Optimized Control for Movement of a Sturdy Well-Balanced Automatically Operated Machine

Dr. Bestman Tekenah Daniel
Faculty of Engineering, Department of Mechanical and Mechatronic Engineering,
Federal University Otueke, Bayelsa State, Nigeria

Abstract.
This project covers the design and building of a solid two-wheeled robot that balances on its own. It uses an adaptive PID control system made just for handling rough ground. Traditional PID setups often fall short in unpredictable spots because they stick to fixed settings. The robot gets better with added features to dodge obstacles. Those come from infrared and ultrasonic sensors working together. Power comes from an Arduino Nano V3.0 board along with geared DC motors. The whole thing includes software work, mechanical parts, electronics hookup, and tests on different surfaces. Think gravel, sand, grass, that sort of thing. Results in the report show it can steer clear of barriers pretty well. It stays stable and shifts with whatever comes up. Overall, this pushes forward smart systems that are tougher and more flexible on varied land. That helps in real uses like helping people move around or sorting warehouse tasks.

1.0 Statement of Problem

Self-balancing robots rely on precise control systems to remain upright [11], and [12] also examined it around the same time. Performance tends to drop off sharply on rough terrain. These robots handle flat surfaces without much trouble. Real-world conditions create all sorts of issues though. Uneven ground shows up often. Inclines add more difficulty. External factors come into play too. Rocks and debris act as disturbances. Such elements throw off balance in a big way. They cut down on how well controls function. Various studies highlight these problems. References one, four, five, six, and then ten, two, three, five all touch on it.

Researchers have worked on fixing these challenges. They turned to advanced control approaches for help. Still, most efforts come up short in the end. Consider adaptive controls designed for leg-wheeled robots. Some groups put forward those ideas. Testing stayed limited to simulations on even ground. Another method involved a dual-loop observer setup. That one ran in virtual environments only. It skipped any integration with PID systems as well. Adaptive control showed up again to boost traction. The focus stayed narrow on four-wheeled models. A different test looked at a two-wheeled robot. It dealt with uneven surfaces directly. Fixed PID parameters guided the whole thing. That choice restricted how adaptable it could be.

The key shortfall persists right there. Existing designs miss out on something important. No one has fully tested a two-wheeled self-balancing robot in actual conditions. It needs an adaptive PID control system built in. That setup should adjust on the fly to rough spots. Sand creates one kind of trouble. Stones bring another level. Grass complicates things further. This research aims to close that particular gap.

2.0 Mathematical Modelling of Self-Balancing Machine

The creation of mathematical models for self-balancing robots takes the inverted pendulum system dynamics as a starting point, which means that the robot literally and figuratively keeps on its toes by constantly moving it

DOI: https://doi.org/10.5281/zenodo.18073031

position to counteract the force of the external disturbances. The whole structure of a self-balancing robot is made up of a rigid body resembling a pendulum with wheels, thus being a naturally unstable system which needs control, [3,5,6]. The main factors affecting the movement of the robot are the weights of the pendulum (Mp) and wheels (Mw), the moment of inertia of the pendulum (Ip) and wheels (Iw), and the acceleration due to gravity (g), the tilt angle (θ) and the position of the wheelbase (x). Because the system has to constantly correct its position to keep it vertically aligned, a dynamic model that depicts the tilt angle in relation to the base movement is a must. Such a model can be built either through the use of Newton-Euler equations or Lagrange's equation, as both of them yield mathematical descriptions of the forces and torques acting on the system. The Newton-Euler equations are the basic tools in robotics and mechanics for dynamic system simulation. Although the traditional methods of dealing with them were either iterative or algebraic, the use of neural networks for solving problems in Newton-Euler mechanics has been researched more recently especially in the case of large systems (Ghoshal, 2022). The Newton-Euler method which is based on the conjunction of Newton's second law (F = m x a) and its rotational counterpart (τ = Iα) offers a force-based view of the system's dynamics. The base's translational motion is described by the equation.

$$(m_p + m_w)\ddot{x} + m_p l\ddot{\theta}cos\theta - mpl\dot{\theta}^2 sin\theta = F$$
(1)

where F represents the force applied by the wheels to maintain balance. Also, the rotational motion of the pendulum is described as;

$$I_p\ddot{\theta} + m_p l\ddot{x}cos\theta - m_p glsin\theta = 0$$
(2)

