TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

A MINOR PROJECT REPORT ON PID IMPLEMENTATION ON A SELF BALANCING BOT

SUBMITTED BY:

BISHAKHA POKHREL(PUR078BEI010)

RUDRA KHATRI(PUR078BEI031)

SNEHA YADAV(PUR078BEI040)

SUSANT DAHAL (PUR078BEI046)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

2 MARCH , 2025

A MINOR PROJECT REPORT ON PID IMPLEMENTATION ON A SELF BALANCING BOT

By

BISHAKHA POKHREL(PUR078BEI010)

RUDRA KHATRI(PUR078BEI031)

SNEHA YADAV(PUR078BEI040)

SUSANT DAHAL (PUR078BEI046)

Project Supervisor

Er. Manoj Kumar Guragai,

A project submitted to the Department of Electronics and Computer Engineering in partial fulfillment of the requirements for the Bachelor's Degree in Computer Engineering

Department of Electronics and Computer Engineering
Purwanchal Campus, Institute of Engineering
Tribhuvan University
Dharan, Nepal

2 March , 2025

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to the Department of Electronics and Computer Engineering, Purwanchal Campus, Institute of Engineering, for their invaluable support and resources throughout the preparation of this proposal.

Our sincere appreciation goes to **Er. Manoj Kumar Guragai**, for his expert guidance, insightful suggestions, constructive feedback, and continuous encouragement, which have been instrumental in shaping this proposal.

We are deeply thankful to the Department for providing us with the opportunity to undertake this collaborative project. This experience has significantly enhanced our practical understanding and application of knowledge, allowing us to develop a project as part of our Minor Project for Third Year. The skills and insights gained from this endeavor will undoubtedly contribute to our academic and professional growth.

Our appreciation also extends to our respected seniors, whose valuable knowledge, experiences, and advice have helped us navigate challenges during this project. We are equally grateful to our friends for their support and assistance, both directly and indirectly, throughout this journey.

BISHAKHA POKHREL(PUR078BEI010)

RUDRA KHATRI(PUR078BEI031)

SNEHA YADAV(PUR078BEI040)

SUSANT DAHAL (PUR078BEI046)

ABSTRACT

This project focuses on the development of a self-balancing robot designed to demonstrate the application of robotics and control systems. The robot is built using an Arduino Nano, MPU6050 sensor, encoder motors, L298N motor driver, breadboard, jumper wires, and a battery for power.The primary goal of the robot is to maintain its balance autonomously, responding to changes in its orientation. The MPU6050 sensor detects the tilt angle of the robot and sends this information to the Arduino Nano, which then controls the motors through the L298N motor driver. The encoder motors provide the necessary movement to adjust the robot's position, ensuring that it remains upright. This project highlights the integration of essential hardware components and provides insight into the real-time processing required to keep a robot stable. The robot serves as an example of how sensors and microcontrollers work together to achieve dynamic stability and movement, making it an excellent tool for understanding basic robotics and sensor-driven systems.

Keywords: Arduino Nano, MPU6050 sensor, encoder motors, L298N motor driver, breadboard, jumper wires

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

PID	: Proportional-Integral-Derivative
MPU	: Motion Processing Unit
GND	:Ground
TX and RX	:Transmit and Receive
An and Dn	:Analog pins and Digital pins
I2C	:Inter-Integrated Circuit

1. INTRODUCTION

1.1 Background

Robotics has significantly impacted modern technology by automating processes and offering innovative solutions to complex challenges. Self-balancing robots serve as a prime example of advanced control systems, showcasing the principles of dynamic balance and stabilization. These robots find applications in fields such as education, robotics competitions, and prototypes for personal transportation devices. The goal of this project is to simplify the design process and improve the understanding of control systems, making them more accessible to students and enthusiasts

1.2 Problem Definition

The primary challenge in developing a self-balancing robot lies in achieving dynamic stabilization of an inherently unstable two-wheeled system. This requires precise control algorithms to process real-time sensor data and adjust motor outputs accordingly. The robot must continuously monitor its orientation and make rapid corrections to maintain an upright position. Ensuring seamless integration of hardware and software components is crucial for the robot's effective performance.

