International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Two Wheeled Self Balancing Robot

Cheluva Kumar K[1], Hemanth Kumar A R[2], Krithik P[3], Channakeshava K R[4], Manasa M[5]

[1,2,3,4,5]Department of Electronics and Communication Engineering, Sapthagiri College of Engineering Bengaluru-57, KA, India

Abstract

A self-balancing robot is a two-wheeled system designed to maintain its upright position using the principle of an inverted pendulum. It continuously adjusts its orientation and movement based on the real-time feedback from onboard sensors such as the gyroscope and an accelerometer. These sensors relay important data regarding the robot's tilt and angular velocity, which are processed by an Arduino microcontroller. The microcontroller implements a PID (Proportional-Integral-Derivative) control algorithm to make precise corrections, ensuring stability by dynamically adjusting the speed and direction of the wheels. The primary objective of this project is to develop a low-cost self-balancing robot capable of maintaining its balance while responding to external disturbances. Such robots have significant applications in warehouse automation, robotic assistants, and compact personal transportation systems due to their maneuverability and efficient space utilization. This paper provides a detailed exploration of the robot's design, hardware components, control algorithms, and tuning methods required to achieve effective balancing. Additionally, it discusses the advantages and limitations of this approach, including cost considerations, real-world applications, and challenges encountered during implementation.

Keywords: Arduino, Self-Balancing Robot, Inverted Pendulum, PID Control, Gyroscope, Accelerometer, Robotics.

INTRODUCTION

A self-balancing robot is a two-wheeled robotic system that maintains its upright position using the inverted pendulum principle, where continuous adjustments are made to counteract any tilting motion. Unlike traditional four-wheeled robots that naturally remain stable due to a wide base, a two-wheeled robot is inherently unstable and requires real-time correction to stay balanced. This correction is achieved using a feedback control system, which continuously reads the tilt angle and angular velocity, processes the data, and applies necessary motor adjustments. The core of this project is the MPU6050 sensor, which consists of a 3-axis accelerometer for measuring tilt angle and a 3-axis gyroscope for detecting angular velocity. By combining these readings, the system determines the robot's orientation and makes adjustments accordingly. The motor control mechanism, driven by an Arduino Nano and a PID (Proportional-Integral-Derivative) controller, ensures that balance is maintained efficiently. Self-balancing robots have applications in fields such as autonomous transportation, warehouse automation, and service robotics, making them a valuable area of study.

A. Motivation

The motivation behind designing a self-balancing robot stem from the limitations of traditional mobile

robotic systems. Conventional robots with four or more wheels provide inherent stability but lack the maneuverability needed for compact environments. A two-wheeled balancing robot, on the other hand, offers a smaller footprint and greater agility, allowing it to move efficiently in confined spaces. This feature makes self-balancing robots ideal for applications such as indoor robotics, warehouse navigation, and robotic assistants. Additionally, these robots possess superior maneuverability, as they can turn in spot, making them viable for areas with obstacles or dynamic environments. Another key advantage is cost-effectiveness; self-balancing robots require fewer components, reducing both material costs and power consumption. By utilizing widely available components such as Arduino, MPU6050 sensors, and motor drivers, this project aims to develop an affordable yet efficient balancing system. The principles explored in this project are widely applicable, as self-balancing robots are used in devices like Segways, autonomous delivery robots, and robotic medical assistants. Thus, studying and implementing this system contributes to the advancement of low-cost robotic mobility solutions.

B. Objective of the Research

The primary objective of this project is to design and develop a self-balancing robot that can autonomously maintain its balance using real-time sensor feedback and a PID control algorithm. The system must be able to detect and correct deviations from its upright position while also allowing for controlled movements such as forward motion, backward motion, and turning. Additionally, the robot should be capable of recovering from external disturbances, such as being pushed, by quickly compensating for the resulting tilt. Another important objective is to integrate manual control via a mobile app or joystick, allowing users to navigate the robot while it maintains balance autonomously. The PID controller plays an important role in ensuring stability by adjusting motor speeds based on proportional, integral, and derivative calculations of the tilt error. By achieving the above objectives, the research lays the groundwork for further enhancements, such as autonomous navigation, obstacle detection, and adaptive learning algorithms.