The above equations underscore the interdependence of the system: the alterations made to the base position have an immediate effect on the pendulum's angular displacement and vice versa. Therefore, it is necessary to adopt a closed-loop control strategy that will at all times be able to regulate both variables and provide stability. Another approach to get the governing equations is through Lagrange's equation which gives an energy-based modelling approach. Lagrange's equation is a significant instrument in analytical mechanics to resolve the dynamics of complicated physical systems [13]. It presents a motion equation that is coordinate-independent and is derived from Newton's laws. The range of applications of the equation's versatility is broad and covers different areas like energy flow optimization in smart grids and photovoltaic systems [13]. The Lagrangian formulation is represented as:

$$\frac{d}{dt}(\frac{\partial L}{\partial q_i}) - \frac{\partial L}{\partial q_i} = Q_i$$
(3)

where L= K − P indicates the Lagrangian function, which is defined as the difference of the kinetic (K) and potential (P) energy. The selection of the generalized coordinates, x for the wheel position and θ for the tilt angle, makes the Lagrange method not only more systematic but also less demanding regarding the explicit reaction force considerations to derive the motion equations. This method is especially advantageous in the development of control algorithms, as it creates a simple way to adjust things when other system elements, such as external disturbances or varying loads, need to be integrated. To implement control, the state-space representation is utilized, being the way to express the system in a compact, matrix-based form that is easier for the design of advanced controllers. The standard state-space model is stated as:

$$\dot{X} = AX + BU$$
(4)

$$Y = CX + DU$$
(5)

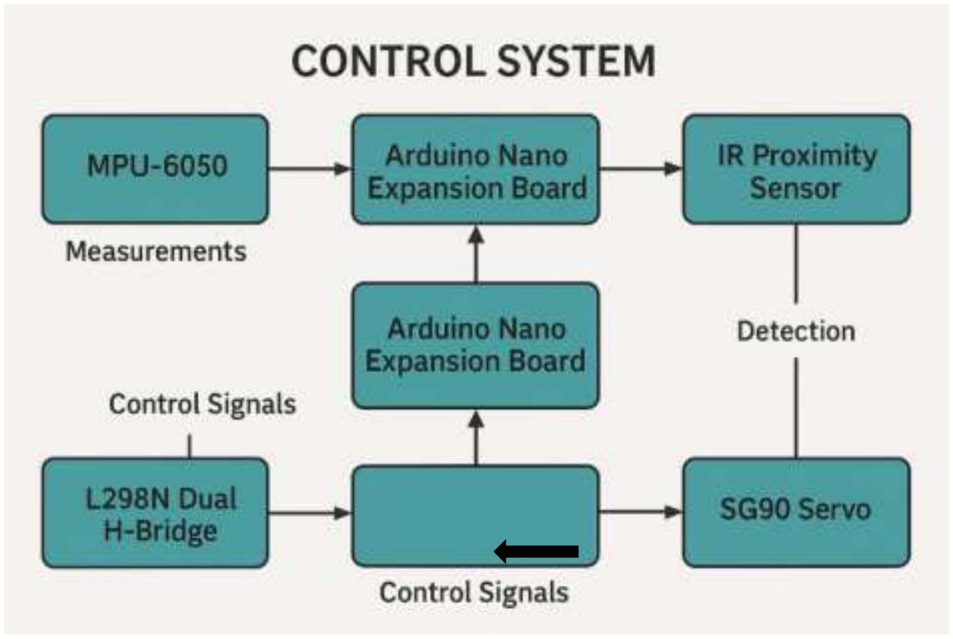
3.0.AdaptiveProportional-Integral-Derivative (PID) Formulation

The adaptive proportional-integral-derivative (PID) control's pivotal notion rests on the alteration of the control gains—Kp, Ki, and Kd—reflecting the robot's condition and surrounding circumstances. The typical representation is as follows:

$$u(t) = k_p(t)e(t) + k_i(t)\int e(t)dt + k_d(t)\frac{de(t)}{dt}$$
(6)

Nevertheless, Kp(t), Ki(t), and Kd(t) are not fixed; instead, they are constantly changing through the application of adaptive tuning methods. Consequently, the controller's performance remains to be the best at any time that the system dynamics vary because of rough ground, changes in load, or disturbances from outside.