1.3 Motivation

Developing a self-balancing robot offers practical experience in control systems and sensor integration, enhancing understanding of dynamic stabilization. This project addresses the classic inverted pendulum problem, providing a hands-on approach to mastering complex control challenges. Insights gained can inform the development of advanced transportation solutions and robotic assistants, contributing to technological progress in personal mobility and service robotics.

1.4 Objectives

The objective of this project is to develop a two-wheeled robot capable of autonomously maintaining its balance using sensor feedback and a PID (Proportional-Integral-Derivative) controller.

1.5 Scope of The Project

The scope of this project includes designing, programming, and assembling a self-balancing robot using affordable components. Applications extend beyond educational purposes to include robotics research, competition scenarios, and even industrial demonstrations of control system principles. The modular design allows for future enhancements, such as obstacle detection and autonomous navigation.

2. RELATED THEORY

2.1 Hardware Tools

2.1.1 Arduino Nano

Arduino Nano is a small and compact microcontroller board based on the ATmega328P chip. It is like the brain of a self-balancing robot, responsible for processing data and controlling the motors. Key featuresof it are that it is small and lightweight (ideal for robots),30 pins (14 digital, 8 analog, and power pins),supports USB, I2C and can run on 5V (USB) or 7-12V (external power).

In a self-balancing robot, the Arduino Nano reads data from the MPU6050 sensor, processes it, and sends signals to the motor driver (L298N) to adjust the motor speed and direction to maintain balance.

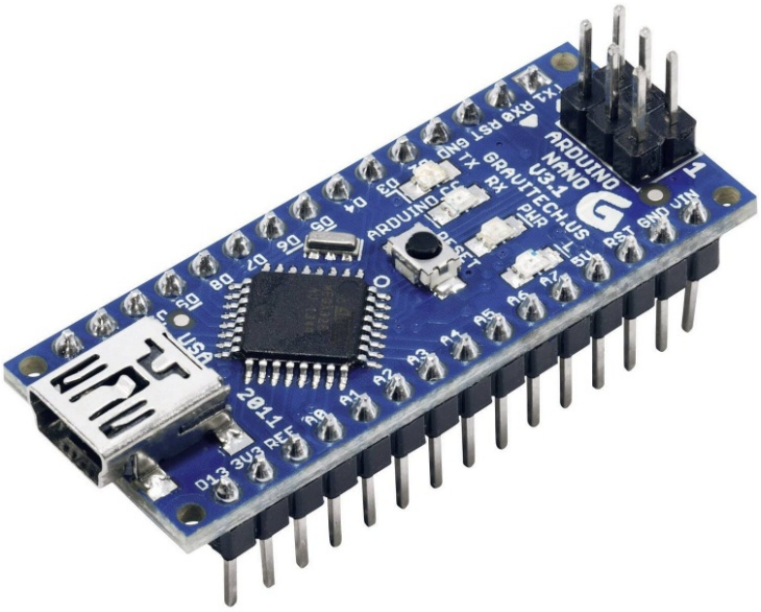

Figure 2.1: Arduino Nano.

2.1.2 MPU6050 – Gyroscope and Accelerometer Sensor

The MPU6050 is a motion sensor module that combines a 3-axis gyroscope and a 3-axis accelerometer in a single chip. It helps measure the robot's tilt, angular velocity, and acceleration, which are crucial for maintaining balance in a self-balancing robot. In self balancing robot, accelerometer detects the tilt angle of the robot, gyroscope measures the angular movement (how fast the robot is tilting), Arduino Nano processes this data and adjusts the motor speed through the L298N motor driver to keep the robot balanced and Complementary Filter is used to improve accuracy.

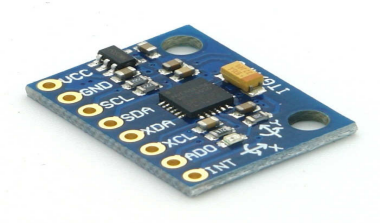

Figure 2.2: MPU6050 – Gyroscope and Accelerometer Sensor.

