

M. Tech. Dissertation Report

**FABRICATION AND TESTING OF  
SELF-BALANCING ROBOT  
WITH REMOTE CONTROL**

IS SUBMITTED IN PARTIAL FULFILLMENT OF THE  
MASTER OF TECHNOLOGY IN EMBEDDED SYSTEMS

Submitted by:

VINITA KUMARI

2017PEB5178

Supervisor:

RAKESH BAIRATHI

Associate Professor, ECE



Department of Electronics and Communication Engineering

Malaviya National Institute of Technology, Jaipur

June 2019

© Malaviya National Institute of Technology, Jaipur

All Rights Reserved

# **CERTIFICATE**

This is to certify that the Dissertation Report on “FABRICATION AND TESTING OF SELF-BALANCING ROBOT WITH REMOTE CONTROL” by VINITA KUMARI is bonafide work completed under my supervision, hence approved for submission in partial fulfillment for the Master of Technology in EMBEDDED SYSTEMS, Malaviya National Institute of Technology, Jaipur during academic session 2018-19.

Date:

(Rakesh Bairathi)

Associate Professor, ECE Deptt.

MNIT Jaipur

# DECLARATION

This is to certify that the dissertation report entitled “FABRICATION AND TESTING OF SELF-BALANCING ROBOT WITH REMOTE CONTROL” is being submitted by me in partial fulfillment of degree of Master of Technology in Embedded Systems during 2017-19. This work is carried out by me under the supervision of Rakesh Bairathi, Associate Prof., Deptt. of ECE, MNIT Jaipur. I am fully responsible for the material used in this report in case if any discrepancies arises. The report (fully/partly) is not submitted for the award of any other degree. Wherever I have consulted the published work (in any form) of others, it is clearly attributed. Wherever I have quoted from the work of others, the source is always given. I have acknowledged all the relevant (to the best of my knowledge) sources in the report. With the exception of above, this report is entirely my own work.

VINITA KUMARI

EMBEDDED SYSTEMS

M.Tech

2017PEB5178

## ACKNOWLEDGEMENT

First of all, I would like to take this opportunity to thank my Supervisor (Guide), **Rakesh Bairathi**, Associate Prof., Dept. of Electronics and Communication Engineering, Malaviya National Institute of Technology. I am very much indebted to him for his valuable support, expertise and guidance that I received while working on my dissertation and throughout my studies. He consistently steered me in the right direction whenever he thought I needed it. Without him, completion of this project on time would have seemed a far-fetched dream. In this respect, I express my deep sense of gratitude and respect towards him.

I would also like to thank **Dr. D. Boolchandani**, HOD, ECE and **Dr. Ghanshyam Singh**, PG Coordinator for their cooperation and help rendered in numerous ways for the successful completion of this work.

I take this opportunity to express my deepest regards and obligation to my family and friends whose support, encouragement and cooperation I can never forget in life.

Lastly, I am thankful to **ALL** those who have supported me directly or indirectly during my dissertation work.

VINITA KUMARI

(2017PEB5178)

## ABSTRACT

A two-wheeled self-balancing robot is a classic example of inverted pendulum system; i.e., it is a naturally unstable system. In recent years, it has become a hot area of research due to vast possibilities of its use in various areas like Segway vehicles, better wheelchairs, efficient robots, etc. In this project, an attempt has been made to address the inherent instability property of two-wheeled self-balancing robot using PID control system. The goal is to construct a low-cost prototype which can successfully balance itself, and can be used to practically analyze the responses and stability of the system for different variables.

**Keywords:** Segway, PID Controller, Arduino

# CONTENTS

<b>Title Page</b> .....	<b>i</b>
<b>Certificate</b> .....	<b>ii</b>
<b>Declaration</b> .....	<b>iii</b>
<b>Acknowledgement</b> .....	<b>iv</b>
<b>Abstract</b> .....	<b>v</b>
<b>Contents</b> .....	<b>vi</b>
<b>List of Tables</b> .....	<b>viii</b>
<b>List of Figures</b> .....	<b>ix</b>
<b>List of Abbreviations</b> .....	<b>xi</b>
Chapter 1: Introduction	1
1.1 Motivation	1
1.2 Concept of stability	1
1.3 Project objective	2
1.4 Scope of work	2
1.5 Outline of project	3
Chapter 2 : Literature Review	4
2.1 Introduction	4
2.2 Literature review	5
Chapter 3 : Hardware and Software	8
3.1 Project Conceptualization	8
3.2 Hardware Development	10
3.2.1 Arduino UNO	13
3.2.2 Stepper motor	14
3.2.3 Motor Controller	15
3.2.4 IMU Sensor	16
3.2.4.1 Three axis MEMS accelerometer	18
3.2.4.2 Three axis MEMS gyroscope	18
3.2.4.3 DMP Processor	18
3.2.5 Bluetooth module	18
3.2.6 Arduino Nano	19

3.2.7 Joystick	20
3.2.8 Power source	21
3.2.9 Frame	21
3.2.10 Wheel	23
3.3 Software used	24
Chapter 4 : Kinematics and Control Basics	26
4.1 Basics of stability	26
4.2 Types of controllers used	28
4.3 PID controller	28
4.3.1 Overview	28
4.3.2 The characteristics of PID controller	31
Chapter 5 : Construction details	32
5.1 Connection in robots	33
5.2 Wiring in remote control	34
5.3 I2C interface	35
Chapter 6 : Working and Results	38
6.1 Challenges	38
6.2 Working of the robot	38
6.3 Concept of balancing	39
6.4 PID tuning	43
6.5 Algorithms for hardware testing	49
6.5.1 Algorithms for MPU6050	49
6.5.2 Algorithms for Joystick	50
6.6 Algorithms for bluetooth	53
6.6.1 For slave device	53
6.6.2 For master device	54
6.7 Algorithm for self-balancing robot	55
6.8 Algorithm to steer the robot by remote control	56
Chapter 7 : Conclusion and Future Work	58
7.1 Conclusion	58
7.2 Future work	58
References	60