C. Literature Survey

Self-balancing robots have been a subject of extensive research due to their applications in automation, personal transportation, and robotics. Various studies have examined different aspects of design, control, and optimization for achieving stable and efficient balance control in two-wheeled robots.

Harris and Adams (2006) [1] laid the foundation for self-balancing robot design by incorporating a feedback controll systems that continuously adjusted the robot's motion to maintain stability. Their study highlighted the fundamental challenges associated with designing two-wheeled robots and emphasized the importance of precise control mechanisms. Similarly, Borenstein and Koren (1991) [2] examined the role of gyroscopic sensors and accelerometers in measuring the tilt and orientation of self-balancing robots. Their research provided important insights into real-time sensor data processing for accurate balance estimation.

Control algorithms play a critical role in achieving stability in self-balancing robots. Li and Li (2007) [3] conducted a comparative analysis of different control strategies, particularly PID control and fuzzy logic control, for two-wheeled robots. Their findings demonstrated the advantages of using PID controllers for precise balance correction, while fuzzy logic provided adaptability to varying conditions. Further advancements in control techniques were introduced by Mori and Okada (2013) [4], who implemented an adaptive control system using Kalman filtering. Their approach improved balance accuracy by filtering out sensor noise and providing a more reliable state estimation.

Energy efficiency is another critical factor in self-balancing robot design. Wang, Hins, and Suryavanshi

(2015) [5] optimized control algorithms to enhance energy efficiency in self-balancing robots. Their study aimed to reduce energy consumption while preserving balance, enhancing the robots' feasibility for long-term use. Additionally, Zhang and Wu (2019) [6] investigated trajectory tracking by implementing Model Predictive Control (MPC), ensuring precise movement control in dynamic environments.

Recent progress in machine learning has significantly improved the capabilities of self-balancing robots. Chien and Lin (2020) [7] applied reinforcement learning (RL) to improve the robot's ability to adapt to different terrains and disturbances. Building upon this, Zhou and Xu (2022) [8] extended RL applications by incorporating deep reinforcement learning (DRL) to enable self-balancing robots to navigate complex and unpredictable environments, such as uneven terrains. Their study showcased how deep learning models could enhance robotic adaptability beyond traditional control methods.

In addition to control and learning mechanisms, material optimizations has also been checked to increase the efficiency of self-balancing robots. Kim, Pandit, and Sikka (2023) [9] investigated the use of lightweight composite materials for robotic frames, reducing total weight while maintaining structural integrity. Their research showed that material selection plays a vital role in improving energy efficiency and performance.

Overall, these studies contribute to the ongoing development of self-balancing robots, addressing key challenges in stability, control, energy efficiency, learning capabilities, and material design. By integrating advancements in control algorithms, machine learning, and materials, researchers continue to refine self-balancing robots for diverse real-world applications.

WORKING PRINICIPLE

A two-wheeled self-balancing robot tries to maintain the upright position by dynamically adjusting its posture using a feedback control system based on the Inverted Pendulum model. The robot continuously measures its tilt angle with accelerometers and gyroscopes, providing real-time orientation data connected as shown in figure 1. A control algorithm, typically a Proportional-Integral-Derivative (PID) controller, processes this data to compute necessary motor adjustments for balance correction.

When the robot tilts forward or backward, the system activates the motors to adjust the wheel movement and restore balance. For instance, if the robot tilts forward, the wheels move backward, and if it tilts backward, the wheels move forward to counteract gravity. This continuous correction ensures stability even on uneven surfaces or under external disturbances. The system's responsiveness is crucial for maintaining dynamic equilibrium, similar to how humans adjust their posture when balancing on a bicycle. The motor driver generates the necessary torque for real-time balance correction by precisely controlling the speed and direction of the motors. Additional sensors, such as wheel encoders and distance sensors, assist in movement tracking and obstacle avoidance, allowing the robot to navigate its environment effectively while maintaining balance.

