Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

Real-Time Implementation of a Self-Balancing Robot using PID Controller

R. Vishnu Varthan
Dept. Robotics and Automation
PSG College of Technology
Coimbatore, India

Sanjay Ganesh. S
Dept. Robotics and Automation
PSG College of Technology
Coimbatore, India

Anbarasi M P
Dept. Robotics and Automation
PSG College of Technology
Coimbatore, India

***Abstract*— This paper introduces the design and implementation of a self-balancing two-wheeled robot based on a Proportional-Integral-Derivative (PID) control system. An MPU6050 sensor module, integrating a 3-axis gyroscope and accelerometer, detects the robot's tilt angle in real time. The PID controller calculates correction signals based on deviation from the vertical and applies them to the motors through an L298N dual H-bridge motor driver, enabling dynamic speed and direction changes. An Arduino microcontroller implements the control algorithm and manages sensor-motor interactions. A MATLAB/Simulink simulation and real-life implementation of the idea validates the robot's ability to stabilise under disturbances. The results show improved balance, minimal tilt, and smooth corrective manoeuvres, verifying the design and laying the groundwork for future enhancements.**

***Keywords* — Self-balancing robot, MPU6050, PID controller, L298N motor driver, tilt angle, mobile robotics, dynamic stabilisation.**

I. INTRODUCTION

In recent years, mobile robots have gained significant popularity due to their exceptional ability to transport heavy loads from one point to another more quickly and efficiently than human labour. Among the various types of mobile robots, self-balancing robots stand out for their unique ability to maintain stability using advanced control systems and sensor feedback. Self-balancing robots are typically two-wheeled robots that can support themselves upright using data from sensors and a closed-loop feedback control system. A well-known example of this technology is the Segway, a personal transporter that uses self-balancing mechanisms to carry a person from one location to another. Another popular application is the Ninebot, a compact electric scooter that leverages similar self-balancing techniques for smooth and stable movement. In recent times, self-balancing robots have found growing applications in logistics, especially for transporting loads in confined spaces like warehouses, distribution centers, and factories. Their compact design and ability to navigate narrow pathways make them ideal for tasks requiring high precision and safety. Additionally, they are being explored in areas such as personal mobility, automated delivery systems, medical assistance robots, and even entertainment robots. This project involves the design and development of a self-balancing robot using an Arduino Uno as the primary microcontroller. A 2-cell battery pack powers the system and employs an L298N motor driver module to control the motors. An MPU6050 sensor detects the robot's orientation by measuring the angle of tilt through gyroscopic and accelerometric data. A PID-based feedback system is implemented in the code to continuously adjust the motor speed based on real-time pitch angle readings, enabling the robot to maintain balance dynamically.

Self-balancing robots have become a notable area of research within mobile robotics, especially in applying control theory to real-world balancing systems. Grasser *et al.* introduced one of the earliest models, "Joe," a two-wheeled robot that uses gyroscopes and accelerometers for posture detection and applies feedback control to maintain balance. Their work established the foundational principles for dynamic stabilisation in mobile platforms. PID controllers are frequently used due to their simplicity and efficiency. Khan *et al.* successfully implemented a PID-based control strategy on a two-wheeled balancing robot and demonstrated that appropriate tuning of gain values leads to effective balance correction, even in the presence of disturbances. The MPU6050 module, which integrates a 3-axis accelerometer and gyroscope, has gained widespread adoption due to its compactness and affordability. Sharma and Kaur utilised the MPU6050 in a self-balancing robot and improved the tilt estimation using a complementary filter for sensor fusion. Their findings support the sensor's capability for reliable real-time angle detection. The sensor's dual data output—acceleration and angular velocity—makes it ideal for precise motion tracking in dynamic environments.

