Kadali Devi Sindhuja, 2025, 13:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Design and Implementation of Self Balancing Robot Using PID Controller

**Assistant Professor Anita M. Hanchinal, Mr. Harish R Savadatti,
Mr. Kevin S Anthony, Mr. Mahamed Yasin Baig M Savanur, Mr. Vinodkumar V Jaladi
Deptpartment of ECE, KLS VDIT, Haliyal India**

Abstract- Accomplishing stability in transportable mechanical technology, mainly in two- wheeled setups, is essential as it mirrors the factors of an changed pendulum framework. This contemplate offers the development of a cost- powerful selfbalancing robotic that makes use of a Proportional-Integral-Derivative (PID) controller along side an MPU6050 inertial estimation unit (IMU). The manage aspect leverages real-time sensor statistics to determine the robotic`s tilt and modify motor reactions in like way to hold upright modify. The look into lines the mechanical system, integration of sensors, flag sifting strategies, and PID tuning methodologies. Actualized using an Arduino microcontroller, L298N engine driver, and general DC engines, the version gives a common sense and affordable association capin a position of assisting vertical advent with negligible misKeywords— CAM, IOT, WIFI.

Keywords: Self-balancing robotic, PID controller, Arduino, Whirligig, Accelerometer.

I. INTRODUCTION

Self-Balancing Robots: Learning Control Systems the Hands- On Way- Self-balancing robots are inspired by the idea of an inverted pendulum—think of trying to keep a stick upright on your palm. The robot constantly adjusts itself to avoid falling over, using quick feedback whenever it begins to tilt. This makes it a great tool for learning how feedback-based control systems work in real life. To figure out its position, the robot uses an Inertial Measurement Unit (IMU), which combines a gyroscope and an accelerometer to track movement and orientation. This data is processed by a PID controller (short for Proportional- Integral-Derivative), which quickly calculates how to adjust the motors to keep the robot steady.

Working with this kind of robot gives learners valuable experience with how real systems behave—from understanding how a system moves, to learning how sensors and motors work together. Although balancing something like this is a nonlinear and complex problem, the PID controller offers a reliable and easy-to-implement solution, especially for beginners.

These robots also provide a chance to dive deeper into topics like filtering out sensor noise using techniques such as complementary or Kalman filters. Many self-balancing robot setups include wireless communication, battery management, and motor driver systems—offering a full view of how hardware and software are integrated.
Building and testing these robots also introduces learners to common real-world issues, such as motor lag, sensor drift, and delayed responses. Dealing with these challenges teaches how to build more reliable and robust systems.

In short, self-balancing robots are more than just cool projects—they're practical learning platforms.

They help connect theory to hands-on work and pave the way toward more advanced innovations like self-driving cars, robotic arms, or even humanoid robots. Because they're flexible and easy to work with, they're widely used in both education and research focused on intelligent control systems.

II. LITERATURE REVIEW

Recent work by Nada and Bayoumi showcased how Wi-Fi-enabled microcontrollers, like the ESP32, can bring wireless monitoring and control to self-balancing robots. By combining this compact hardware with a PID control algorithm, they were able to maintain system stability while enabling remote operation—a setup that's both efficient and practical for real-time applications.

In the realm of data-driven control, Homburger and colleagues explored how machine learning techniques could be used to model and control self-balancing vehicles. Their study compared several approaches and revealed important trade-offs in terms of accuracy, adaptability, and real-time performance, offering valuable guidance for anyone aiming to optimize robot control through data.

Meanwhile, Gandarilla and his team turned to artificial intelligence to boost control flexibility. They introduced a method that uses adaptive neural networks for trajectory tracking, allowing the robot to adjust on the fly to changes in its environment. This approach significantly improved performance in dynamic and unpredictable scenarios.

On the hardware side, Kim and Kwon focused on the design aspect of two-wheeled robots. They proposed a parametric design framework centred around co-axial drive systems, highlighting how thoughtful structural design, combined with tailored control strategies, can lead to better balance and smoother movement.

Traditional control methods still hold their ground. For instance, Das and his team built a self-balancing robot using a classic PID controller, demonstrating a straightforward, easy-to-implement solution that's perfect for mobile robots and beginner projects. In a similar spirit, Mahowik's Balancing Wii project has become a favourite in the DIY community. By using low-cost components and clever sensor fusion, it shows that building a functional self-balancing robot can be affordable and accessible to enthusiasts and hobbyists alike.