The robot operates on a 12V battery, supplying power to all components to ensure stable performance. The motors, typically stepper or DC motors, provide controlled and precise movements necessary for stabilization. The system is designed for real-time operation, with minimal computational delay to enable quick response times, which are essential for effective balance control. Furthermore, an HC-05 Bluetooth module enables external control via a smartphone or computer, allowing movement commands such as forward, backward, left, and right. This feature extends the robot's usability beyond autonomous balancing, enabling remote operation for research and practical applications. By continuously processing sensor feedback, executing motor adjustments, and incorporating advanced control techniques, the self-

balancing robot achieves stable operation, making it best for various applications, including personal mobility, automation, and robotics research.

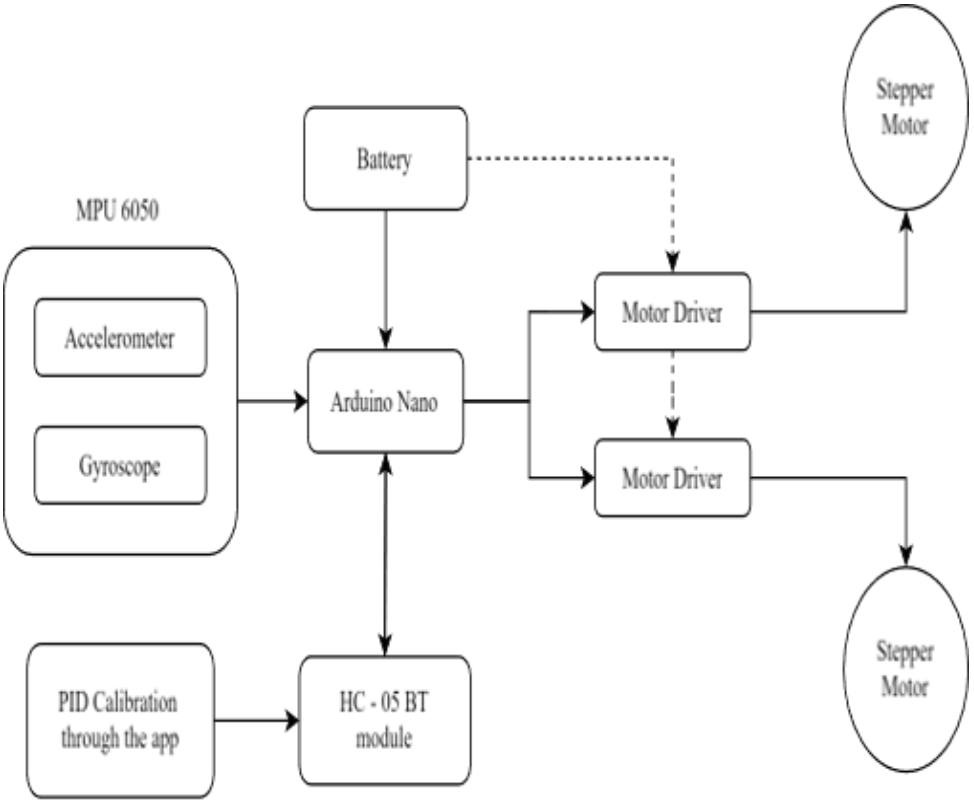

Figure 1: Block Diagram

METHODOLOGY

Figure 2 indicates the circuit of the self-balancing robot as indicated in the figure 4, which integrates several components to achieve precise control and stability. At the core of the system is the Arduino Nano, which acts as the main controller. It is powered by a LiPo battery rated at 11.7V and 1500mAh, which also supplies power to the A4988 stepper motor drivers through their VMOT pins. The Arduino Nano is connected to the MPU-6050 sensor via the I2C interface, using pins A4 (SDA) and A5 (SCL). This sensor provides crucial accelerometer and gyroscope data, allowing the Arduino to calculate the robot's orientation and maintain balance. The HC-05 Bluetooth module is connected to the Arduino Nano for wireless communication. It uses serial communication, with the TX out pin of the HC-05 connected to RX pin of Arduino, and the RX pin of HC-05 connected to TX pin of the Arduino. This configuration allows remote control and monitoring of the robot. Two A4988 stepper motors drivers are used to control the NEMA 17 stepper motors. Each driver receives STEP and DIR signals