## List of Tables

3.1	Components required and their specifications	11
3.2	Specification of the Arduino UNO	14
3.3	Specifications of DC stepper motor	15
3.4	Specifications of MPU6050	17
3.5	Bluetooth specification	19
3.6	Specification of battery	21
4.1	Parameters of inverted pendulum system	27
4.2	Effects of PID tuning	31
5.1	Specification of I2C communication	37
6.1	Tilt angle of the robot for forward direction	41
6.2	Tilt angle of the robot for backward direction	41



# List of Figures

2.1	Segway PT by Segway Inc.	5
2.2	nBot version 6.0 developed by Anderson	5
2.3	Bimbo developed by Martins and Nunes	6
3.1	Flow chart of the project conceptualization	9
3.2	Block diagram of self-balancing robot	12
3.3	Block diagram of remote control	12
3.4	View of Arduino UNO	13
3.5	View of stepper motor (NEMA-17)	15
3.6	View of DRV8825 and heatsink	16
3.7	View of IMU sensor (MPU6050)	17
3.8	View of bluetooth module (HC05)	18
3.9	View of Arduino Nano	19
3.10	View of Wii nunchuk	20
3.11	Pin diagram of Wii nunchuk	20
3.12	Image of 1800mAh Li-Po battery	21
3.13	View of the robot frame	22
3.14	View of wheel	23
3.15	View of the Arduino IDE software	25
4.1	Inverted pendulum	26
4.2	FBD of inverted pendulum	27
4.3	Set point and actual tilt angle of the robot	29
4.4	PID controller block diagram	30
5.1	View of the robot after wiring of components	32
5.2	Connection diagram for robot components	34
5.3	Connection diagram for remote control components	35
5.4	I2C Protocol	36
6.1	Response of the robot corresponding to tilt angle	42
6.2	Step response for $K_p = 10$ , $K_d = 0$ and $K_i = 0$	44
6.3	Step response for $K_p = 15$ , $K_d = 0$ and $K_i = 0$	44

6.4	Step response for $K_p = 15$ , $K_d = 10$ and $K_i = 0$	45
6.5	Step response for $K_p = 15$ , $K_d = 40$ and $K_i = 0$	45
6.6	Step response for $K_p = 15$ , $K_d = 40$ and $K_i = 0.5$	46
6.7	Step response for $K_p = 15$ , $K_d = 40$ and $K_i = 2.0$	46
6.8	View of robot with four supports with no power supply	47
6.9	View of robot with four supports when it starts balancing itself	48
6.10	Output of MPU6050 testing on the serial monitor	50
6.11	Output displayed on serial monitor when stick moved right	51
6.12	Output displayed on serial monitor when stick moved left	51
6.13	Output displayed on serial monitor when stick moved up	52
6.14	Output displayed on serial monitor when stick moved down	52
6.15	Serial monitor output after entering AT commands for slave device	53
6.16	Serial monitor output after entering AT commands for master device	54
6.17	Final working model of robot without four supports	56
6.18	Final working model of robot with four supports	57

## List of Abbreviations

1	SBR	Self-Balancing Robot
2	PIC	Peripheral Interface Controller
3	AVR	Advanced Virtual RISC
4	RISC	Reduced Instruction Set Computer
5	SRAM	Static Random Access Memory
6	EEPROM	Electrically Erasable Programmable Read-Only Memory
7	DC	Direct Current
8	DOF	Degree Of Freedom
9	MEMS	Micro Electro-mechanical system
10	DMP	Digital Motion Processor
11	IMU	Inertia Measurement Unit
12	3-D	Three Dimensional
13	ADC	Analog to Digital Converter
14	12C	Inter-Integrated Circuit
15	DAC	Digital to Analog Converter
16	FIFO	First In First Out
17	PWM	Pulse Width Modulation
18	USART	Universal Synchronous Asynchronous Receiver Transmitter
19	LED	Light Emitting Diode
20	RF	Radio Frequency
21	Li-Po	Lithium Polymer
22	IR	Infrared Radiation
23	IDE	Integrated Development Environment
24	MAC	Media Access Control
25	SDA	Serial Data line
26	SCL	Serial Clock line

# CHAPTER 1

## Introduction

### 1.1 Motivation

In today's era, whenever we talk about technological advancements and the resulted automation in our lives, the first thing that comes in our minds is the recent developments in field of electronics and the better, faster, efficient and smart machines and robots it has created. We desire for machines with smart thinking, self-sustaining and decision making capabilities. We want better robots which can automate the entire work chain in a company, which can replace a human workforce. Such is the desire for optimized systems that today we prefer wheeled robot over two-legged humanoid robot for having faster navigation capability, and among that we prefer two-wheeled robot over three/four-wheeled robot for having compact design and faster movement speed.

Also, since the introduction of commercial Segway vehicles, two-wheeled self-balancing robot has become a popular area of interest for researchers for its vast possibilities to be used for

- Means of accident free, environment friendly local transportation.
- Self-balancing wheelchairs for disabled patients.
- Compact and efficient robots to be used in hotels, warehouses, etc.
- And many more areas of application.

All these offered challenges and the vast area of possible applications by a two-wheeled self-balancing robot were the major driving force behind me choosing this project.

### 1.2 Concept of stability

A two-wheeled robot is basically an inverted pendulum system, with its body acting as an inverted pendulum (has natural tendency to fall down under gravity) pivoted on wheels' axis. In order to keep the robot stable, its center of mass should lie just above the wheels' axis. Since the robot is free to move in only forward and reverse directions, the control system on robot should move the wheels in the same direction

that of the falling robot, so that an opposite torque can be applied on robot body which can oppose the falling motion. In this way, the robot can keep balancing itself. The speed with which the wheels should move will depend on various parameters like angle of tilt, falling velocity, direction of motion (straight forward and backward linear motion, or, rotational motion while taking turns), etc. This is where the effectiveness of control systems come in picture. The more accurate and responsive control system is, more will be the resulted stability of system.

### **1.3 Project Objective**

Our aim is to design and fabricate a two-wheeled self-balancing robot with the help of PID control system. An IMU sensor, which combines gyroscope & accelerometer, will be used to continuously measure angle, direction and velocity of tilt of robot, and will send this data to microcontroller, which in turn, through motor controller, will give command to stepper motors to move in the same direction that of the tilt. It will exert an opposite torque to the body of robot, which in turn will counter the falling action and will stabilize the system by bringing back the center of gravity of robot just above the wheels. The robot will be able to balance itself in the mean position while taking turns, and even will be able to stabilize itself quickly when destabilized by an external force/blow/jerk. Also, a Bluetooth device will be added to run the robot with help of joystick.

