for maintaining equilibrium, a PD controller for position, and a PI controller for the direction of motion [8].

3) Linear-Quadratic Regulator (LQR)

This optimal control method has also been implemented and compared against PID. Simulation studies have concluded that LQR has a better performance than PID for the self-balancing subsystem control [9].

B. Fuzzy Logic-Based Intelligent Control

1) Fuzzy Logic Controller (FLC)

This approach is often chosen for its ability to handle uncertainty and imprecision in real-world applications [10]. FLC has been effectively used to improve the robot's balancing ability, especially when subjected to external forces [7].

2) Fuzzy Fractional-Order PID (FOPID)

A more advanced variant, FOPID, has been applied to control the robot on inclined surfaces, demonstrating better control performance and anti-interference capabilities compared to conventional controllers [11].

3) Adaptive Neuro-Fuzzy Inference System (ANFIS)

As a hybrid method, ANFIS combines the learning capabilities of neural networks with the rule-based framework of fuzzy logic. One study proposed an LQR-based ANFIS controller that successfully addressed the robustness issues in LQR and the tuning difficulties associated with fuzzy controllers [2].

C. Neural Network-Based Intelligent Control

1) ANN with Other Methods

Artificial Neural Networks (ANNs) have been integrated with other robust control methods, such as Sliding Mode Control (SMC). In this combination, the ANN's role is to compensate for system noise, thereby improving the overall quality of the control system [12].

2) ANN-Adaptive PID

This area is the most relevant to this research. Several studies have specifically proposed using ANNs to adaptively tune PID parameters. In these approaches, an ANN, such as a Multi-Layer Perceptron (MLP), is trained to dynamically adjust the Kp, Ki, kd gains online based on the system's current state, allowing the controller to expand its robustness and adaptive capabilities [4, 5].

III. METHODOLOGY

This chapter provides a detailed breakdown of the methodology employed in this research. It covers the hardware design, the control system architecture, and the specific design and training process of the implemented Artificial Neural Network (ANN).

A. Hardware Design

The self-balancing robot prototype was constructed using a set of integrated hardware components. The system's hardware architecture is depicted in Fig. 1.

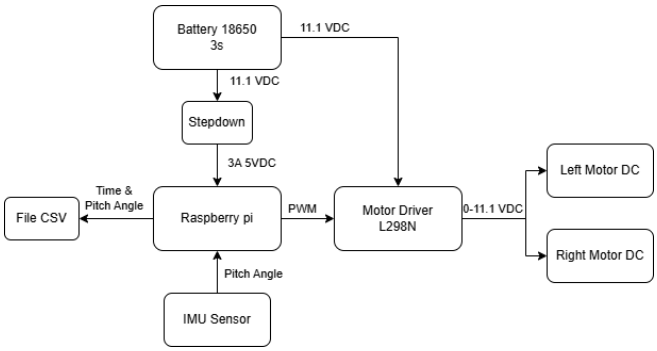

Fig. 1. Hardware Block Diagram.

The main components are as follows:

- **Main Processing Unit:**
  A Raspberry Pi 3 Model B+ serves as the computational core, responsible for running the entire control algorithm and the ANN model inference.
- **Inertial Measurement Unit (IMU):**
  An MPU6050 sensor is used to acquire essential orientation data, namely the pitch angle and angular velocity. To address the sensor's inherent weaknesses—the accelerometer being prone to vibrational noise and the gyroscope being prone to drift over time—a Complementary Filter is implemented [13]. This filter effectively fuses the data from both sensors to produce a stable and accurate angle estimation, which is crucial for the control system's reliability.
- **Actuators:**
  A pair of JGA25-370 DC motors act as the actuators. These motors are responsible for generating the torque required to drive the wheels and maintain the robot's balance.
- **Motor Driver:**
  An L298N driver serves as the interface between the Raspberry Pi and the DC motors. This component receives low-power control signals from the Raspberry Pi and delivers the higher power needed to manage the speed and direction of the motors.
- **Power Source:**
  The system is powered by a 3s Li-ion 18650 battery pack (11.1 VDC). A step-down module is used to regulate the voltage to a stable 5V 3A, ensuring a safe and consistent power supply for the Raspberry Pi.

