PROJECT ABSTRACT

Self-Balancing Two-Wheel Robot with PID Control

A two-wheel self-balancing robot using MPU6050 tilt sensing and a tuned PID loop on Arduino, with Bluetooth control and a documented tuning procedure. Delivered built-to-order with working prototype, firmware, tuning guide, report, PPT and viva kit.

01. Title

Self-Balancing Two-Wheel Robot with PID Control

02. Introduction / Background

The inverted pendulum is the canonical controls problem: a body that falls unless a feedback loop corrects it continuously. A self-balancing two-wheel robot is that problem with wheels — it survives only because tilt is measured and corrected about a hundred times per second. Students meet PID control as equations; this project makes it physical, where wrong gains have immediate, visible, face-planting consequences, and right gains feel like a small miracle.

03. Problem Statement

Balancing-robot builds found online ship with copy-pasted PID gains that work on one specific chassis and fail on every other. The project addresses this by making tuning the deliverable's core: a documented oscillation-method procedure the student performs on their own build, so the report's gains are measured on the actual hardware.

04. Objectives

- Sense tilt with an MPU6050 and complementary filter.
- Implement a ~100 Hz PID balance loop on Arduino.
- Drive the wheels through H-bridges with fast corrective commands.
- Provide Bluetooth drive via balance-setpoint offset.
- Deliver a student-performed PID tuning procedure with recorded gains.
- Deliver the complete student kit: prototype, firmware, docs, report, PPT and viva Q&A.;

05. Proposed Methodology

The build proceeds in three stages. (1) Mechanical: compact rigid chassis, high-torque geared motors, IMU mounted at the rotation axis height. (2) Firmware: complementary filter for tilt, PID loop at approximately 100 Hz, H-bridge drive, Bluetooth setpoint-offset control. (3) Tuning: the student runs the oscillation-method procedure, records the gains, and verifies upright hold and driven motion on a flat floor.

06. System Architecture / Working

The complementary filter fuses accelerometer and gyroscope data into tilt angle. The PID loop compares tilt to the upright setpoint 100 times per second (design) and commands the wheels through the H-bridges — driving under the fall direction. Bluetooth input offsets the setpoint, making the robot lean and travel while balancing. Tuned P, I and D gains convert the initial wobble into a stable upright hold.

07. Hardware / Software Requirements

- Arduino Nano
- MPU6050 IMU
- 2 high-torque geared DC motors
- H-bridge driver (L298N/TB6612 class)
- HC-05 Bluetooth module
- Li-ion battery + 5 V regulator
- Fabricated compact chassis
- Arduino IDE (C/C++ firmware)

08. Expected Outcomes

- A robot that balances upright and drives under Bluetooth command.
- Firmware with complementary filter and ~100 Hz PID loop.
- Student-tuned PID gains recorded with the tuning log.
- A complete report, PPT and viva Q&A; on inverted-pendulum control and PID tuning.

09. Applications

- Controls-lab demonstrations of PID on real hardware.
- Teaching sensor fusion (complementary filter) in embedded courses.
- A platform for advanced control experiments (LQR, fuzzy).

10. Future Scope

- Encoder-based velocity and position hold.
- LQR or fuzzy-logic controller comparison study.
- Obstacle sensing with ultrasonic modules.
- Segway-style rider platform scaling study.