The ultimate goal to achieve will be fabricating a low cost, compact, Bluetooth-controlled two-wheeled self-balancing robot with PID controller which can balance itself with utmost precision and responsiveness; and using to robot to practically analyze the responses and stability of the system for different control gain parameters.

### **1.4 Scope of work**

The scope of work is as follows:

- i. To conceptualize design of robot.
- ii. To select components/hardware meeting our requirement.
- iii. To fabricate the robot by assembling those components.
- iv. To write the codes required.
- v. To test the robot for performance & outcomes

## 1.5 Outline of project

The outline of this thesis is as follows:

**Chapter 1:** This chapter deals with the introduction part, which includes motivation behind this project, fundamentals of the project, objective of the project and layout of the dissertation.

**Chapter 2:** This chapter deals with the literature review of all those relevant documents which we have referred during the completion of this project.

**Chapter 3:** This chapter covers selection criteria of components and their specifications along with application notes and data sheets. Also, Arduino IDE software is discussed here.

**Chapter 4:** This chapter gives a systematic idea about kinematics and control basics of robot, controller types available and PID controller overview.

**Chapter 5:** This chapter covers the connection details in robot and remote control.

**Chapter 6:** This chapter covers concept of working of robot including algorithms, PID tuning and the final results obtained.

**Chapter 7:** This chapter concludes this dissertation and projects the future work and applications possible.

# CHAPTER 2

## Literature Review

### 2.1 Introduction

From the beginning of development of robots, humans had been obsessed with humanoid robots, which can look, walk, talk and function like an actual human being. But a humanoid robot is very difficult to make because of its higher degree of freedom. So, slowly the researchers have drifted from legged robots to wheeled robots because of much lesser complexity offered by them due to their lesser degree of freedom. Among these wheeled robots, 3 or 4 wheeled robots are easier to fabricate and operate because of easier balancing; but they are bulkier than two-wheeled robots. Two-wheeled robots are much compact and faster, but they are difficult to balance because of their naturally unstable state.

For many years, two-wheeled self-balancing robot has been a major area of interest for researchers due to the opportunities and challenges it poses. They are trying to develop different balancing techniques with the help of different methods, controllers, sensors and filters to achieve a fast, responsive and stable system. The higher will be the reliability of the system, higher will be chances to commercialize that technique.

Also, since the commercial release of “Segway PT (personal transporter)” in 2001, invented by Dean Kamen and manufactured by Segway Inc., big MNCs have started investing in R&D of Segway vehicles. The latest Segway product has a maximum driving speed of 20 km/hr, and its ride is much safer, comfortable and reliable. People are using these battery operated vehicles as mode of pollution free personal transportation for short distances/local convenience. Also, these are being used to roam around in parks, malls, shopping complexes, and large institutional campuses. In fact now police in Germany and Italy have started using these Segway vehicles in place of their traditional 4-wheelers.

## 2.2 Literature review

The popular **Segway PT** was invented by Kamen, D. [1] and commercialized by Segway Inc. in 2001. The vehicle is shown in figure 2.1. This robot was a battery operated, two-wheeled personal transporter which was able to balance itself using a complex control system with sensors to measure tilt angle and five gyroscopic stabilizers (3 working, 2 standby) [2]. It was the first successful commercial robot vehicle gaining Segway Inc. lots of profit, fame, and recognition.



Fig. 2.1 Segway PT by Segway Inc. [1]

Anderson, D. P. [3] created **nBot** (fig. 2.2), an award winning robot from NASA, in 2003. The robot uses sensors to measure tilt angle and direction of falling down of robot to move wheels in the same direction, so that the center of gravity of robot can always be brought back above wheels, and robot can be balanced.

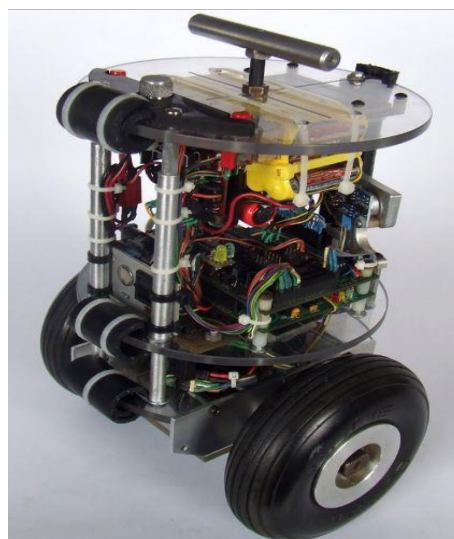


Fig. 2.2 nBot version 6.0 developed by Anderson [3]



Martins, R. S. and Nunes, F. [4] developed a robot named **Bimbo** (fig. 2.2), and used it to study the efficiency and performance of different control systems such as PID, pole placement, and adaptive control.

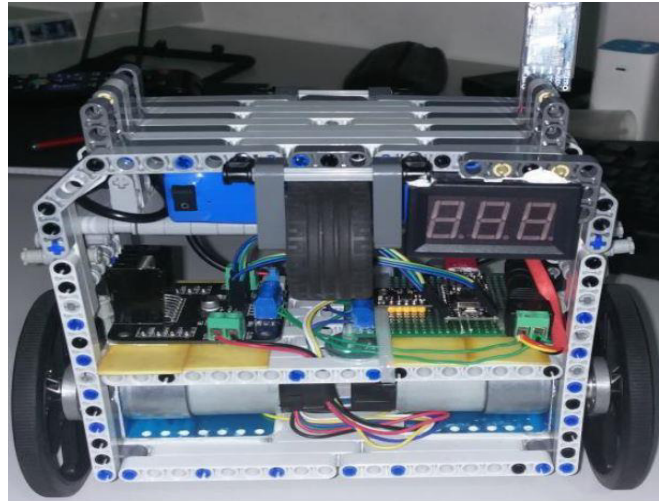


Fig. 2.3 Bimbo developed by Martins and Nunes [4]

After testing different kinds of controllers on Bimbo, they concluded that pole placement method has poor performance, and PID with position control shows good performance.