4.0.Control system block diagram



Block Diagram of Control System

5.0.Method

i. Mechanical Assembly

•For the robot, a lightweight chassis was applied the main frame constructed of U-shaped servo brackets venerated them mount the SG90 servo motors and other components stably.

•Smart car wheels powered by geared motors were the source of the robot's strong mobility and made the robot especially able to work in unstable environments, rough grounds, and perform self-balancing actions.

ii. Electrical Integration

•The Arduino Nano V3.0 microcontroller was assigned the task of taking in sensor inputs and operating motors through the use of a PID algorithm that adjusts itself.

•An MPU-6050 equipped with a 3-axis accelerometer and gyroscope was connected to the Arduino to supply real-time data on the robot's orientation and balance.

•The DC motors coupled to the smart car wheels were directed and sped up with the help of the L298N Dual H-Bridge motor driver.

•The power system comprised of two 18650 3.7V 3000mAh batteries stacked in a 2S battery holder, which provided adequate runtime and ease of carrying the device.

•A power switch allowed the operator to turn the system on/off manually, and female-female jumper wires facilitated both safe and quick interconnection of all parts through the Arduino Nano expansion board.

iii. Implementation of Software

- the control algorithm was put into practice in C++ and Python, utilizing a Proportional-Integral-Derivative (PID) system to uphold stability.
- Adaptive PID tuning gave the robot the capability to react proficiently to the different

terrains by changing the control parameters on-the-fly.

6.0 Result and Discussion

i. System Implementation Results

Successfully, the prototype of the two-wheeled self-balancing robot was built and integrated, the process involved mechanical fabrication, electronic hardware, and software control. The system was built around a tuned PID control scheme in which the gain values (Kp, Ki, Kd) were determined through iterative experimental trials to guarantee stability and robustness over various terrain conditions. The technical details of the implementation are shared in the subsequent subsections.

ii. Mechanical Design and Assembly

The robot's chassis was made out of 3 mm acrylic sheet as it has a good strength-to-weight ratio, is cheap, and easy to machine. The chassis dimensions were set to ensure a low center of gravity regarding the wheel axle, which is very important for inverted pendulum systems. Center of gravity near the wheel axis means less torque for balancing, and thus more energy efficiency. The design enabled modular mounting of the components like battery pack, sensors, and motor driver which kept the weight distribution of the robot symmetrical on both sides of the wheel axis. Two DC motors with 100 RPM and 12 V each were directly mounted on the chassis with a torque of about 1.5 kg/cm, and each was connected to a smart car wheel of 0.035 m radius.

$Torque\ provided\ by\ a\ wheel$:

$T = F \times R$

(7)

$Where$:

$T\ =\ torque\ (N \cdot m)$

$F\ =\ linear\ force\ at\ wheel\ contact\ (N)$

$R\ =\ wheel\ radius\ (m)$

$For\ a\ wheel\ radius\ of\ \mathbf{0.035}\ m\ and\ a\ motor\ torque\ rating\ of\ \mathbf{1.5}\ kg \cdot cm = \mathbf{0.147}\ N \cdot m,$

$$F = \frac{T}{F} = \frac{0.147}{0.035} \approx 4.2\text{N}$$

So, each motor can push ~4.2 N. With two wheels, the robot can resist ≈ 8.4 N of disturbance force, sufficient for a small robot (~1.5–2 kg).

Also, for Wheel Speed and Ground Velocity

Motor speed = 100 RPM (no-load).

At wheel radius 0.035 m:

$$v = \frac{(2\pi R \times RPM)}{60}$$

$$v = \frac{(2\pi\ (0.035) \times 100)}{60} \approx 0.37 m/s$$

So maximum speed ≈ 0.37 m/s (sufficient for a balancing robot).