from the Arduino, which dictate the movement and direction of the motors. The EN, MS1, MS2, and MS3 pins on the drivers are used to enable the drivers and set the micro stepping mode, allowing for precise control of the motor steps. The motor coils are connected to the 1A, 1B, 2A, and 2B outputs on the A4988 drivers, ensuring proper motor operation. The use of 1K ohm resistors in the circuit helps limit the current to the drivers, protecting the components and ensuring stable operation. This comprehensive setup allows the Arduino to process real-time sensor data, adjust motor control for balance, and facilitate wireless communication, resulting in a responsive and stable self-balancing robot as shown in the figure 4.

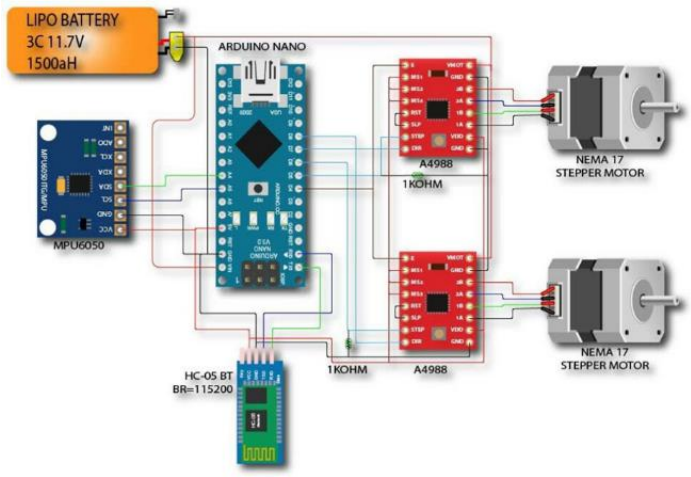

Figure 2: Circuit Diagram

The PID controller values are essential for fine-tuning the control system of a self-balancing robot, ensuring stability and responsiveness across different axes like roll, pitch, and yaw. Each axis is adjusted using proportional, integral, and derivative gains to achieve the desired balance and movement. These values are coupled with a virtual controller as shown in figure 3, which is provided by the EZ-GUI app, which allows for real-time adjustments and monitoring. The virtual controller offers an intuitive interface for users to work with the robot, enabling accurate control over its behavior. This integration enables smooth communication between the user and the robot, allowing for dynamic tuning and improved performance in various conditions indicated in the flowchart figure 5.

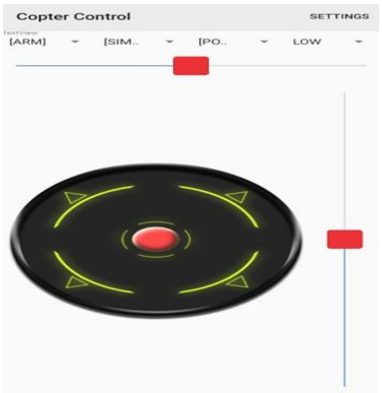

Figure 3: PID Controller

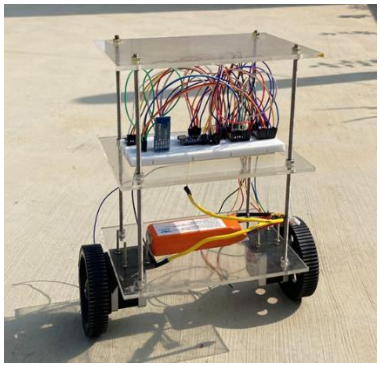

Figure 4: Self balancing robot model

A. Algorithm

1. Start the system.
2. Initialize accelerometer, gyroscope, and Bluetooth module.
3. Read real-time sensor data.
4. Calculate the tilt angle and determine the correction needed.
5. Adjust the motor speed and direction to maintain balance.
6. Continuously monitor stability and repeat the process.
7. Allow external Bluetooth control for movement.
8. Stop when powered off.