Junfeng, W. and Wanying, Z. [5] worked out mathematical models based on LQR and pole placement. After simulating in MATLAB, they concluded that LQR controller has better performance compared to pole placement.

Feng, T., Wang, X., Liu, T., Xu, Z., Zhang, M. and Han, S. C. [6] fabricated a robot with 3 feedback sensors: an accelerometer for tilt angle, a gyroscope for angular movement and 2 encoders for position & velocity of robot base. They used Kalman filter for sensor fusion and pole placement controller for balancing. The robot was stable and robust for a tilt angle of  $\pm 3^\circ$  and was on exhibition for months.

Wu, J., Liang, Y., and Wang, Z. [7] derived a mathematical model for two-wheeled self-balancing robot and proposed a robust control system based on SMC (Sliding Mode Control) and simulated the model to examine performance.

Ferdinando, H., Khoswanto, H., and Tjokro, S. [8] designed the robot which had its chassis constructed with Lego Mindstorm, an accelerometer was used for feedback, and AVR ATmega16 microcontroller was used to control it. They also

mounted load under the axis to mechanically stabilize it. After experimentation, they concluded that PD controller gives best performance.

Juang, H. S., and Lum, K. Y. [9] constructed a robot using 2 DC motors, an accelerometer and a gyroscope with a complementary filter. They studied both PID and LQR based PI-PD control designs, and concluded that PI-PD control system gives much better stability than PID.

Sun, C., Lu, T., and Yuan, K. [10] proposed a new control method which was a combination of LQR and NN (Neural Network), both compensating each other's shortcomings. They concluded that LQR-NN controller had improved system's robustness, but the stabilization time didn't improve compared to other control systems.

An, W., and Li, Y. [11] derived mathematical models for PID and LQR control systems, simulated them in MATLAB, and concluded that LQR had better balancing performance than PID.

Jamil, O., Jamil, M., Ayaz, Y., and Ahmad, K. [12] developed a mathematical model with PID and LQR control systems, simulated them in MATLAB and Simulink, and concluded that PID controller is sufficient if robot has to only balance itself vertically and not move around; but if we want a robot which can balance itself vertically as well as move around, then we better go for LQR for best results.

Prasetio, B. H. [13] designed a model with Ensemble Kalman filter (EnKF) and PID controller. He manually tuned the covariance filter and tested the system on software to find out optimal PID parameters.

Prakash, K., and Thomas, K. [14] derived mathematical models for PID, LQR and LQG (Linear Quadratic Gaussian) and compared them by simulating them in MATLAB. They concluded that LQG controller shows best performance among all three controllers.

Ali, M. I., and Hossen, M. M. [15] derived mathematical model of a two-wheeled robot using MPU6050 and two types of filter, one was Kalman filter and the other was complementary filter, and tested it on software for P, PI and PID control systems. They concluded that PID controller gives the best stability results. Also, they tested battery positions at different locations of robot, and concluded that more stability is achieved when the battery is positioned near shafts of wheels.

# CHAPTER 3

## Hardware and Software

### 3.1 Project Conceptualization

Fig. 3.1 shows flow chart of schematic process of working of the project. The project starts with the conceptualizing and making basic idea of the working of project. Next step is designing and fabrication of robot hardware by selecting all the necessary electronic components. This process includes building complete frame of the robot and embedding components over it with care by keeping its center of mass in mind. To get and maintain center of mass of robot is very important part in making of robot because set point or offset values are affected by it. This step is followed by software algorithm implementation and its integration with the hardware. Since we are using manual method or hit and trial method for robot balancing, finally robot will be tested for different control gain parameters namely  $K_p$ ,  $K_i$  and  $K_d$  until we achieve our desired objective. After getting a balanced stationary robot, we will try to operate it wirelessly with the remote control that uses low radio frequency band, and then it will be able to move itself in forward, backward, right and left direction as instructed by the remote control.