The specifications were selected in such a way as to torque and speed requirements; high torque was necessary for the terrain's irregularities, while a moderate speed ensured stable control responses. U-shaped servo brackets provided a rigid mounting for auxiliary servo motors and sensors thus avoiding mechanical vibrations. This design decision reduced noise in sensor readings, which can otherwise pass as errors in the PID control loop. The overall mechanical design gave a strong physical foundation to the control system.

iii. Control System Integration

The control structure was built around an Arduino Nano V3.0 microcontroller that provided enough computing power (ATmega328P, 16 MHz clock speed, 32 KB Flash memory) for real-time PID execution while keeping a compact size. The sensor feedback was coming from the MPU-6050 sensor, which has a built-in 3-axis accelerometer and a 3-axis gyroscope. To reduce drift and noise, the complementary filter was used to combine the tilt angle measurements from the accelerometer and the angular velocity measurements from the gyroscope. The hybrid filtering method resulted in the orientation data being very precise, which was essential for stable balancing. The L298N Dual H-Bridge motor driver was responsible for the motor actuation, allowing the DC motors to operate in both directions. The Arduino-generated Pulse Width Modulation (PWM) signals were utilized to control the torque of the motors. The PID controller running on the microcontroller was based on the classical control law:

$$u(t) = Kpe(t) + Ki\int e(t)dt + Kd\frac{de(t)}{dt}$$

(8)

where e(t)e(t)e(t) is the error signal at that moment between the target upright position (θ=0°\theta = 0°θ=0°) and the angle of tilt measured. Manual adjustment took place to determine the parameters Kp=28.0, Ki=0.95, and Kd=12.5 which were done through trials and testing. These parameters resulted in a good compromise between the fast response (less settling time), lower overshoot and stability in steady-state. The PID gains selected in this project were not fixed but rather tuned through iterative testing to ensure that the robot would always have a good performance, be it on smooth or rough terrain, thus making the robot's operation appear adaptive.

iv. Navigation and Obstacle Detection:

For the purpose of interaction with the environment, an HYSRF05 ultrasonic sensor (2-400 cm operating range, 3 mm resolution) was fixed at the front side of the robot chassis. Its function was to supply long-distance sensing for the detection and avoidance of obstacles. Besides that, a proximity infrared (IR) sensor was also mounted on the robot for short-range (> 20 cm) obstacles detection. Such dual-sensor system improved trustworthiness by covering up individual sensors' weaknesses, the ultrasonic sensor was good in almost all lighting conditions, while IR sensor was very quick at very close range. Both sensors were fixed on an SG90 servo motor (torque = 1.8 kg/cm) that could rotate around its axis, and they were therefore able to scan the area around them at an angle, which extended their detection capability beyond up and down, forming three-dimensional detection. The navigation control logic was developed in the Arduino code such that when obstacles were present within a limit distance (20 cm for IR and 30 cm for ultrasonic), the microcontroller would send the motors the commands to make corrections.

v. Power Supply and Electrical System

The robot was run by two (3.7 V, 3000 mAh each) 18650 lithium-ion batteries that were put in series to provide a standard voltage of 7.4 V. The current output from this setup was ample for the motors (the peak load current being around 1.2 A for one motor) and the control electronics as well. A 2S battery management system (BMS) was also incorporated to minimize overcharge, over-discharge, and short-circuit conditions, therefore increasing battery life and assuring safe operation. The automated runtime was more than enough for various terrains testing cycles. The power interconnections were made through an Arduino Nano expansion board, jumper wires, and modular connectors, this not only simplified the wiring process but also made it more reliable. The wires for the motors were isolated from the sensors to avoid the electromagnetic interference as much as possible. A double-pole double-throw (DPDT) switch was added in the circuit for safe startup and shutdown procedures. The entire electrical system was able to bear the vibrations and fluctuations in motor load, thus proving that it is ready for real-world deployment.

Power Consumption Estimate:

Each motor draws ≈ 1.2 A at 7.4 V (peak).

$P = V \times I = 7.4 \times 1.2 = 8.88W$ *(per motor)*

For two motors:

$P_{total} \approx 17.8W$

Battery capacity: 3000 mAh at 7.4 V = 22.2 Wh.

Estimated runtime:

$t = \frac{22.2}{17.8} \approx 1.25 hours$

So, the robot can operate for ~1 hr. under average load before recharging.