2.1.3 L298N Motor Driver

It is a dual H-Bridge motor driver module for DC motor speed and direction control.

How It Works in a Self-Balancing Robot:

1. The Arduino Nano sends control signals to the L298N based on data from the MPU6050 sensor.
2. The L298N adjusts the motor speed and direction to keep the robot upright.
3. If the robot tilts forward or backward, the L298N changes the motor speed to counterbalance the movement.

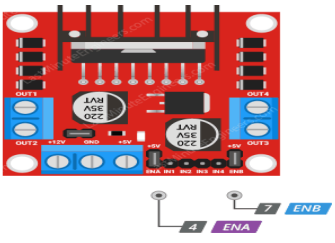

Figure 2.3: L298N Motor Driver.

2.1.4 Encoder Motors

Encoder motors are DC motors with built-in rotary encoders that help measure speed and position accurately.They provide real-time feedback on how fast and in which direction the wheels are moving.

How They Work in the Robot:

1. Motion Control: The L298N motor driver controls the power and direction of the motors.
2. Speed Feedback: The encoder generates pulses as the motor rotates, which are read by the Arduino Nano to calculate speed and position.
3. Balance Adjustment: The Arduino processes the MPU6050 sensor data and adjusts the motor speed to correct the robot's tilt and keep it upright.

Figure 2.4: Encoder Motors.

2.1.5 BreadBoard

A breadboard (sometimes called a plug-block) is used for building temporary electronics circuits. It is useful to designers because it allows components to be removed and replaced easily. It is useful to the person who wants to build a circuit to demonstrate its action, then to reuse the components in another circuit.

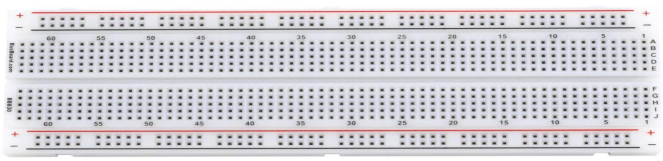

Figure 2.5: BreadBoard.

2.2 Software Tools

2.2.1 Arduino IDE

The Arduino IDE is the main software used to program the Arduino Nano for the self-balancing robot. We write code in C/C++ to control the robot's parts, like the MPU6050 sensor, L298N motor driver, and encoder motors. The MPU6050 library helps read the sensor data, while a PID control algorithm adjusts the motor speeds to keep the robot balanced based on its tilt. The Serial Monitor and Serial Plotter in the Arduino IDE let you see real-time data and debug the system.The Arduino IDE is the main tool for writing, uploading, and testing the code.

Figure 2.6: Arduino IDE

3. LITERATURE REVIEW

3.1 Relevant Research

Studies highlight the effectiveness of PID controllers in maintaining balance for two-wheeled robots. Research by [Duc An Pham and Trung Nghia Pham,2022] in title ”Determination of a Tilt Angle for the Automatic Balancing System with the Inertial Measurement Unit MPU6050” demonstrated the integration of MPU6050 for precise tilt measurement in robotic systems. Various open-source designs have explored the balance of simplicity and functionality, providing valuable insights into optimizing robot stability.

3.2 Innovations and Challenges

While existing designs are efficient, challenges include real-time tuning of control parameters, power management, and achieving stability over varied terrains. Bluetooth-based PID tuning presents a novel approach to addressing these challenges. Additionally, innovations in sensor fusion algorithms and motor control have significantly improved the efficiency of self-balancing systems.