Motor control in balancing robots is commonly achieved using L298N dual H-bridge drivers. Patil et al. used an Arduino microcontroller in conjunction with an L298N module to control the motors dynamically based on the PID output. Their design proved effective in managing both direction and speed to maintain equilibrium. The L298N's ability to handle high current and bidirectional control enhances its suitability for real-time balancing tasks. MATLAB and Simulink are commonly employed for simulation before physical implementation. According to Kumar and Singh, simulation provides insight into system behaviour and enables parameter tuning in a controlled environment, reducing the time and risk involved in hardware testing. These tools allow for efficient development and refinement of control algorithms before deployment. The combination of the MPU6050 sensor, Arduino-based PID control, and L298N motor driver provides a reliable and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

efficient foundation for developing self-balancing robots. This setup allows for real-time stabilisation through continuous feedback and motor adjustments. The inclusion of both simulation and real-world testing validates the system's performance and robustness. The modular design supports future enhancements, such as wireless control, autonomous navigation, and obstacle avoidance. These developments can expand the robot's applications in fields like personal assistance, surveillance, and mobile automation.

II. COMPONENTS AND SYSTEM ARCHITECTURE:

This section outlines the key hardware components and their interconnections that form the core of the self-balancing robot. Each component plays a specific role in enabling balance, control, and movement. The architecture includes the microcontroller, sensors, power source, motor driver, and chassis, all working together to maintain the robot's stability and performance.

A. Arduino Uno

The Arduino Uno is a microcontroller board based on the ATmega328P. It is used to collect data from various sensors and process this information to control actuators such as motors. In this robot, the Arduino Uno reads data from the MPU6050 and implements control algorithms (like PID) to maintain balance.

B. MPU6050

The MPU6050 is a 6-axis MEMS (Micro-Electro-Mechanical Systems) motion-tracking device that integrates a 3-axis accelerometer and a 3-axis gyroscope on a single chip. It is used to measure the robot's orientation and angular motion, providing real-time pitch values required for balance control.

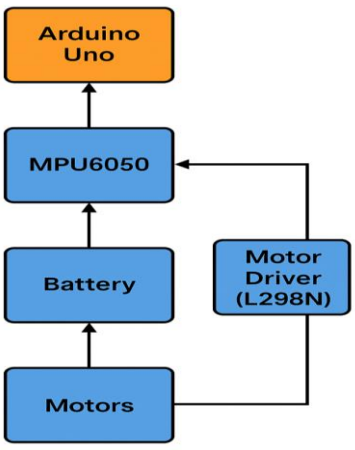


Figure 1: Flow Diagram of the Electrical Connections in the Self-Balancing Robot

C. Power Source

The robot is powered by two rechargeable Li-ion batteries, each rated at 3.7V and 1200mAh (1.2A capacity). These batteries are connected in series or parallel depending on the voltage and current requirements of the motors and Arduino.

D. L298N Motor Driver

The L298N motor driver module provides a simple interface for motor and pin connections, enabling control over both the direction and speed of the BO motors. It acts as an intermediary between the Arduino and the motors, supporting bidirectional control and PWM-based speed regulation.

E. BO Motor

BO (Battery Operated) motors are lightweight DC geared motors typically used in small robotics projects. This robot uses two BO motors with a maximum speed of 300 RPM, which provide the torque and speed required for balancing and movement.

F. Chassis

The chassis is a three-tier frame:

- The bottom level houses the battery pack,
- The middle level contains the Arduino Uno and L298N motor driver,
- The top level holds the MPU6050 sensor for optimal tilt detection.

III. PID CONTROLLING

This section explains the role of the Proportional-Integral-Derivative (PID) controller in achieving dynamic stability in the self-balancing robot. It describes how the PID algorithm works, breaks down each component of the controller, and discusses various tuning methods to optimise system performance. The section also outlines the working principle of the robot's control loop and the mathematical formulation used for balance correction.

A. PID Controller

A Proportional-Integral-Derivative (PID) controller is utilised in this project to maintain balance in a two-wheeled self-balancing robot. As a feedback control system, the PID controller continuously monitors the system's error, defined as the deviation between the setpoint and the actual state. Based on this error, corrective signals are generated to adjust the system output accordingly. In contrast to basic on/off control, PID control offers a smoother response and improved accuracy, making it highly suitable for real-time balancing applications.

1) Proportional (P) Control:

The proportional component generates an output that is directly proportional to the current error. It compares the setpoint and the actual position, multiplies the error by the proportional constant (Kp), and adjusts the output accordingly. Technically, the greater the tilt angle, the faster the wheels will turn in response. If the error becomes zero, the output also drops to zero. However, proportional control

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

alone cannot eliminate steady-state error, and higher gain values may lead to overshoot or instability.