III. SYSTEM HARDWARE

Arduino Nano (ATmega328P)

The Arduino Nano is a tiny yet capable microcontroller board built around the ATmega328P chip. It offers 14 digital pins and 8 analog inputs, providing plenty of flexibility for different kinds of embedded projects. Its small form factor and integrated USB port make it easy to connect and program, which is why it's a favorite for robotics, sensor integration, and various DIY electronics tasks. The board is known for being energy-efficient and compatible with a wide range of hardware components, making it an excellent choice for both beginners and experienced developers.

Figure1: Arduino Nano (Atmega32p)

MPU-6050

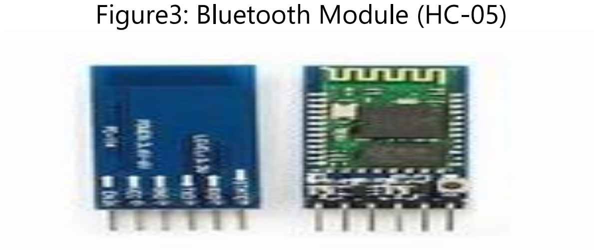
The MPU-6050 is a compact motion sensor used to determine how the robot is tilted or positioned. It plays a key role in helping the system stay balanced by continuously monitoring the robot's orientation. This sensor combines a 3-axis accelerometer and a 3- axis gyroscope on the same chip, both built using MEMS (Micro-Electro-Mechanical Systems) technology. It also includes an internal temperature sensor. Capable of taking hundreds of measurements per second, the MPU-6050 tracks motion along the x, y, and z axes in real-time. It communicates with the Arduino through the I^2C protocol, allowing smooth and efficient data transfer. To wirelessly adjust the robot's PID control settings, an HC-05 Bluetooth module is used. This module allows communication between the robot and an Android smartphone. Known for its simplicity and reliability, the HC-05 features six pins and is widely used in Bluetoothbased control systems.

Figure2: The MPU-6050

Bluetooth module (HC-05)

The HC-05 is a reliable and user-friendly Bluetooth module that enables seamless wireless communication between electronic devices. Whether you're connecting a microcontroller to a smartphone, a PC, or another embedded system, this module provides a simple and effective solution. It supports the Serial Port Protocol (SPP) and can function as either a master or a slave device, giving it the flexibility to fit into a wide range of project requirements. One of the key advantages of the HC-05 is its easy integration through a UART (Universal Asynchronous Receiver-Transmitter) interface, which makes it especially accessible for beginners and hobbyists. Its consistent performance and wide compatibility make it an ideal choice for robotics, home automation, and Internet of Things (IoT) systems. Additionally, the module's six-pin configuration and clear command structure allow for smooth configuration and communication setup. Because of its low cost, stability, and ease of use, the HC-05 remains a go-to option for developers working on wireless control or data transfer applications.

Figure3: Bluetooth Module (HC-05)



Stepper Motor

Stepper motors serve as the driving force behind the robot's movement, functioning much like its legs. What sets them apart from standard geared motors is their ability to move in small, precise steps, allowing for better control and improved stability. Inside the motor, several coils are arranged in a specific sequence. When electrical current is applied to these coils in a particular order, the motor rotates in fixed increments, or steps. This step-by-step motion makes it possible to control both position and speed with high accuracy. By carefully managing the flow of current to the coils—especially when guided by a microcontroller—stepper motors can produce smooth and controlled movements. This level of precision makes them a great fit for robotics and other applications where exact positioning is critical.

Figure4: The stepper motor

Stepper Motor Driver (A4988)

The A4988 is a small but powerful driver used to control bipolar stepper motors with high precision. It allows fine-tuned motor movement through adjustable current control and supports several stepping modes—ranging from full steps to very fine sixteenth steps. With built-in thermal protection, the driver can operate safely and reliably, even under demanding conditions.

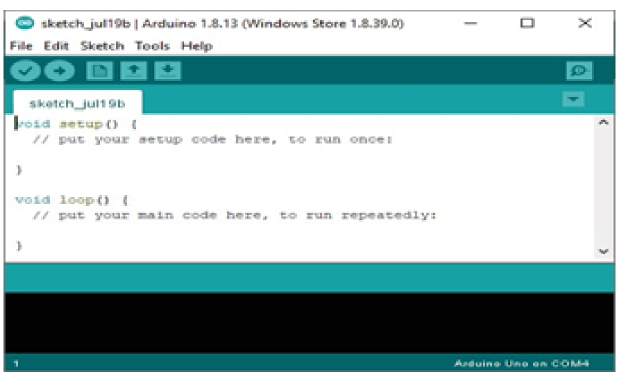
Figure5: A4988 Stepper motor driver module.