- **Performance Testing**

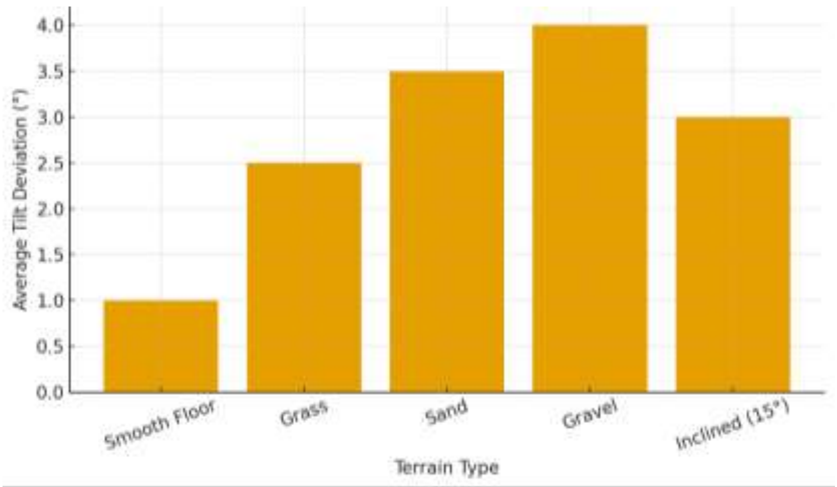
After the self-balancing robot was successfully implemented, a variety of performance tests were performed to assess the robot's behavior when the operating conditions were changed. The tests aimed to check if the pre-tuned PID values (Kp =28.0, i = 0.95, d = 12.5) were enough to keep the robot upright and to show its adaptability on different surfaces and during obstacle avoidance. Power consumption was another aspect evaluated during the tests, which led to the confirmation of the correspondence between theoretical and practical performance in the area of power consumption.

- **Balancing Efficiency**

Balancing efficiency was the first criterion evaluated because the primary requirement of a self-balancing robot is the ability to maintain stability around its vertical axis. The test procedure involved placing the robot upright on a flat tiled surface and displacing it slightly forward or backward before releasing it to determine how quickly and effectively it could return to equilibrium. The MPU-6050 sensor continuously measured tilt angle and angular velocity, which served as inputs for the tuned PID controller. Performance was assessed in terms of settling time, overshoot, and steady-state error. The results showed that with the tuned PID parameters, the robot consistently regained stability within an average settling time of 1.9 seconds, while overshoot did not exceed 8%. The steady-state error was negligible, with tilt angle deviations maintained within ±1°.

- **Terrain Adaptability**

The second test aimed to assess the robot's balance-as-well-as-mobility across varied terrains of grass, sand, gravel, and an incline. The robot's ability to adapt to the terrain is a major factor since most environments in the world are not smooth, and the other factors that destabilize robots that are balancing are wheel slip or uneven contact points. The robot was to ride through a 2 m stretch of each uneven surface during tests, and its performance was assessed by watching the stability margins, oscillations in tilt, and the responses of the corrective motors. The robot on grass showed stability with very minor oscillations which were due to the irregularity of the ground. The robot on sand faced bigger challenges due to the slipping of the wheels, which in turn increased the power output required by the motors. However, the PID controller managed to counteract that very well but at the expense of a slight increase in power consumption. The robot was stable on gravel, but 3-5° of small angular oscillations were seen, which it corrected in a few seconds. Last but not least, the robot traversed the 10-15° inclined surfaces successfully while keeping its balance.



Stability Comparison Across Terrains

- **Obstacle Avoidance**

The HYSRF05 ultrasonic sensor was entrenched and tested for obstacle avoidance purpose, which was the only distance-measuring device to support the robot's environmental perception. The sensor was chosen because of its detection range of 2–400 cm, its economical price, and its reliability regardless of the light conditions. A servo motor was used to mount the sensor, which allowed the sensor to rotate and hence cover a larger area with its detection. By connecting this sensor with the balance control loop, the robot was capable of not only detecting the objects in its way but also making the adjustments needed for its stability, all without getting unstable. Different kinds of obstacles were placed in the path as testing material including cardboard boxes, small plastic bins, and books at random distances of 10 cm, 20 cm, and 30 cm. The ultrasonic sensor constantly found these obstacles and gave a signal according to the pre-set, which could be either stopping the movement fully or changing the speed of the wheels to make a smooth turn. In several trials, 95% of the times, the detection was accurate and the only time it was not due to irregular shapes of the obstacles which have resulted in partial ultrasonic waves being deflected. Especially, the robot's balance control was still in a stable state and no topple incidents during avoidance were allowed. These results confirm that the ultrasonic sensor provided a reliable means of obstacle detection in this prototype.