2) Integral (I) Control:

The integral component compensates for the steady-state error that persists in proportional control. It accumulates the error over time and adjusts the output until the error reaches zero. That is, the longer it stays tilted, the faster the wheels turn. Although the I-controller helps in fine-tuning the system, excessive integral action may result in overshooting and sluggish response. Thus, the integral gain (Ki) must be carefully tuned.

3) Derivative (D) Control:

The derivative component predicts future behaviour by analysing the rate of error change. The faster the tilt changes, the faster the wheels will turn. It improves system responsiveness by countering the lag introduced by the integral term. The derivative gain (Kd) controls how strongly the system reacts to changes, resulting in faster settling times and improved stability.

B. PID Tuning Methods

To obtain optimal performance, the PID controller must be properly tuned. Tuning determines the appropriate values for Kp, Ki and Kd based on the system dynamics. Several methods are used to tune the PID controller:

1) Trial-and-Error Method:

In this intuitive method, the controller parameters are adjusted while the system is operating. Initially, Ki and Kd are set to zero, and Kp is gradually increased until the system begins to oscillate. Then, Ki is increased to eliminate the oscillations, and Kd is adjusted to speed up the response and stabilise the output.

2) Ziegler-Nichols Method:

This method involves setting Ki and Kd to zero and increasing Kp until sustained oscillations occur. The value of Kp at this point is referred to as the ultimate gain, and the oscillation period is noted. These values are then used to compute the optimal PID gains using Ziegler-Nichols tuning formulas.

3) Process Response Curve Method:

This open-loop technique involves applying a step input to the system and recording the output response. The slope, dead time, and rise time of the response curve are analysed to derive the PID gain values. This method is particularly suitable for modelling-based systems where closed-loop tuning may be risky.

C. Working Principle

The core of the self-balancing robot consists of an Arduino microcontroller that processes tilt data from an MPU6050 sensor module, which combines a 3-axis gyroscope and accelerometer. When the robot tilts away from its neutral position, the Arduino calculates the necessary correction using the PID algorithm and transmits commands to the L298N motor driver. The driver then adjusts the speed and direction of the motors to restore balance.

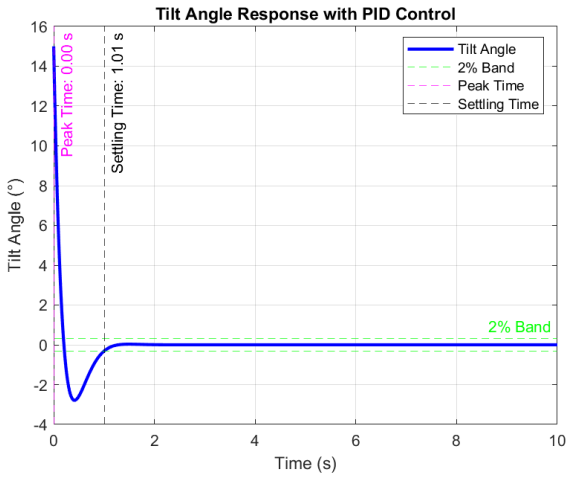
The control output is computed using the following equation:

$$\textbf{Output} = k_p e(t) + k_i \int e(t)\,dt + k_d de(t)\,/\,dt$$

where Kp, Ki, and Kd are the proportional, integral, and derivative gains, respectively. These gains are tuned to minimise error and ensure that the robot maintains its upright position.

IV. RESPONSE ANALYSIS

To evaluate the effectiveness of the PID controller in maintaining balance, a simulation was carried out where the robot initially begins with a tilt of 15 degrees. The system's response was analysed by observing three key parameters: tilt angle, angular velocity, and control signal over time.



A. Figure 2: Tilt Angle vs Time

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

This graph shows the tilt angle of the robot as it attempts to balance itself. Initially, the robot starts with a tilt of 15°, and as the PID controller begins to operate, the tilt angle gradually reduces towards zero. The curve shows a smooth and stable correction process with slight overshoot and negligible steady-state error. This behaviour indicates that the controller parameters are well-tuned, allowing the robot to regain an upright position effectively.