IV. SOFTWARE SPECIATION

Arduino IDE

The Arduino Integrated Development Environment (IDE) is a freely available software platform that simplifies programming for Arduino microcontrollers. Designed to run across different operating systems, it combines a straightforward text editor for writing code with a built-in compiler that prepares the code for uploading to Arduino boards [13]. Programs created within this environment, commonly known as "sketches," are written in a simplified version of C/C++. Once uploaded, the microcontroller interprets these instructions to carry out specific tasks. Thanks to its clean interface and beginner-friendly design, the Arduino IDE has become a popular tool in both educational settings and rapid prototyping, helping users bridge the gap between software and hardware development.

Figure6: A default Arduino IDE window.



EZ-GUI Ground Station.

EZ-GUI Ground Station is a dedicated Android app that helps users interact with flight controllers in drones and robotic platforms. Through Bluetooth or Wi-Fi, it allows for wireless communication, enabling users to view sensor outputs, adjust system configurations, and control the device in real time.

One of its standout features is live telemetry, which provides continuous data feedback. The app also supports PID tuning and GPS-based tracking, making it useful not only for basic control but also for performance optimization and remote troubleshooting. Its ease of use and wide range of functions make it a practical tool for both enthusiasts and engineers working with autonomous systems.

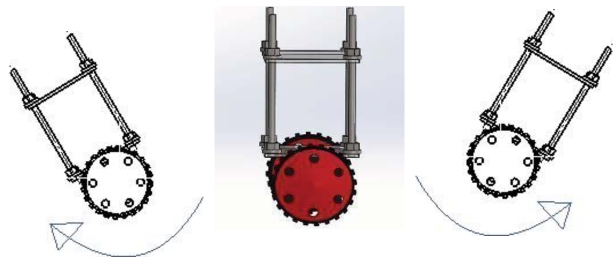
Figure7: EZ-GUI Ground Station

PID Controller.

The self-balancing robot operates using a closed-loop control system, depicted in the figure. Its current position is continuously tracked by the MPU-6050 sensor, which sends real-time feedback. This feedback is compared against the target position—the robot standing upright at a 90° angle relative to the ground. The difference between the actual and desired positions creates the tilt angle, which represents the system's error.
To keep the robot balanced, this error must be corrected efficiently and accurately.

To handle this, a PID controller (Proportional-Integral- Derivative) is utilized. The target angles and PID tuning parameters are transmitted to the Arduino via Bluetooth using a mobile app. The proportional gain, Kp, controls how aggressively the robot responds to the current error, influencing the speed of correction and the balance point. The integral gain, Ki, works to eliminate any lingering errors and smoothens the robot's movements, with even small adjustments having noticeable effects. The derivative gain, Kd, helps the system react to how quickly the error is changing, preventing overshoot—but setting it too high can destabilize the robot.

Figure8: Self-balancing positions of robot.



(a) (b) (c)

(a) Mean position of robot, (b) correction of the robot when tilted at an angle in anticlockwise side, and (c) correction of the robot when tilted at angle in clockwise side.

V. SYSTEM ARCHITECTURE

The surveillance robot is centered around an Arduino Nano board, which acts as the main controller coordinating all its functions. It is equipped with several sensors that gather important environmental information, enhancing the robot's ability to respond and adapt to different situations effectively. Users can control the robot remotely via Bluetooth, using either an Android smartphone or a laptop running the EZ-GUI Ground Station app. The robot's movements rely heavily on real-time video streaming from a camera mounted onboard, allowing the operator to navigate it precisely based on what they see. The complete system's connections and components are clearly laid out in a comprehensive interfacing diagram.

Figure9: Block diagram of the robot

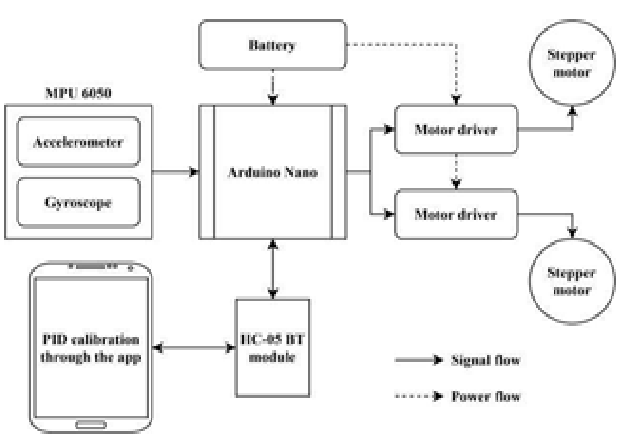
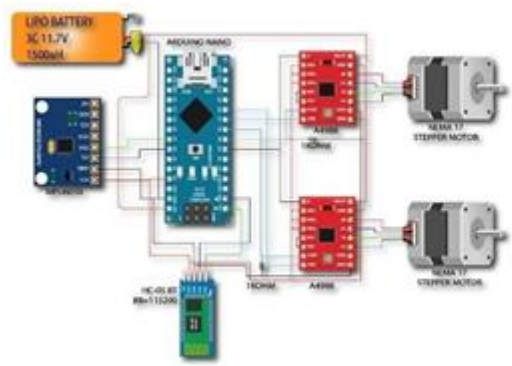