B. Flowchart

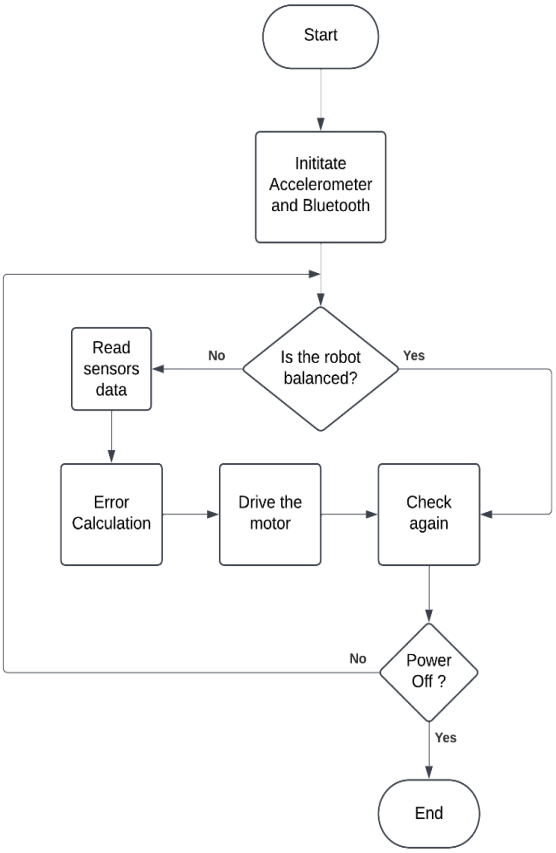

Figure 5: Flowchart

- **Step 1: Start** - The process begins when the robot is powered on, and the system initializes all necessary components.
- **Step 2: Initiate Accelerometer and Bluetooth** - The accelerometer is activated to monitor tilt and movement, while the Bluetooth module is initialized for remote communication if needed.
- **Step 3: Check if the Robot is Balanced** - The system evaluates whether the robot is in an upright position. If balanced, it skips correction; otherwise, it moves to sensor data collection.
- **Step 4: Read Sensor Data** - The accelerometer and motion sensors collect real-time data, including tilt angle and direction, to assess the robot's posture.
- **Step 5: Error Calculation** - The system calculates the difference between the current position and the ideal balanced state, determining the necessary corrective action.

- **Step 6: Drive the Motor** - Based on error calculations, the motors adjust speed and direction to counteract the tilt and maintain stability.
- **Step 7: Check Again if the Robot is Balanced** - The system reassesses the balance state. If unstable, it loops back to sensor data collection for further corrections.
- **Step 8: Power-Off Check** - The system verifies if a shutdown command is given or a predefined condition for shutdown is met.
- **Step 9: End** - The robot stops all motor functions and powers down until restarted.

ADVANTAGES AND DISADVANTAGES

A. Advantages

- Maintains self-balance, preventing falls.
- Automatically detects and corrects errors.
- Optimizes battery usage with controlled motor activation.
- Allows remote monitoring via Bluetooth.
- Improves stability for robotic applications.
- Reduces the need for human intervention.