4. METHODOLOGY

4.1 System Block Diagram

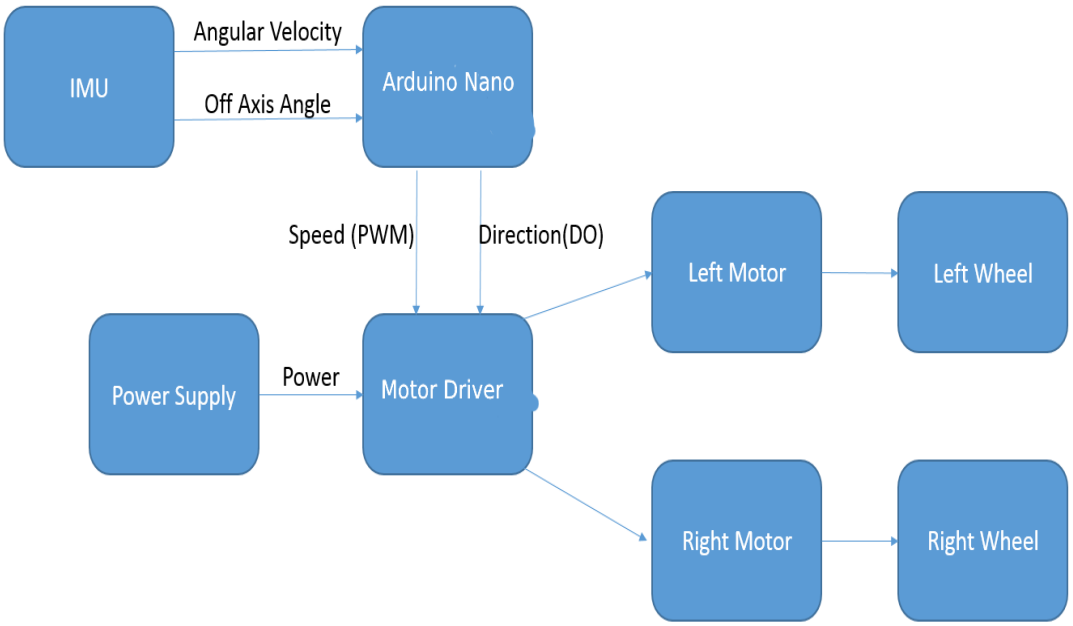

Figure 4.1: System Block Diagram.

4.2 Working Principle

The working principle of a self-balancing robot involves continuously monitoring its tilt and adjusting its movement to stay upright. The MPU6050 sensor measures the robot's tilt and angular velocity, sending this data to the Arduino Nano. The Arduino processes this information to determine if the robot is tilting too far forward or backward. Based on this, the Arduino sends control signals to the L298N motor driver, which adjusts the speed and direction of the encoder motors. The encoder motors provide real-time feedback by counting pulses generated as the motors rotate, allowing the Arduino to track the robot's speed and position. If the robot tilts, the motors adjust their speed to move the robot in the opposite direction, correcting the tilt. This cycle of measurement and adjustment happens continuously, ensuring the robot stays balanced and stable.

4.3 System Flowchart

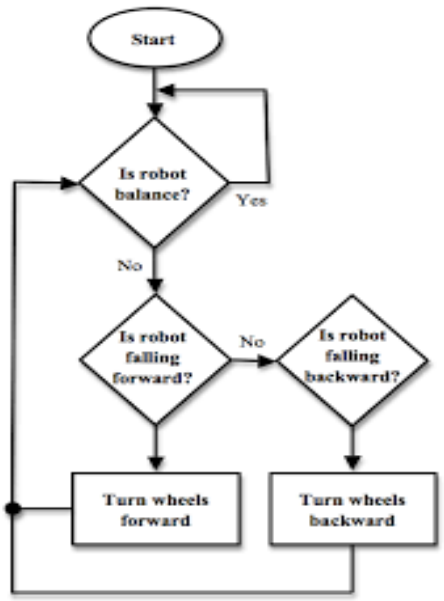

Figure 4.2: System Flowchart.