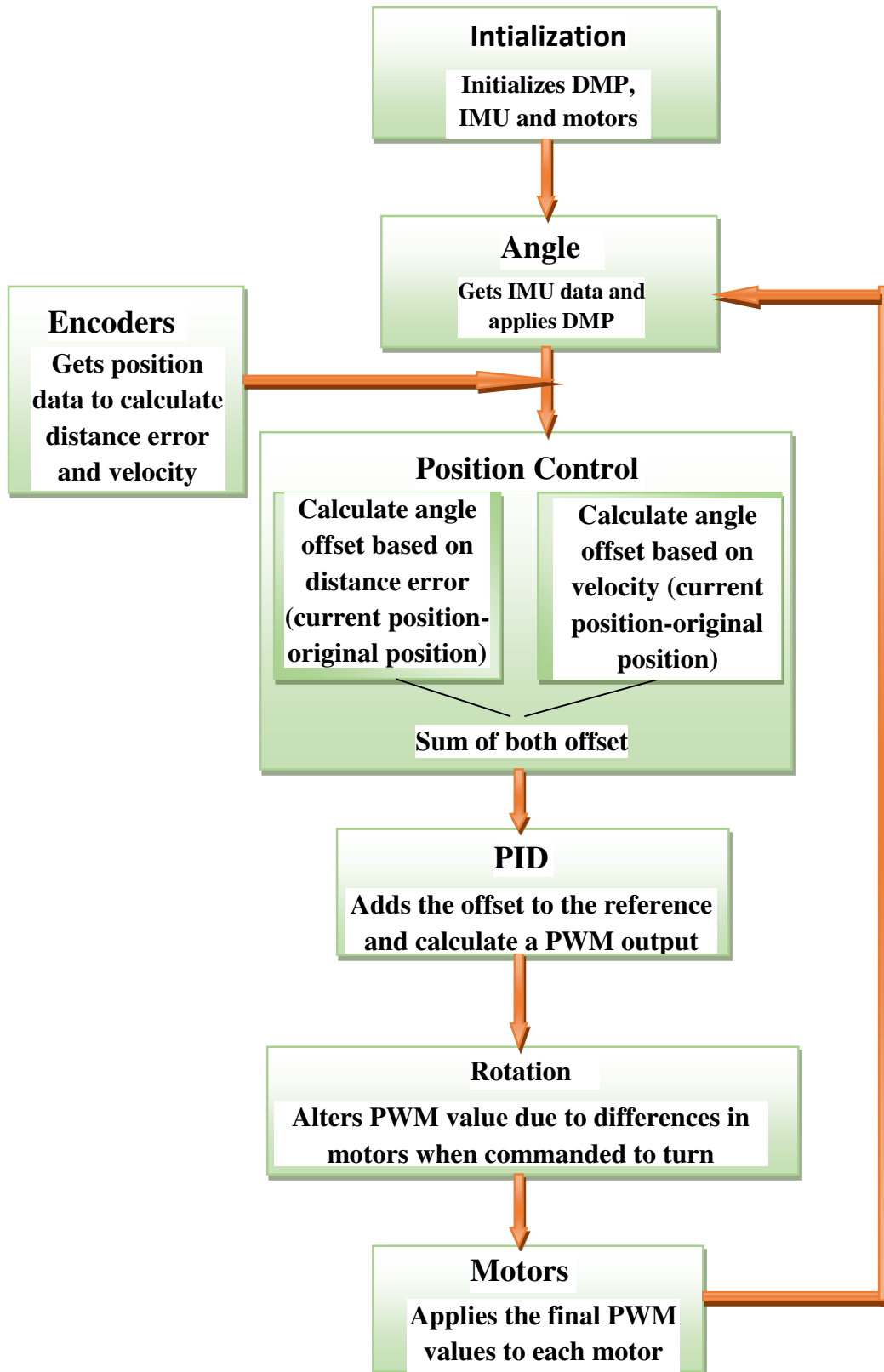


Fig. 3.1 Flow chart of the project conceptualization

## 3.2 Hardware development

The hardware system designing is the building block of any project. Table 3.1 shows the list of components used in the robot. To design a robot we need a microcontroller that acts as the brain of the robot where the required code can be written to control all the electronics components of the robot so that they can work in a synchronized way to achieve the goal. To fulfill microcontroller's need here we selected two Arduino boards, one for the robot and another for the remote control, now to for balancing purpose we need to find the angle of tilt, displacement and velocity of the robot. For this task we need an IMU sensor, so I have selected MPU6050 as IMU sensor which takes these readings from the device.

Since it is a two wheeled robot, so obviously we will need two well griped wheels and to make rotate these wheels, motors will be needed. Here we have taken two stepper motors because it is very precise and helpful in balancing activities. Now to control the direction and speed of motor we need a motor controller that can guide the motors about the direction and speed of movement. Since we have taken two stepper motors so we will need two stepper motor controllers and DRV8825 is perfect for this purpose that comes with the heatsink to prevent it from any damage due to heat.

To supply power to the robot circuitry we need a power source and I have selected a rechargeable Li-Po battery for this purpose because it is cost efficient and we can be free from again and again replacement of the battery. Now we need to direct the direction of movement to the robot, so joystick is used here which is used by in many games for giving right, left, up and down direction movements. It is quite irritating and difficult to handle long wires connected with the robot. So, to reduce wiring complexity and to handle the robot even from a distance away from it, it is better to use wireless device in remote control. I have chosen bluetooth transceiver which uses radio frequency and can able to communicate and control the robot approximately in 9m range.

Now we need a frame to give robot a shape and will act as a base where all the electronics components can be assembled to work together. To arrange the components in specious manner and for proper center of mass we shaped the robot in three layered device, so we have used here three acrylic sheets which is able to handle the weight of all components of the robot without breaking.

Table 3.1 Components required and their specifications

Sl. No.	Components	Specification	Quantity
1	Frame	Made of white acrylic sheets	3
2	Wheel	Small circular wheel	2
3	Motor	DC stepper motor (NEMA 17)	2
4	Motor controller	DRV8825 with heat sink	2
5	Microcontroller	Arduino UNO	1
6	Microcontroller	Arduino Nano	1
7	Bluetooth device	HC05 (10m range)	2
8	Joystick	Wii nunchuk	1
9	Sensor	IMU (MPU6050)	1
10	Power source	Lithium polymer battery	1

The complete hardware has two parts:

A) Self-balancing robot

In this section the IMU is used to get the digital measurements of the angular data and the acceleration of the robot and this raw output is now processed to obtain the tilt angle of the robot. This tilt angle is then provided into the PID controller algorithm to obtain the appropriate speed of the stepper motor in order to balance the robot. There is also a bluetooth device connected in the robot structure. This bluetooth device is actually customized as a slave device whose functionality is to receive instructions from master bluetooth connected in remote control part. This bluetooth device provides the data received from remote control to the Arduino UNO controller and according to that data Arduino UNO decides the direction of wheels, in which we want to make it move. Figure 3.2 shows the block diagram of electronics system of self-balancing robo