B. Disadvantages

- Relies on sensors; failure affects balance.
- Continuous operation drains battery quickly.
- Struggles on rough or uneven surfaces.
- Requires precise calibration for proper functioning.
- Can be costly due to advanced components.
- Delayed response may occur in sudden movement

RESULT COMPARISON

Paper	Methodology	Proposed System
Harris and Adams (2006) [1]	Requirement Analysis & Planning, Design & System Architecture, Schematic Design & Circuit Building, Testing & Tuning	Implements real-time balance correction using sensor feedback and motor adjustments. Uses PID control for dynamic stability.
Li and Li (2007) [3]	Implementing a control algorithm (PID, LQR, MPC, fuzzy logic, or adaptive control) to correct errors and maintain balance.	Uses a PID controller for precise balance correction and stability. Ensures real-time adjustments to maintain equilibrium.
Zhang and Wu (2019) [6]	Self-balancing robots involve modeling dynamics, implementing control algorithms (PID, LQR, MPC, SMC), ensuring stability.	Uses sensors for real-time balance corrections to enhance adaptability and navigation.

Kim et al. (2023) [9]	Involves careful material selection, structural design, appropriate manufacturing techniques, and extensive testing.	Utilizes lightweight yet sturdy materials for durability. Focuses on maintaining stability through sensor feedback and motor control.

CONCLUSION

The development of a two-wheeled self-balancing robot demonstrates the effectiveness of real-time feedback control in maintaining dynamic stability. By utilizing an accelerometer and gyroscope for continuous monitoring and a PID control algorithm for precise motor adjustments, the robot successfully maintains balance while responding to external disturbances. The integration of Bluetooth connectivity further enhances its usability by enabling remote operation. Additionally, the ability to control the robot using the EZ-GUI app provides a convenient and user-friendly interface for manual control, parameter tuning, and real-time monitoring, making the system more accessible and versatile. Through an extensive review of existing methodologies, it is evident that various control techniques, including PID, LQR, and MPC, have been employed in similar projects. Our approach emphasizes a robust yet computationally efficient PID-based system, ensuring quick response times and effective balance correction. The use of lightweight materials and optimized hardware further improves performance, minimizing the energy consumption while trying to maintain stability.

This project serves as a base for the further advancements in autonomous self-balancing systems, with potential and capable applications in personal mobility, industrial automation, and robotics research. Future improvements could include sensor fusion techniques for enhanced accuracy, adaptive control for better environmental adaptability, and AI-based predictive adjustments to refine the balancing mechanism further.

REFERENCES

1. Harris and Adams focused on designing a two-wheeled self-balancing robot by incorporating a feedback control system "Development of a Two-Wheeled Self-Balancing Robot" (2006).
2. Borenstein and Koren research explored the use of gyroscopic sensors and accelerometers to measure the tilt and orientation of a robot in real-time. "The Influence of Gyroscopic Sensors on Robot Balance" (1991).
3. Li and Li Li's work examined different control algorithms for balancing robots, focusing on PID control and fuzzy logic control. "Control Algorithms for Two-Wheeled Robots" (2007).
4. Mori and Okada research introduced an adaptive control system that utilized Kalman filtering to estimate the state of the robot while filtering out sensor noise. "Adaptive Control for Self-Balancing Robots Using Kalman Filtering" (2013).
5. Wang , N. Hinsu and D. Suryavanshi, focused on optimizing self-balancing robots for energy efficiency. "Energy-Efficient Control for Self-Balancing Robots" (2015).
6. Zhang and Wu Wu proposed a trajectory tracking algorithm based on Model Predictive Control (MPC). "Trajectory Tracking for Self-Balancing Robots" (2019).
7. Chien and Lin Lin implemented reinforcement learning (RL) for improving the balance of self-balancing robots. "Reinforcement Learning for Adaptive Self-Balancing Robots". (2020).

8. Zhou and Xu " research extended the concept of reinforcement learning by incorporating deep learning models, particularly deep reinforcement learning (DRL), to handle more complex and unpredictable environments. Deep Reinforcement Learning for Self-Balancing Robots in Uneven Terrain"(2022).
9. Kim, S. Pandit and P. Sikka, explored the use of lightweight composite materials in the construction of wheeled self-balancing robots. "Lightweight Composite Materials for Robotic Frames"(2023).