4.4 System Circuit Diagram

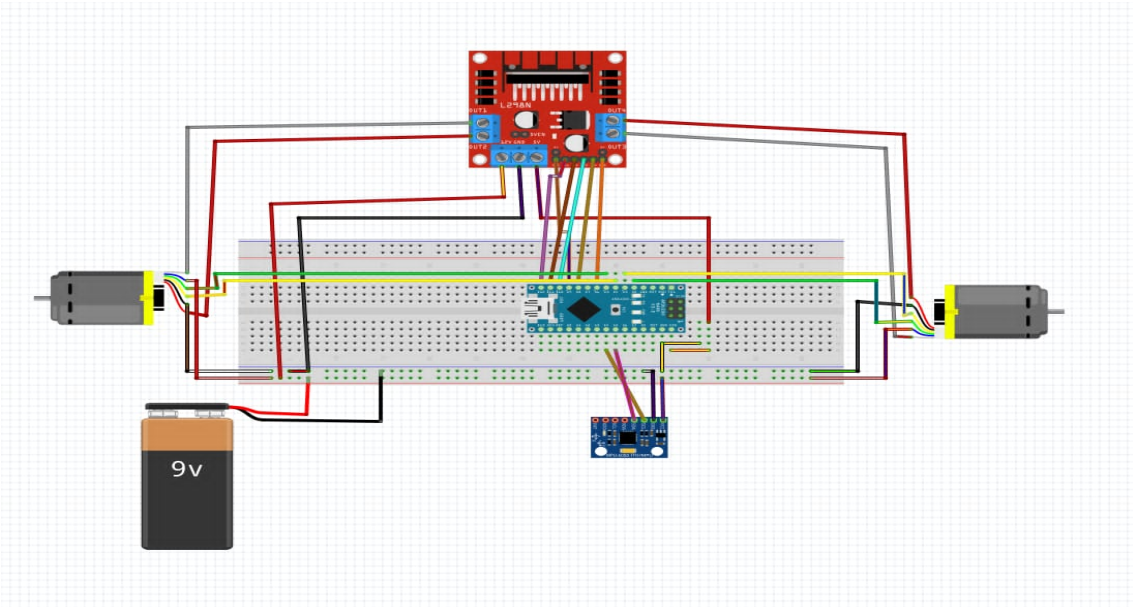

Figure 4.3: System Circuit Diagram.