A photograph of the assembled prototype is shown in Fig. 2, illustrating the physical arrangement of these components.

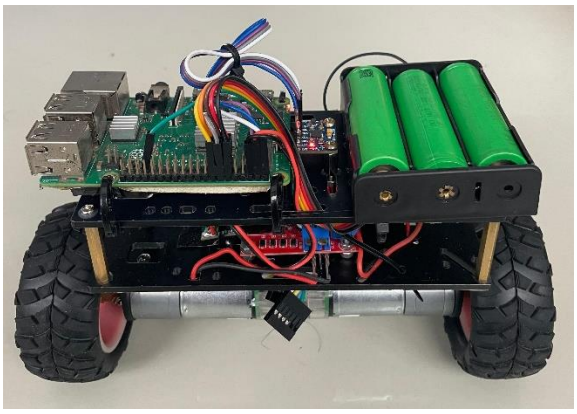

Fig. 2. The self-balancing robot prototype.

It is important to note that due to physical assembly constraints, the IMU sensor is not perfectly centered on the robot's rotational axis. This mechanical imperfection results in a natural balancing angle that is slightly offset from zero. To compensate for this, the control system's setpoint is established at -2.745 degrees instead of 0 degrees, allowing the robot to achieve a stable, upright posture.

B. *Control System Architecture*

The proposed control architecture is a closed-loop system designed to allow the ANN to perform adaptive adjustments to the PID controller in real-time. The overall control system architecture is illustrated in Fig. 3.

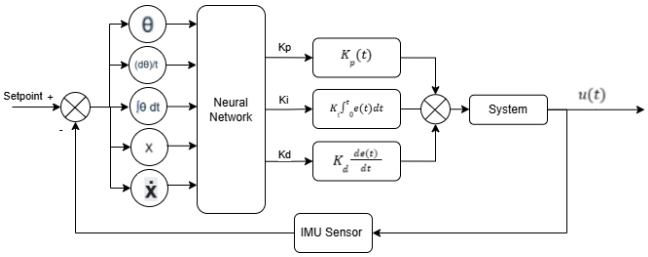

Fig. 3. Control System Architecture.

The control process begins by comparing the setpoint value (-2.745°) with the actual angle (θ) measured by the IMU sensor. This comparison generates an error signal, e(t). This error signal, along with other system state variables (such as angular velocity), is fed as input to the Artificial Neural Network block. The ANN then processes these inputs to generate the adjusted PID parameters (Kp, Ki, Kd) as its output.

These adaptive parameters are then used by the PID algorithm to calculate the final control signal, u(t), which is sent to the motors. The conventional PID control signal is defined in equation (1):

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

Where:

- $u(t)$ is the Control Signal, or the output of the PID controller at time t. In this case, u(t) represents the corrective action (e.g., the voltage or PWM value) sent to the motors to generate the torque required to balance the robot.
- $e(t)$ is the Error Signal at time t. This variable is the difference between the desired angle (the setpoint) and the actual pitch angle measured by the IMU sensor. For your robot, this is calculated as $e(t) = (-2.745°) - \theta actual(t)$.
- Kp is the Proportional Gain. This parameter provides a control action that is proportional to the current error. A larger angular error results in a stronger corrective action.
- Ki is the Integral Gain. This parameter addresses the accumulated error over time. Its purpose is to eliminate steady-state error, ensuring the robot eventually settles precisely at the setpoint angle without any deviation.
- Kd is the Derivative Gain. This parameter provides a control action based on the rate of change of the error. Its function is to provide a damping effect, which reduces overshoot and oscillations as the robot approaches its equilibrium position.

Kp, Ki & Kd are the proportional, integral, and derivative gains that are dynamically updated by the ANN.

C. *ANN Design and Training*

The core of this adaptive approach is the ANN model, which is designed to map system states to optimal control parameters.