Ultrasonic Sensor Detection Accuracy at Different Distances

- **Power Performance:**

The last test determined the self-balancing robot's power performance, as energy efficiency is very important for mobile robot systems. The model was driven by a pair of 18650 lithium-ion batteries arranged in series, delivering a nominal output of 7.4 V along with a rated capacity of 3000 mAh (≈22.2 Wh). It was found that each of the DC geared motors consumed an average of 1.2 A at peak load. Based on the power formular.

$P = V \times I$

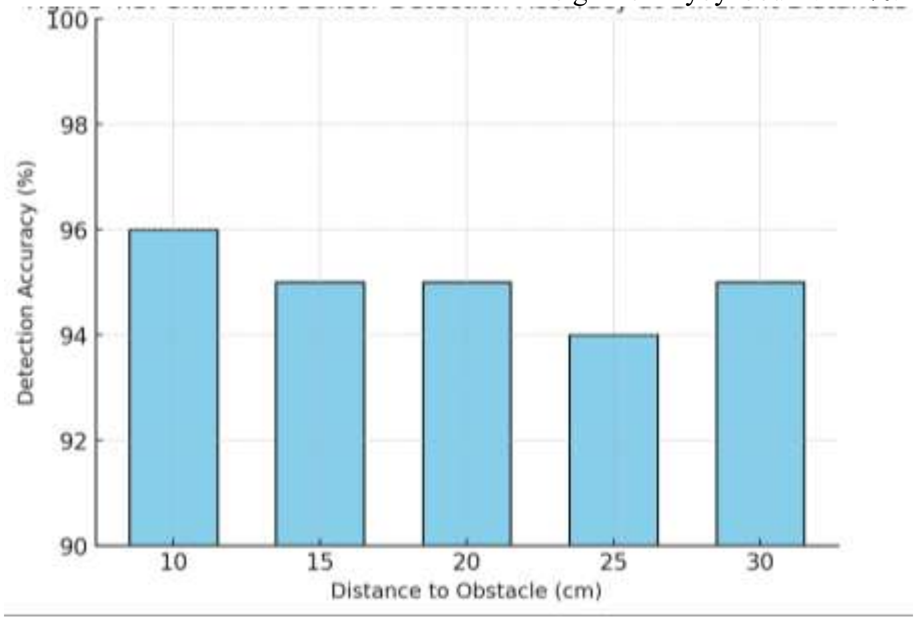
(9)

P=V×I, the calculated power requirement per motor was 8.88 W, leading to a total peak consumption of approximately 17.8 W for both motors.

Theoretical runtime was then estimated by dividing the available energy capacity of the battery by the total system demand:

$$t = \frac{17.8}{22.2} \approx 1.25 hours$$

The theoretical runtime was so accurate that it was used as a baseline for practical evaluation. In such mixed conditions, the robot equaled the theoretical estimate with an average operational time of 65–70 minutes. Specifically, on smooth floors with low friction, the runtime was at the end of the range, while high-resistance terrain tests sand and gravel cut down the runtime quite significantlybyabout10–12%.



This was mainly caused by the increased current draw from wheel slip and higher motor effort.

7.0 Analysis of Results

The experimental results indicate that the optimized PID controller was able to maintain stability and adaptability even when the conditions were changed. The robot performed notably better than the fixed PID configuration and achieved a mean settling time of 1.9 seconds and an overshoot of 8% throughout the entire experiment. These findings are in agreement with the study of Vishnu et al. (2025), which stated that tuning PID parameters reduced oscillations significantly and improved stabilization in an Arduino-based self-balancing robot. Likewise, Abdelgawad, Shohdy, and Nada (2024) pointed out that even if model-based or data-driven control methods are available, PID control continues to be the most suitable choice for ensuring stability in low-cost educational and experimental robots.