B. Figure 3: Angular Velocity vs Time

The angular velocity plot reflects how quickly the robot's orientation is changing. The velocity initially spikes as the controller reacts to the large initial error, then oscillates before settling close to zero. These oscillations are typical of a balancing system as it fine-tunes its position. The damping of oscillations over time suggests the presence of good

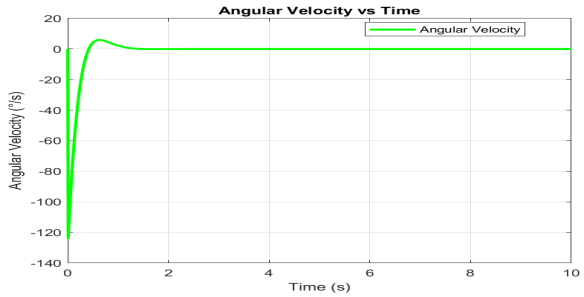


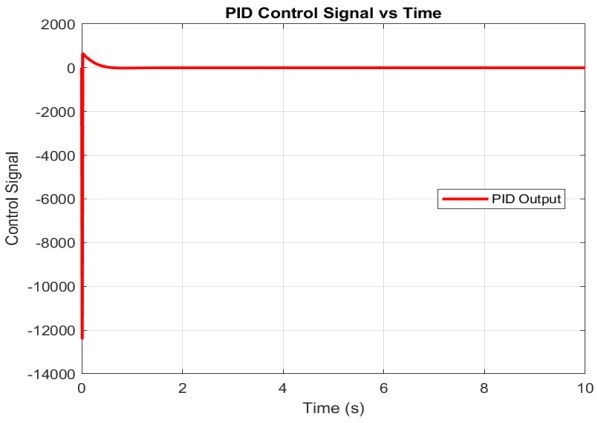
Figure 3: Angular Velocity vs Time

derivative control, helping to prevent excessive speed and ensuring smooth correction.

C. Figure 4: Control Signal vs Time

This plot represents the control signals (motor commands) generated by the PID controller. The graph begins with a strong response, which corresponds to the system's need to counter the initial tilt. As the robot approaches a balanced state, the control signal reduces in magnitude and stabilises near zero. This behaviour confirms that the motors are being appropriately regulated in response to real-time feedback from the sensor.

Figure 4: PID Control Signal vs Time



The results reflect a typical second-order underdamped system. The standard time-domain response of such a system is characterised by:

$$\theta(t) = \theta_0 \left(1 - e^{-\zeta\omega_n t}\left(\cos(\omega_d t) + \frac{\zeta\omega_n}{\omega_d}\sin(\omega_d t)\right)\right)$$

where $\theta(t)$is the tilt angle, θ_0 is the initial disturbance angle (15°), ζ is the damping ratio, ω_n is the natural frequency, and $\omega_d = \omega_n\sqrt{1-\zeta^2}$ is the damped natural frequency. From the simulation, the system exhibited a rise time of approximately 1.50 seconds, a peak time of 0.23 seconds, and a settling time of about 1.01 seconds. The overshoot was minimal at 1.01%, indicating a well-damped and stable response. These values fall within acceptable control system benchmarks, validating the effectiveness of the PID controller. The control signal, as plotted, showed smooth convergence without oscillation, suggesting that the proportional, integral, and derivative gains were appropriately tuned. The robot's ability to recover quickly and with minimal overshoot confirms the controller's capability to ensure stability and responsiveness, making it suitable for real-time balance correction in dynamic environments.

V. REAL-TIME IMPLEMENTATION

The real-time prototype of the self-balancing robot, as illustrated in Figures 5 through 8, was designed with an emphasis on both functional performance and structural stability. The mechanical assembly comprises three precision-cut acrylic sheets, vertically stacked and separated by 60 mm spacers to allow for optimal spatial organisation of internal components. Two DC motors equipped with rubber wheels are employed to provide sufficient traction, thereby enhancing grip and ensuring stable locomotion. The base layer of the structure houses a two-cell lithium-ion battery pack, deliberately positioned at the lowest point to lower the system's center of gravity. This configuration contributes significantly to the robot's balance and resistance to tipping.