1) *Data Generation and Training Dataset*

The training dataset was generated through direct experimentation with the robot:

- **Initial Tuning:** A conventional PID controller was first tuned manually via trial and error until the robot achieved a reasonably stable response. This process yielded the baseline parameters: $Kp = 13.8, Ki = 0.92, Kd = 11.5$.
- **Data Logging:** While the robot operated with these stable parameters, five key state variables—angular error, angular velocity, integral angular error, position, and velocity—were logged to a CSV file.
- **Parameter Variation:** To enrich the dataset, three additional data sets were created by applying small variations to each of the PID parameters individually. From the four resulting data sets, 10 samples were drawn from each, creating a final training dataset of 40 representative samples.

2) *ANN Architecture*

A Multi-Layer Perceptron (MLP) was designed with the architecture shown in Fig. 4.

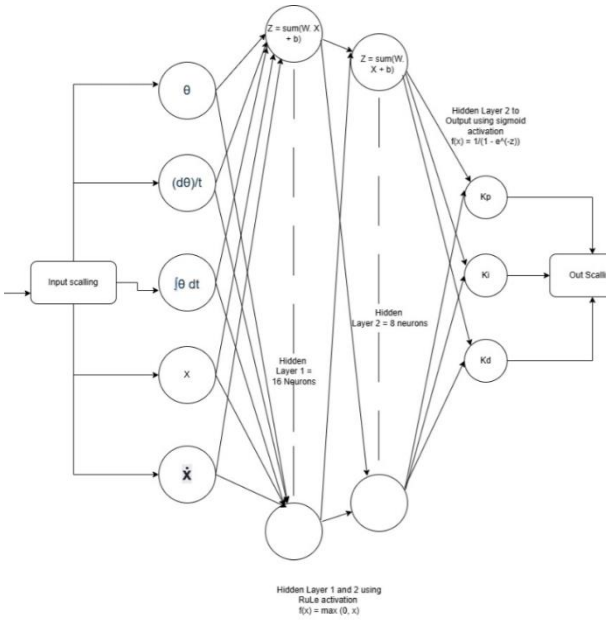

Fig. 4. ANN Architecture.

This structure was chosen for its proven ability to approximate complex nonlinear functions. It consists of:

- **Input Layer (5 neurons):** Receives five key state variables that provide a comprehensive snapshot of the robot's current dynamic condition. These inputs are:
  - Angular Error (θ): The difference between the desired setpoint angle (-2.745°) and the actual measured pitch angle. This is the primary input that drives the proportional response of the controller.
  - Angular Velocity (dθ/t): The rate at which the robot is tilting. This is crucial for the derivative action, allowing the controller to anticipate future error and provide necessary damping to prevent oscillations.
  - Integral Angular Error (∫θ dt): The accumulated error over time. This input is essential for the integral action to eliminate any steady-state error, ensuring the robot precisely maintains its setpoint over the long term.
  - Position (x): The linear position of the robot on the ground.
  - Velocity ($\dot{x}$): The linear velocity of the robot, which gives the network additional context about its overall movement.
- **Hidden Layer 1 (16 neurons):** Utilizes the ReLU (Rectified Linear Unit) activation function to process the features from the input layer.
- **Hidden Layer 2 (8 neurons):** Also utilizes the ReLU activation function, further refining the features for the output layer.
- **Output Layer (3 neurons):** Produces the three values representing the adaptive gains Kp, Ki & Kd.

The ReLU activation function, known for its computational efficiency in deep learning models, is defined in equation (2):

$$f(x) = \max(0, x) \tag{2}$$

The output layer uses the Sigmoid activation function, which is ideal for normalizing the output gains into a consistent range. It is defined in equation (3):

$$f(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

3) Data Scaling and Deployment

Because the Sigmoid function outputs values in the range [0, 1], a data scaling process is essential. System inputs are normalized before being fed into the ANN, and the ANN's output is subsequently de-normalized to obtain the actual PID gain values. The ANN model was trained on a PC and then exported into the ONNX (Open Neural Network Exchange) format. This format was chosen for its high efficiency and portability, allowing the trained model to be deployed and executed optimally on a low-power embedded device like the Raspberry Pi [6].