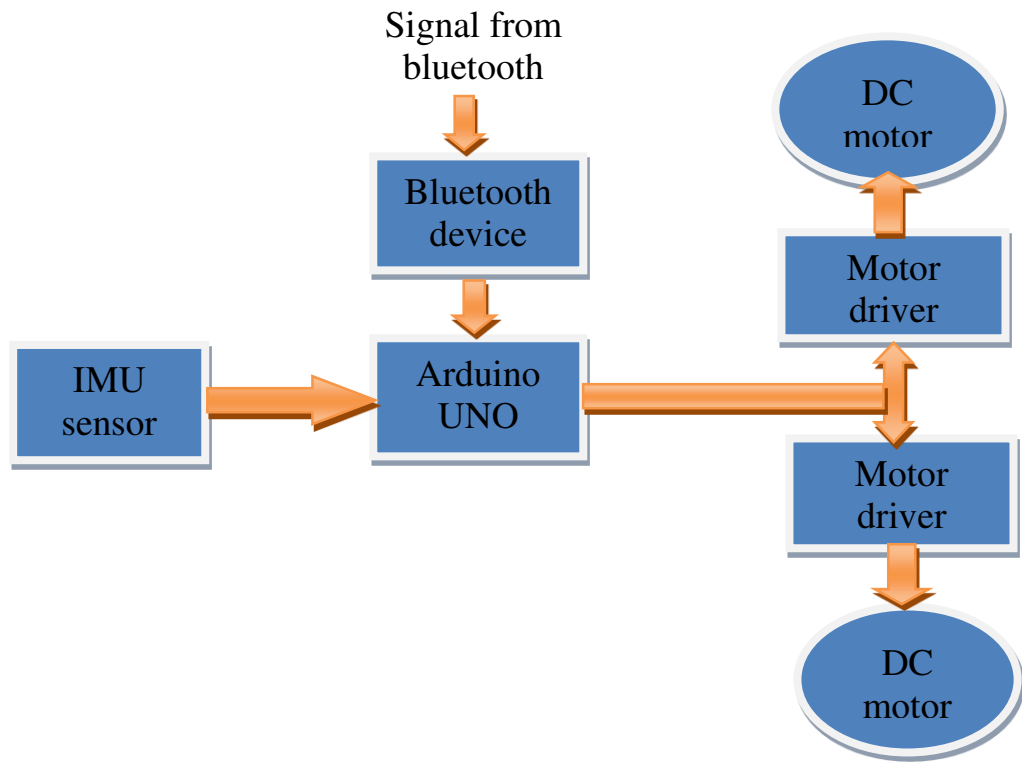


Fig. 3.2 Block diagram of self-balancing robot

### B) Remote control

In the remote control part, a joystick uses x and y axes to give the instructions to move the robot in forward, backward, right and left directions. The data from joystick is fed into Arduino NANO which further provides this data into the master bluetooth device. This master bluetooth is paired with the bluetooth device connected with Arduino UNO of the robot. The master bluetooth sends the direction instructions or data to the paired slave device wirelessly. Figure 3.3 shows the block diagram of remote control.

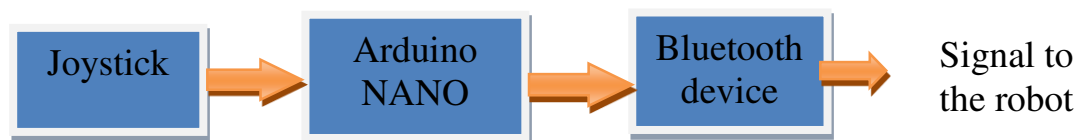


Fig. 3.3 Block diagram of remote control

### 3.2.1 Arduino UNO

The main microcontroller used to balance the robot is Arduino UNO. It can be considered as the brain of the SBR. There are many microcontrollers like PIC series, AVR series, 8051, raspberry, etc. available in the market, but I have selected the Arduino because it comes with a complete package which includes 5V regulator, an oscillator, a microcontroller, serial communication interface, LED, a burner and headers for the connection, which makes it very easy in use and very simple to assemble components with it. It is an open source prototyping platform i.e. the board can be easily modified and optimized for better functionality. It easily scales between different members of the family. Its software has many available libraries and functions which makes it easy to do its coding even for beginners. Fig. 3.4 shows an Arduino UNO.

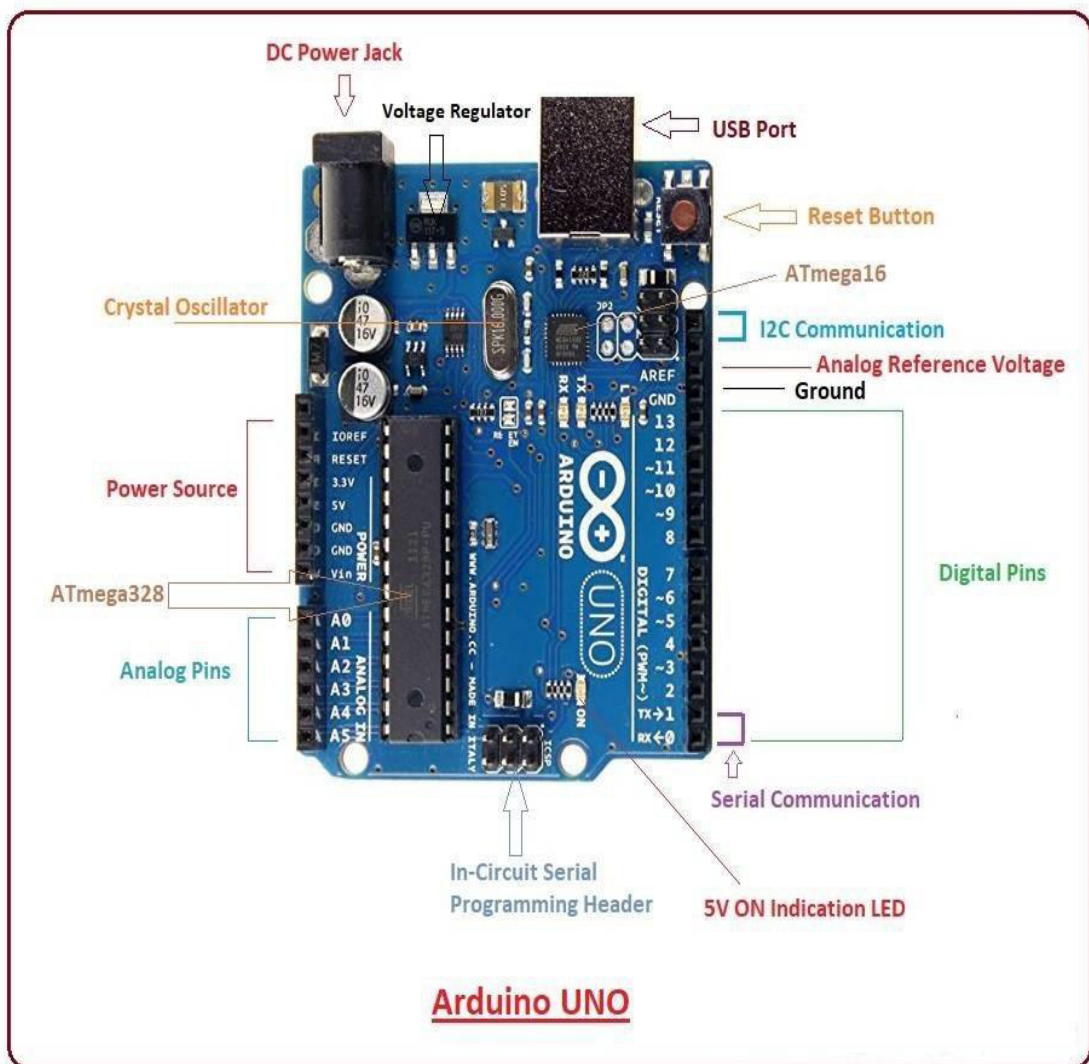


Fig. 3.4 Arduino UNO [20]



The Arduino UNO is mainly based on AVR microcontroller Atmega328 and it comes with 6 analog input pins, USB interface and 14 I/O digital ports out of which 6 pins can be used for PWM output. These I/O digital pins can be used to connect it with external electronics circuits. In the SBR it is connected with the IMU sensor to get the tilt angle information and after processing the data it communicates with the motor drivers with the instructions of speed and direction of the motor adjustments. Its specifications are tabulated under table 3.2.