4.5 System Schematic Diagram

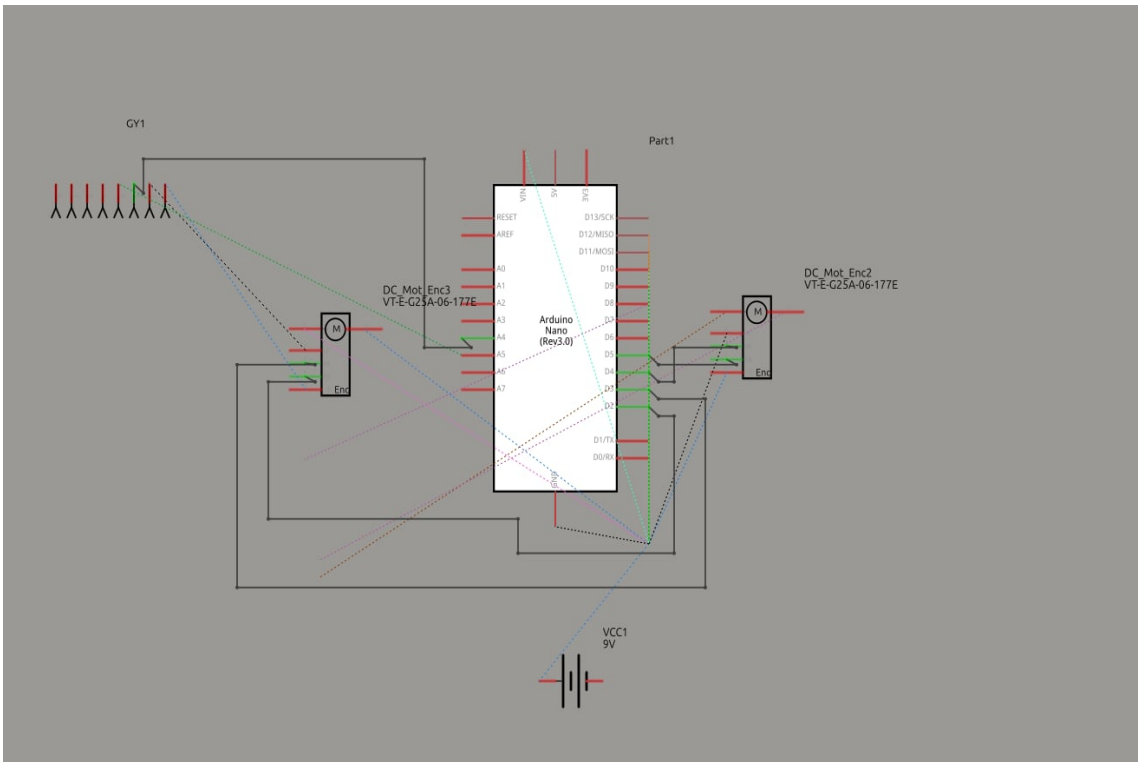

Figure 4.4: System Schematic Diagram.

5. RESULTS

5.1 Results

The self-balancing bot successfully maintains its balance for a short period using a combination of sensors (MPU6050) and a PID controller algorithm. It reacts to tilts by adjusting the motor speeds, helping it stay upright. The encoder feedback system ensures that both wheels move in coordination, further improving stability. The complementary filter effectively reduces noise in sensor readings, allowing the bot to make real-time corrections. However, the bot still experiences oscillations, meaning it continuously moves slightly forward and backward instead of staying perfectly still. While the system works, it is not yet stable for long durations.

Figure 5.1: Serial plotter:Stable postion

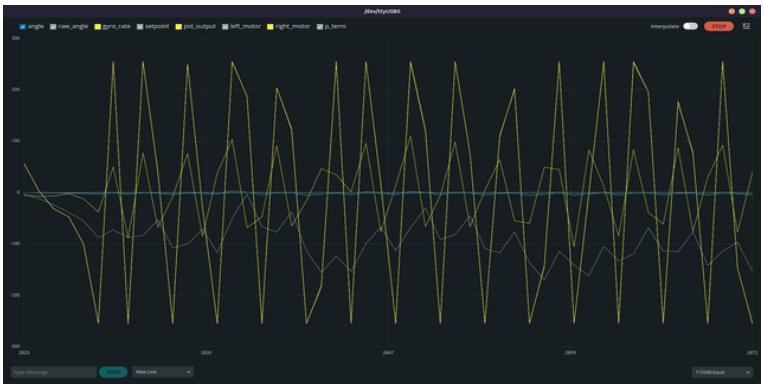

Figure 5.2: Serial plotter:Bot fixing/trying to be stable using PID

5.2 Limitations of model

1. Oscillations and Overcorrection: The bot continuously moves back and forth instead of staying steady, likely due to imperfect PID tuning.
2. Sensor Drift: The MPU6050 sensor can develop small errors over time, affecting angle calculations and stability.
3. Motor Response Delay: Motors need a minimum power threshold to start moving, making small adjustments difficult and leading to instability.
4. Weight Distribution and Frame Issues: Uneven weight balance or a less rigid frame can introduce vibrations, affecting overall performance.

6. DISCUSSION AND CONCLUSION

The self-balancing bot successfully maintains balance for a short period but struggles with long-term stability due to oscillations and sensor drift. The PID controller plays a crucial role in stability, but fine-tuning is required to reduce overcorrection. The MPU6050 sensor provides accurate tilt readings, but minor drift over time affects performance. Additionally, motor response delays and weight distribution impact smooth balancing. To improve the bot's performance, better PID tuning, sensor calibration, and a more rigid frame design are needed. Overall, the project demonstrates the feasibility of self-balancing technology but highlights the need for further refinement to achieve stable and long-lasting balance.

REFERENCES

[1] R. Rajasekaran, S. R. S. S. R. V. V. Shastry, ”Design of Self-Balancing Robot using Arduino,” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 7, no. 4, pp. 345-352, 2018.

[2] M. K. Soni, A. Tiwari, ”Arduino-Based Self-Balancing Robot,” IEEE International Conference on Robotics and Automation, pp. 489-495, 2019.