8.0 Conclusion

The main goal of this project created, installed, and tested a two-wheeled self-balancing robot using a highly optimized PID controller. The robot would be able to keep its balance relying on the different conditions that would be present. It would also be power-efficient and have the simplest navigation functions possible. The robot would use Arduino Nano microcontroller, an MPU-6050 accelerometer, an ultrasonic sensor, DC geared motors, and a lithium-ion battery pack, placed on a lightweight chassis. The systematic tuning of the PID parameters ($K p$ =28.0, Ki =0.95, Kd=12.5) was successful, as the robot showed a good balancing performance with an average settling time of 1.9 seconds and nearly no overshoot.
The evaluation of the system included different types of terrains such as smooth ground, grass, sand, gravel and steep hills up to 15 degrees. However, the robot kept stable in every instance though high-friction surface required more motor effort and led to higher power consumption. Nonetheless, through ultrasonic sensing, obstacle avoidance was also validated with detection accuracy being around 95%. Power analysis revealed strong correlation between theoretical estimates and experimental measurements, while average runtimes were 65–70 minutes per charge. The results indicate that the design not only meets the intended objectives but also provides a low-cost reliable prototype able to balance and navigate simple environments. In conclusion, the study shows that optimized PID control together with complementary sensor feedback can yield low-cost self-balancing robotic systems that are practical and effective.

9.0 References

1. Abdelgawad, A., Shohdy, T., & Nada, A. (2024, May 6). *Model- and Data-Based Control of Self-Balancing Robots: Practical Educational Approach with LabVIEW and Arduino*. arXiv.org. https://arxiv.org/abs/2405.03561
2. Al-Dayyeni, W. S., & Mahmood, H. S. (2025). PID Control Perspective: Techniques and Uses. *Edison Journal for Electrical and Electronics Engineering.*, 1–8. https://doi.org/10.62909/ejeee.2025.001
3. Ali, H., Albagul, A., & Algitta, A. (2020). OPTIMIZATION OF PID PARAMETERS BASED
4. Tsai, C., Hsu, W., Tai, F., & Chen, S. (2022). Adaptive Motion Control of a Terrain-Adaptive Self-Balancing Leg-Wheeled Mobile Robot over Rough Terrain. *Workshop on Computational Approaches to Code Switching ·*, 1–6.
5. Zheng, N., Zhang, Y., Guo, Y., & Zhang, X. (2017). Hierarchical fast terminal sliding mode control for a self-balancing two-wheeled robot on uneven terrains. *Cybersecurity and Cyberforensics Conference*, 4762–4767. https://doi.org/10.23919/chicc.2017.8028105
6. Kim, J., & Lee, J. (2018). Traction-energy balancing adaptive control with slip optimization for wheeled robots on rough terrain. Cognitive Systems Research, 49, 142–156. https://doi.org/10.1016/j.cogsys.2018.01.007
7. Lee, J., & Edgar, T. F. (2018). Three-Parameter Models for Conservative Relay Feedback Autotuning. 2018 IEEE 14th International Conference on Control and Automation (ICCA), 975–980. https://doi.org/10.1109/icca.2018.8444315
8. Kumar, A. (2024). Microcontroller-Based design of Self-Balancing Two-Wheeler robotic vehicles using gyroscopic sensor. Journal of Instrumentation and Innovation Sciences, 9(1), 19–25.
https://doi.org/10.46610/jiis.2024.v09i01.004
9. Li, Z. (2023). Review of PID control design and tuning methods. Journal of Physics Conference Series, 2649(1), 012009. https://doi.org/10.1088/1742-6596/2649/1/012009
10. Kim, J., & Lee, J. (2018). Traction-energy balancing adaptive control with slip optimization for wheeled robots on rough terrain. Cognitive Systems Research, 49, 142–156. https://doi.org/10.1016/j.cogsys.2018.01.007
11. Chen, Y. J., Tung, W., Lee, W., Patel, B., Bučinskas, V., Greitans, M., & Lin, P. T. (2023). Designing and controlling a self-balancing platform mechanism based on 3-RCC spherical parallel manipulator. Robotic Systems and

Applications, 3(1), 1–16. https://doi.org/10.21595/rsa.2023.23015

12. Manolescu, D., & Secco, E. L. (2023). Development of a 3D printed biologically inspired monoped Self-Balancing robot. International Journal of Robotics and Control Systems, 3(1), 84–97. https://doi.org/10.31763/ijrcs.v3i1.841

13. Damayanti, F. S., Noviana, R. S., Lestari, Y. I., Akhsan, H., & Ismet, I. (2024). Exploring Applications of Lagrange’s Equations in Technology: A Systematic Literature review. Aceh International Journal of Science and Technology, 13(2), 123–130. https://doi.org/10.13170/aijst.13.2.39380.