Figure10: Overall Interfacing diagram of the robot



Step by step assembly instruction:

- Gather Parts: Collect Arduino Nano, MPU-6050, 2 stepper motors with A4988 drivers, HC-05 Bluetooth, LD1117-3.3V regulator, 100nF & 1µF capacitors, and a battery.
- Set Up Power Lines :Create positive and ground rails. Connect Arduino's Vin to the power rail and GND to ground.
- Wire MPU-6050 : Power with 3.3V from LD1117 + 1µF cap. Connect GND to ground, SCL to A5, SDA to A4.
- Connect Stepper Motor 1 : A4988 outputs to motor (D, B, C, A). DIR → D2, STEP → D3, V+ to power, GND to ground.
- Connect Stepper Motor 2: Same as above, but DIR → D4, STEP → D5.
- Set Up HC-05 : GND to ground, TX → D10, RX → D11, VCC to 3.3V from LD1117 + 100nF cap.
- Install Voltage Regulator: Vin from Arduino 5V, Vout powers MPU-6050 and HC-05, GND to ground.
- Attach Capacitors: Connect both capacitor grounds to the ground rail for stable voltage.
- Upload Code : Plug Arduino into your PC via USB and upload your program.
- Connect Battery : Battery + to power rail, – to ground rail to power the system.
- Test & Tune: Check sensor, motors, and Bluetooth. Adjust PID through the app and retest.

Result

The self-balancing robot was developed using an Arduino Nano, paired with an MPU-6050 sensor, stepper motors driven by A4988 modules, and a

PID controller to maintain stability. The MPU-6050 continuously tracked the robot's tilt angle and sent this information to the Arduino, where the PID algorithm processed it to keep the robot upright. By fine-tuning the PID parameters, the robot was able to hold an upright position near 90°, swiftly correcting small disturbances with minimal shaking. Additionally, the inclusion of the HC-05 Bluetooth module enabled wireless communication, allowing users to remotely adjust the PID settings through a smartphone app. Experimental tests confirmed that the robot could stabilize itself within 1 to 2 seconds after tilting and maintain its position within a narrow range of ±3°, highlighting its effective and responsive control system.

VII. Conclusion

To conclude, this project effectively showcased the use of a PID controller in maintaining the balance of a two-wheeled robot. The robot's stability was achieved through the seamless integration of accurate sensor readings, timely control actions, and precise motor responses. The inclusion of wireless tuning through a mobile interface added convenience and made real- time adjustments easier for the user. This implementation offers a strong starting point for future upgrades, such as dynamic PID tuning, improved performance on uneven terrain, and autonomous operation using additional sensors and advanced control techniques.

REFERENCES

1. Nada and M. Bayoumi explored how Wi-Fi connectivity can be integrated into a self-balancing robot using the ESP32 microcontroller and PID control techniques in their 2024 research.
2. H. Homburger and colleagues (2023) compared various data-driven modeling methods for identifying and controlling a self-balancing vehicle, offering insight into performance trade-offs and effectiveness.
3. I. Gandarilla et al. (2022) developed a control approach using adaptive neural networks to ensure accurate trajectory tracking for a selfbalancing robot, highlighting the power of AI in dynamic control systems.
4. Y. Kim and S. Kwon (2022) presented a parametric design approach for two-wheeled robots, focusing on co-axial drive mechanisms and stability enhancement via customized control.
5. A. Das and team (2021) designed a self-balancing robot that utilizes a PID controller, offering a straightforward application of classical control theory for mobile robot stabilization.
6. Mahowik's open-source project "BalancingWii" (2020) on GitHub provides practical resources for building a self- balancing robot using affordable hardware and sensor integration.
7. Arduino (2020) offers access to its official software tools used in developing embedded systems, which play a foundational role in hobbyist and educational robotics.
8. Q. Yu, T. Sun, and Y. Zhao (2019) built a two-wheeled robot using PID and complementary filters to enhance stability and filtering of noisy sensor data, demonstrating effective motion control.