[3] G. T. A. S. R. Pradeep, ”Self-Balancing Robot using MPU6050 and Arduino,” Journal of Electrical and Electronics Engineering, vol. 8, pp. 65-72, 2020.

[4] M. K. Soni, Arduino Projects for Beginners: A Hands-On Guide for Building Robotics Systems, Prentice Hall PTR, 2018, ch. 3 and ch. 5.

[5] Google, ”Self-Balancing Robot using Arduino,” [Online]. Available: https://www.google.com.

[6] OpenAI, ”ChatGPT: Self-Balancing Robot Project Assistance,” [Online]. Available: https://chat.openai.com.

APPENDIX A

Item	Quantity	Unit Cost (NPR)	Total Cost (NPR)
Arduino Nano	1	1000	1000
L298N Motor Drivers	1	400	400
MPU6050	1	500	500
Encoder motors	2	938	1876
Breadboard	1	170	170
Battery	1	989	989
Wheels	1 set	320	320
Jumper Wires	1 set	200	200
Miscellaneous	-	200	200
Total	-	-	**5655**

Table 6.1: Project Budget for PID Implementation - Self Balencing Bot

APPENDIX B

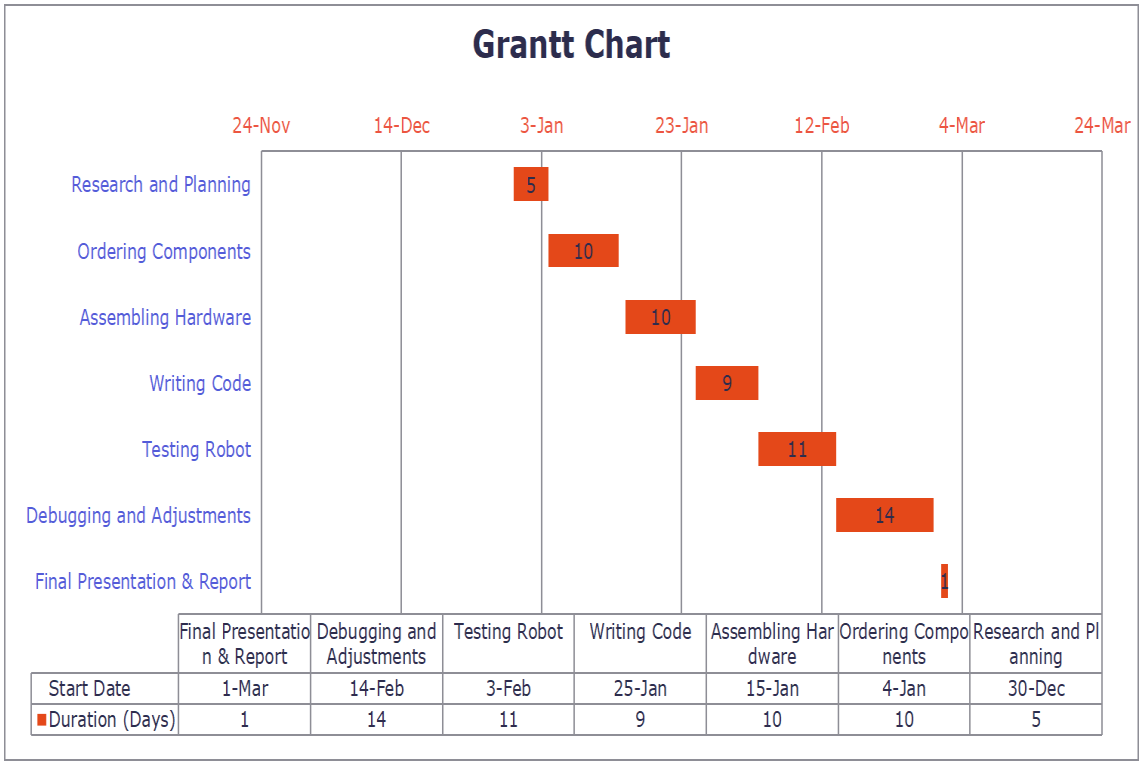

Figure 6.1: Gantt Chart