Table 3.2 Specification of Arduino UNO

Sl. No.	Parameters	Specifications
1	Microcontroller	ATmega328
2	Input voltage (limit)	6-20V
3	Input voltage (recommended)	7-12V
4	Operating voltage	5V
5	Analog Input Pins	6
6	Digital I/O Pins	14
7	DC Current for 3.3V Pin	50 mA
8	DC Current per I/O Pin	40 mA
9	SRAM	2 KB (ATmega 328)
10	EEPROM	1 KB (ATmega 328)
11	Flash Memory	32 KB (ATmega 328) out of which 0.5 KB used by bootloader
12	Clock Speed	16 MHz
13	Width	68.6 mm
14	Length	53.4 mm
15	Weight	25 g

### 3.2.2 Stepper motor

Fig. 3.5 shows the view of stepper motor which is used as an actuator in the self-balancing robot. There are different motors like DC geared motor, servo motor, etc. are available for use in the market but I preferred to use stepper motor in the project because it is a better choice whenever we need any controlled movement. It is precise

and has no any performance loss when the battery voltage drops. There is no lag in performance due to moment of inertia of the motor. Stepper motor is widely used in robotics and in other fields where precise and accurate movement is needed.

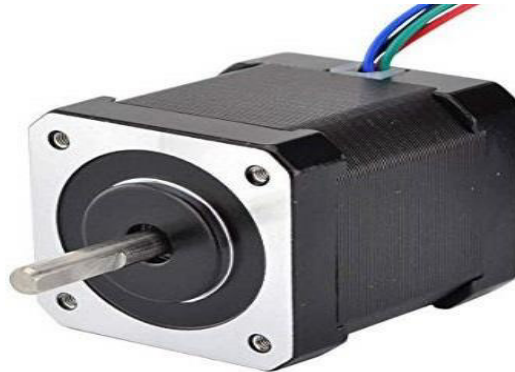


Fig. 3.5 DC stepper motor (NEMA-17) [37]

The motor's rotation has a direct relationship with the applied input pulses. I have used NEMA 17-size hybrid bipolar stepper motor [21]. This motor has 1.8° step angle which means 200 steps per revolution. It has two coils and there are two wires connected with each coil. It needs a motor driver module to control it. Its specifications are tabulated under table 3.3.

Table 3.3 Specifications of DC stepper motor

Sl. No.	Parameters	Specifications
1	Type	Stepper motor
2	Phase	2
3	Step angle	1.8°
4	Motor length	34mm
5	Voltage	12V
6	Rated current	0.3A
7	Rated power	5W
8	Holding torque	2.2kg.cm

### 3.2.3 Motor controller

Motor controller is an essential module for motor to make it run in different directions and different speeds. In my robot I have used a micro stepping motor driver DRV8825. It is very compact in size and easy to use. The reason behind its selection

was its best performance with the NEMA-17 stepper motor. It is cooler than other stepper motor controller like A4988 and also very cost efficient. It has only 2 pins for controlling steps and spinning direction of a motor. Fig. 3.6 shows an image of DRV8825 motor driver module.

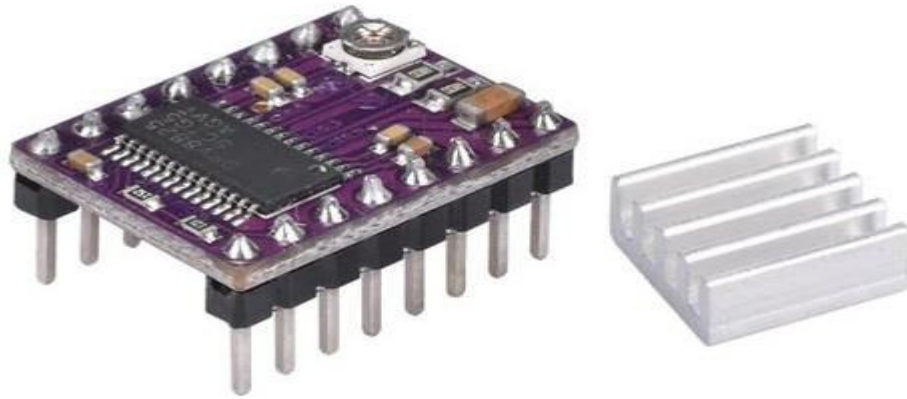


Fig. 3.6 The view of DRV8825 and heatsink [22]

This driver can be operated in six different step resolutions e.g. full-step, 1/2-step, 1/4-step, 1/8-step, 1/16-step and 1/32-step by setting proper logic levels to its pins M0, M1 and M2. It can supply power for motor in the range of 8.2V-45V and deliver current up to maximum 2.2A but approximately 1.5A per coil without getting overheated. A heat sink or any other cooling method is required to prevent it from damage caused by rise in temperature due to excessive power dissipation of the DRV8825 driver module.

### 3.2.4 IMU Sensor

IMU (Inertia Measurement Unit) sensor is an electronics device which is used to measure velocity, acceleration, orientation, displacement and other motion features. There are a number of IMU sensors like ITG 3200 gyroscope, ADXL 345 accelerometer etc. available in the market but I have selected to use MPU6050 it consists of both gyroscope and accelerometer on a single chip. The MPU-6050 is the most accurate, cheaper and reliable IMU sensor. It is MEMS (Micro Electro-Mechanical System), which consists of 3-axis gyroscope, 3-axis accelerometer and DMP (Digital Motion Processor) embedded inside a single chip. MPU6050 consists a 16-bit ADC hardware, due to which it is able to capture x, y and z axes motion at the same time with high accuracy. It operates on I2C communication protocol to interface with an Arduino.