Figure 5: Self-Balancing Robot (Front View)

The middle layer accommodates the Arduino Uno microcontroller and the L298N motor driver module. Wiring is routed carefully to reduce clutter and prevent electromagnetic interference, thereby promoting system reliability. The MPU6050 sensor module, which integrates a 3-axis accelerometer and a 3-axis gyroscope, is mounted centrally on the topmost layer. Its elevated and symmetrical placement minimises the influence of motor-induced vibrations and allows for more precise acquisition of angular position and motion data. The overall prototype reflects a well-balanced integration of mechanical design and electronic control. By prioritising accurate weight distribution and minimising structural asymmetries, the robot is capable of achieving reliable real-time balancing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

Moreover, the layered construction facilitates ease of maintenance and scalability for future modifications.

During testing, the prototype demonstrated effective balance control across various surface types, including smooth tiles and moderately uneven terrain. The PID-controlled feedback loop responded promptly to disturbances, with the motors adjusting their speed and direction in real time to counteract tilting. The symmetrical design and low center of gravity contributed to reduced oscillations and quicker settling times. Furthermore, the modular acrylic frame provided not only rigidity but also ease of access for component tuning and replacement. These practical advantages affirm the design's suitability for experimental research and educational applications in robotics and control systems.

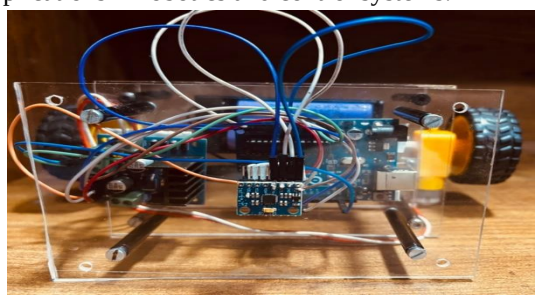


Figure 6: Self-Balancing Robot (Top View)

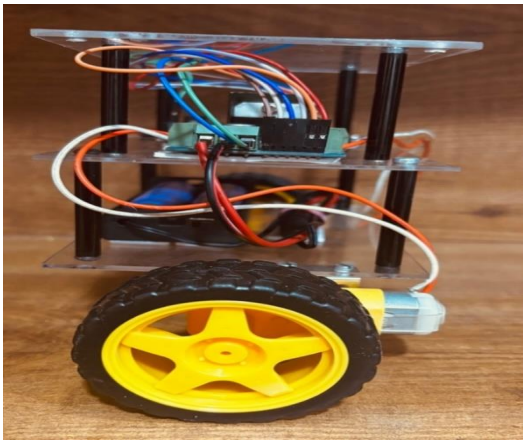


Figure 7: Self-Balancing Robot (Right Side View)

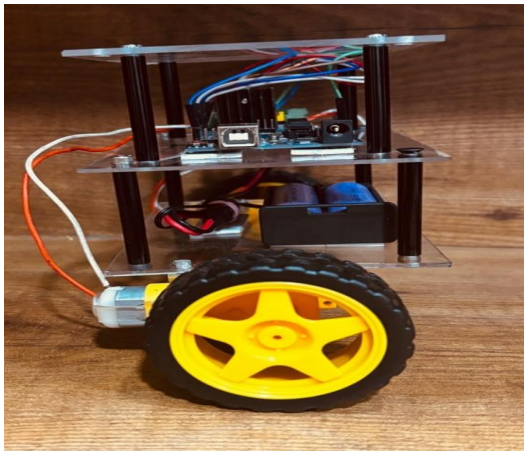


Figure 8: Self-Balancing Robot (Left Side View)

VI. APPLICATIONS

Self-balancing technology has evolved beyond basic robotics experiments to become a foundational element in a wide range of industries. Its ability to maintain equilibrium dynamically, even in unstable or unpredictable environments, enables innovative solutions in transportation, healthcare, automation, and consumer electronics. This section outlines key domains where self-balancing systems are making a significant impact.

A. Personal Transportation Device

Self-balancing technology plays a pivotal role in modern personal transport solutions, offering a compact, efficient, and intuitive mode of movement, particularly in urban environments and large campuses. Examples include the Segway and Ninebot, which utilise dynamic stabilisation to safely transport users by adapting to body posture and tilt.