IV. RESULTS AND DISCUSSION

This chapter presents the implementation and comparative analysis results of two control systems applied to the self-balancing robot prototype. The pitch angle and time data, recorded during the experiments, were processed using Python to generate system response graphs. The performance of each controller is quantitatively evaluated through a transient response analysis, which includes key metrics: rise time, peak time, settling time, overshoot/undershoot, and steady-state error.

The primary analysis in this chapter is a performance comparison between the conventional PID controller with fixed parameters (Kp=13.8, Ki=0.92, Kd=11.5) and the proposed ANN-based Adaptive PID controller. Each test session for both controllers was conducted for approximately 5 seconds to observe and compare their respective capabilities in maintaining the robot's stability.

A. Conventional PID Controller Performance

The performance of the conventional PID controller, which uses fixed, manually tuned parameters, is presented in Fig. 5.

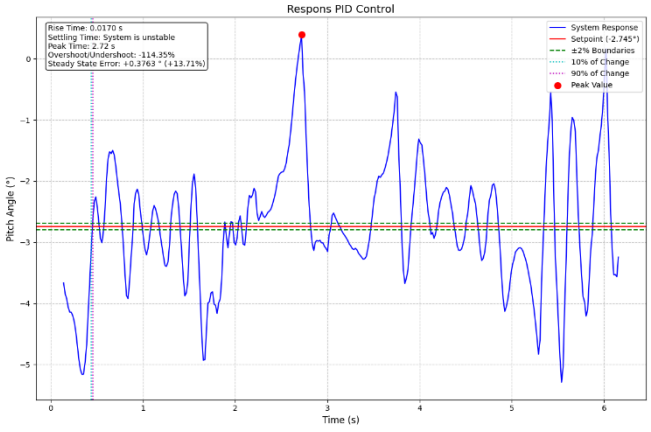

Fig. 5. Response PID Control.

The results show that the system exhibits significant and persistent oscillations around the setpoint. The performance

metrics record a very large overshoot/undershoot, reaching -114.35%, with a steady-state error of +0.3763° (+13.71%). The system's failure to achieve a stable settling time indicates that the static gain parameters are unable to effectively counteract the robot's nonlinear dynamics and inherent disturbances.

B. *ANN-Adaptive PID Controller Performance*

In comparison, Fig. 6 presents the system response when controlled by the ANN-based Adaptive PID controller.

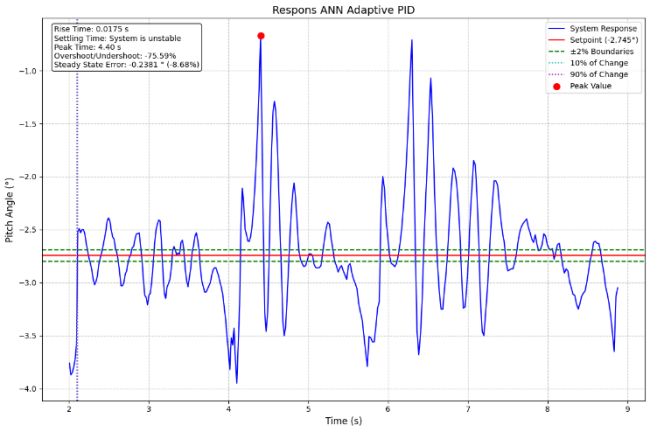

Fig. 6. Response ANN-Adaptive PID Control.

A drastic improvement in stability is visually apparent, where the resulting oscillations are much more damped. This is confirmed by the improvement in the quantitative metrics: the overshoot/undershoot significantly decreases to -75.59%, and the steady-state error is reduced to -0.2381° (-8.68%). Although the oscillations are not completely eliminated, their much smaller amplitude indicates that the adaptive controller is significantly more capable of maintaining the robot's balance.

C. *Discussion and Comparative Analysis*

To facilitate the comparative analysis, the key performance metrics of the transient response for both controllers are summarized in Table I. This table presents a quantitative comparison between the conventional PID controller and the ANN-Adaptive PID controller.