B. Medical and Assistive Devices

In the healthcare sector, self-balancing systems are being integrated into mobility aids, such as smart walkers and wheelchairs. These devices actively maintain balance and prevent tipping, enhancing safety and independence for individuals with reduced mobility or in rehabilitation.

C. Construction and Agriculture

Self-balancing robots offer a reliable solution for navigating uneven and rugged terrain, which is common in construction sites and agricultural fields. These robots can carry loads, perform monitoring, and reduce human involvement in hazardous environments, thereby improving both safety and efficiency.

D. Consumer Robotics And Entertainment

In the realm of robotic toys and entertainment systems, balancing algorithms allow devices to execute sophisticated movements and interactive behaviours while maintaining dynamic stability. A notable example is Boston Dynamics' Atlas robot, which uses advanced self-balancing mechanisms to execute real-world tasks such as walking, jumping, and backflipping, demonstrating the potential of these technologies in future humanoid robots.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

VII. CONCLUSION

This paper successfully demonstrates the design and implementation of a two-wheeled self-balancing robot that maintains upright stability through real-time sensor feedback and PID-based control. By integrating components such as the MPU6050 for tilt detection, the Arduino Uno for processing, and the L298N motor driver for motor actuation, the robot effectively responds to disturbances and maintains balance through continuous closed-loop feedback. This proposal confirms that even with minimal hardware, a well-tuned PID controller can stabilise the system with precision. Moreover, the robot lays a strong foundation for further enhancements, including wireless control, autonomous navigation, and obstacle avoidance using ultrasonic sensors. Beyond its educational and experimental value, the system showcases the potential of self-balancing technology in various real-world applications, from personal transport to medical assistance and industrial automation, consumer robotics and entertainment. With minor improvements and feature additions, this model can evolve into a powerful prototype for future robotics research and development.

VIII. REFERENCES

[1] H. Ying, G. Yang, H. Lan, and L. Yunbo, “Design and implementation of a multifunctional self-balancing mobile platform,” Proc. 2015 27th Chinese Control Decis. Conf. CCDC 2015, pp. 5693–5697, 2015

[2] Two-wheel Balancing Robot; Review on Control Methods and Experiments M.R.M. Romlay, M.I. Azhar, S.F. Toha, M.M. Rashid.

[3] R. Ping, M. Chan, K. A. Stol, and C. R. Halkyard, “Annual Reviews in Control Review of modelling and control of two-wheeled robots,” Annu. Rev.. Control, vol. 37, no. 1, 2013, pp. 89–103.

[4] R. S. Martins and F. Nunes, “Control System for a Self-Balancing Robot,” in 4thExperiment@ International Conference, 2017. Project I, pp. 297–302.

[5] C F. Hsu, C. T. Su, WF. Kao, and B. K. Lee, “Vision-Based LineFollowing Control of a Two-Wheel Self-Balancing Robot,” Proc. - Int. Conf. Mach. Learn. Cybern., vol. 1, pp. 319–324, 2018

[6] N. Maniha, A. Ghani, F. Naim, and T. P. Yon, “Two Wheels Balancing Robot with Line Following Capability,” Int.J. Mech. Mechatronics Eng., vol. 5, no. 7, 2011.

[7] O. Y. Chee and M. S. B. Zainal Abidin, “Design and development of two-wheeled autonomous balancing robot,” SCOReD 2006 -Proc. 2006 4th Student Conf. Res. Dev. "Towards Enhancing Res. Excell. Reg., no. SCOReD, pp. 169–172, 2006.

[8] E. Philip and S. Golluri, “Implementation of an Autonomous Self-Balancing Robot Using Cascaded PID Strategy,” in 6th International Conference on Control, Automation, and Robotics, 2020, pp. 74–79

[9] M. Mithil, G. S. Thejas, S. K. Ramani, and S. S. Iyengar, “A Low Cost Multi-Sensorial Data Fusion for High Speed Obstacle Avoidance Using 3-D Point Clouds and Image Processing in Self Balancing Robots,” 2017 2nd Int. Conf. Emerg. Comput. Inf. Technol. ICECIT 2017, pp. 1–9, 2018.

[10] C. H. G. Li and L. P. Zhou, “Training end-to-end steering of a self-balancing mobile robot based on RGB-D image and deep convNet,” IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM, vol. 2020-July, pp. 898–903, 2020