TABLE I. PERFORMANCE COMPARISON OF PID AND ANN-ADAPTIVE PID CONTROLLERS

Response Transient	**Pid Control**	**ANN-Adaptive PID Control**
Rise Time (s)	0.0170	0.0175
Settling Time (s)	-	-
Overshoot / Undershoot (%)	-114.35	-75.59
Peak Time (s)	2.72	4.40
Steady State Error (°)	0.3763	-0.2381

The quantitative results presented in Table I provide a detailed comparison between the Performance of the conventional PID controller and the proposed ANN-Adaptive PID controller.

The analysis reveals several key findings. Firstly, in terms of response speed, the ANN-Adaptive controller exhibits a marginally slower initial response, with a Rise Time of 0.0175s compared to the PID's 0.0170s. It also takes longer to reach its first peak, as shown by the Peak Time (4.40s for the ANN vs. 2.72s for the PID).

However, the most significant improvement is observed in the system's damping capability. The ANN-Adaptive controller drastically reduces the Overshoot / Undershoot from -114.35% to a much more controlled -75.59%. This demonstrates its superior ability to mitigate the aggressive oscillations that are prominent in the conventional PID controller's response.

Despite this improvement, it is crucial to note that neither controller achieved a stable Settling Time within the experiment's duration (indicated by "-"). This, combined with the non-zero Steady State Error for both systems, confirms that both controllers still exhibit persistent oscillations and have not yet reached a truly stable equilibrium. The overall performance, in terms of final stability, is therefore still comparable.

This limitation is likely attributable to the nature of the training data used for the Artificial Neural Network. The model was trained on a relatively small dataset, consisting of only 40 training samples derived from just four variations of PID parameters. This limited dataset may not be sufficient for the ANN to learn a fully robust and generalized control strategy for all possible system states. To achieve superior stability and further enhance the performance of the ANN-Adaptive system, future work should focus on expanding the training dataset. This would involve collecting more data points with a much wider and more varied range of PID parameters, which would enable the neural network to learn the system's dynamics more comprehensively and improve its overall control performance.

V. CONCLUSION AND FUTURE WORK

A. *Conclusion*

This research has successfully designed, implemented, and validated an Artificial Neural Network (ANN) based adaptive PID control system to enhance the stability and robustness of a self-balancing robot prototype. The primary objective of this study was to address the fundamental weaknesses of conventional PID controllers with fixed parameters, which are often inadequate for inherently unstable and nonlinear systems such as the self-balancing robot.

The experimental results presented in Chapter IV clearly demonstrate the superiority of the proposed approach. The ANN-Adaptive PID controller proved to be significantly more effective than the conventional PID controller. Quantitatively, the adaptive controller successfully reduced the overshoot/undershoot from -114.35% to -75.59%, an improvement of approximately 34%. This performance enhancement is directly attributed to the ANN's ability to dynamically adjust the PID gains (Kp, Ki, Kd) in response to the robot's changing states.

Nevertheless, this study also acknowledges that while significant stability improvements were made, neither control system achieved a perfect steady state, as indicated by the persistent oscillations. This limitation is likely attributable to the relatively small dataset used for training the ANN, which consisted of only 40 data samples from four variations of PID parameters. This limited dataset may not have been sufficient

for the ANN to learn the robot's complex dynamics comprehensively.

B. Future Work and Suggestions

Based on the findings and limitations of this study, the following directions for future research are recommended to further refine the system:

1) *Expansion of the Training Dataset*
   The most critical next step is to significantly expand the training dataset. By collecting more data points from a much wider and more varied range of PID parameters, the ANN will be able to learn a more robust and generalized control model, which is expected to eliminate the remaining oscillations and achieve true stability.
2) *Implementation of Online Learning*
   For further development, online learning or reinforcement learning techniques could be explored. This would allow the controller to continuously learn and adapt on its own while operating in real-world environments, enabling it to handle disturbances and conditions not present in the initial training data.
3) *Testing with Varied External Disturbances*
   Further validation using measurable and varied external disturbances (such as physical pushes or uneven surfaces) is necessary to comprehensively evaluate the controller's robustness and disturbance rejection capabilities in a more challenging, real-world context.

In summary, this research successfully demonstrates the viability and effectiveness of using an ANN-based adaptive PID controller to enhance the stability of a self-balancing robot.

REFERENCES

[1] H. M. Omar, A. M. Elalawy, and H. H. Ammar, "Two-wheeled Self balancing robot Modeling and Control using Artificial Neural Networks (ANN)," in *2019 Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2019, pp. 196-200.

[2] I. Chawla, V. Chopra, and A. Singla, "Robust LQR Based ANFIS Control of x-z Inverted Pendulum," in *2019 2nd International Conference on Artificial Intelligence and Cloud Computing (AICAI)*, 2019, pp. 818-822.

[3] F. Grasser, A. D'Arrigo, S. Colombi, and A. C. Rufer, "JOE: A mobile, inverted pendulum," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 1, pp. 107-114, Feb. 2002.

[4] V. Kumar, P. Gaur, and A. P. Mittal, "ANN based self tuned PID like adaptive controller design for high performance PMSM position control," *Expert Systems with Applications*, vol. 41, no. 17, pp. 7995-8002, 2014.

[5] M. Benrabah, K. Kara, A. Oussama, and L. Hadjili, "Adaptive Neural Network PID Controller," in *2019 1st International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, 2019, pp. 1-6.

[6] S. Nandanwar, "Cross-Framework Validation of CNN Architectures: From PyTorch to ONNX," M.S. thesis, Dept. Computer Sciences, Florida Institute of Technology, Melbourne, FL, USA, 2024.

[7] R. Sadeghian and M. T. Masoule, "An Experimental Study on the PID and Fuzzy-PID Controllers on a Designed Two-Wheeled Self-Balancing Autonomous Robot," in *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*, 2016, pp. 313-318.

[8] N. G. M. Thao, D. H. Nghia, and N. H. Phuc, "A PID Backstepping Controller For Two-Wheeled Self-Balancing Robot," in *2010 International Forum on Strategic Technology (IFOST)*, 2010, pp. 76-81.

[9] W. An and Y. Li, "Simulation and Control of a Two-wheeled Self-balancing Robot," in *Proceeding of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Shenzhen, China, 2013, pp. 456-461.

[10] J. Simon, "Fuzzy Control of Self-Balancing, Two-Wheel-Driven, SLAM-Based, Unmanned System for Agriculture 4.0 Applications," *Machines*, vol. 11, no. 4, p. 467, Apr. 2023.

[11] J. Zhang, T. Zhao, B. Guo, and S. Dian, "Fuzzy fractional-order PID control for two-wheeled self-balancing robots on inclined road surface," *Systems Science & Control Engineering*, vol. 10, no. 1, pp. 289-299, 2022.

[12] D.-M. Nguyen, V.-T. Nguyen, and T.-T. Nguyen, "A neural network combined with sliding mode controller for the two-wheel self-balancing robot," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 592-601, Sep. 2021.

[13] S. Das, S. Nandy, S. Chakraborty, J. Singh, and T. Halder, "Implementation of Complimentary filter on MPU 6050," *World Scientific News*, vol. 199, pp. 218-234, 2025.

[14] H. A. Mayer and R. Schwaiger, "Differentiation of Neuron Types by Evolving Activation Function Templates for Artificial Neural Networks," in *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN'02)*, vol. 2, 2002, pp. 1773-1778.

[15] G. S. Martins, J. F. Ferreira, D. Portugal, and M. S. Couceiro, "MoDSeM: Modular Framework for Distributed Semantic Mapping," in *2nd UK-RAS Robotics and Autonomous Systems Conference*, Loughborough, 2019, pp. 12-15.

[16] . Shi, S. Xu, and G. Zhao, "Variable Universe Type-II Fuzzy Logic Control Design for the GOOGOL's Two-Wheeled Self-Balancing Robot," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 1500-1505.