The MPU6050 sensor also contains a 1024 byte FIFO buffer, where sensor values can be placed and read by the Arduino. This buffer is used in combination with the interrupt signal so that Arduino knows when to read data from the sensor module. There is an INT (interrupt pin) in the MPU6050 module which provides interrupt or instructions to the Arduino only when the sensor values changes. Fig. 3.7 shows the view of IMU sensor (MPU6050), and its properties are tabulated under table 3.4.

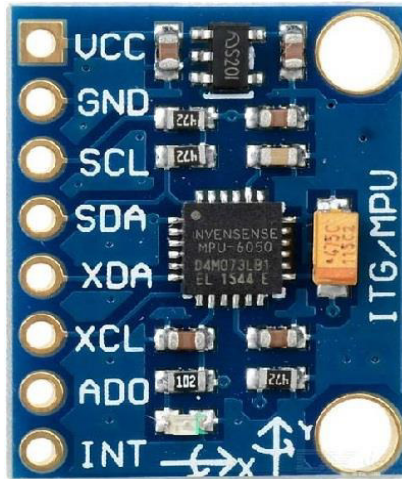


Fig. 3.7 IMU sensor (MPU6050) [38]

Table 3.4 Specifications of MPU6050

Sl. No.	Parameters	Specification
1	Mounting type	Surface mount
2	Power supply	3-5 V
3	DOF	3(yaw, pitch, roll)
4	Sensor	3 axis gyro and accelerometer sensor
5	DMP	Inbuilt
6	Operating temperature	40°-150° C
7	FIFO	1024 bytes
8	Operating current	3.9 mA
9	Communication protocol	I2C protocol
10	ADC	16-bit

### 3.2.4.1 Three axis MEMS accelerometer

The accelerometer works on piezoelectric effect principle [30]. It can be considered as a cuboidal box made of piezoelectric crystals in which a ball is forced to move by tilting the box. We can determine the inclination direction and magnitude depending on the current produced from piezoelectric walls. The accelerometer measures the acceleration in 3-dimensions but it gives quite noisy and unstable data.

### 3.2.4.2 Three axis MEMS gyroscope

The MPU6050 contains three axis gyroscope which works on the principle of Coriolis Effect and senses the angular rotation for x, y and z axes. These signals are amplified, demodulated and filtered to give output voltage proportional to the angular rate. This output voltage is again digitalized and displayed as angular deviations.

### 3.2.4.3 DMP Processor

DMP filter is inbuilt available in MPU6050 which filters the data from the gyroscope and accelerometer. It fuses both data in order to minimize each sensor inherent error effects and gives a desired angle of inclination as output.

## 3.2.5 Bluetooth module

Bluetooth module is used for wireless communication for sending and receiving data in the form of radio waves. Bluetooth has many advantages over other wireless communications like Wi-Fi. It has low power consumption and has better range than infrared communication. It is cheaper and data communication is much secure. Fig. 3.8 shows the view of bluetooth module.

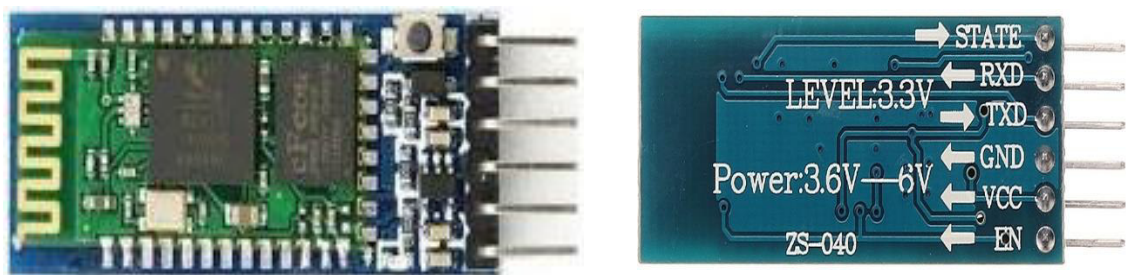


Fig. 3.8 Bluetooth module (HC05) [39]

HC05 is a bluetooth module which can be used as transmitter or receiver depending on master or slave configuration [24]. It uses serial port (USART) for serial wireless communication with the Arduino. The role of the module i.e. master or slave can be configured by using AT commands. It has a red LED for indicating its connection status, whether it is connected or not. Its properties are tabulated under table 3.5.

Table 3.5 Bluetooth specifications

Sl. No.	Parameters	Specification
1	Bluetooth module	HC05
2	Operating voltage	3.3-5V
3	RF transmit power	+4 dBm
4	Default command baud rate	38400 bps
5	Data bits	8
6	Default communication	slave
7	Default mode	data
8	Default data baud rate	9600 bps
9	Maximum range	9 meters
10	interface	USART

### 3.2.6 Arduino Nano

The Arduino Nano is a complete, bread-board friendly and compact board based on Atmega328 processor [25]. Fig. 3.9 shows the view of the Arduino Nano board.

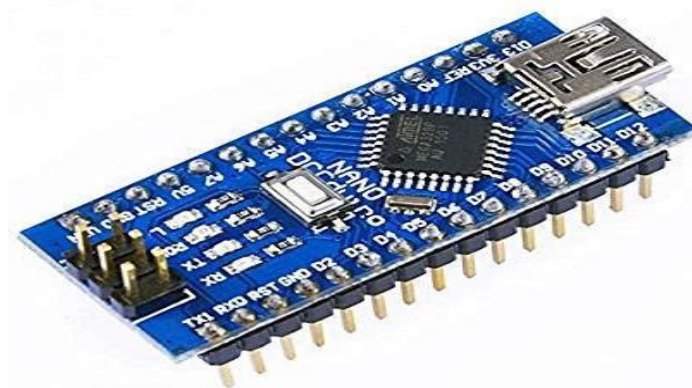


Fig. 3.9 Arduino Nano [36]

Its functionality is similar as the Arduino UNO but in different package. It uses mini USB for power supply and code uploading. It has no power plug for external power supply. It is smaller in size and less in weight than the Arduino UNO. The code is exactly same for both the boards.

### 3.2.7 Joystick

Joystick, also known as control column, is an input device which consists a stick that pivots on a base and guides its angle or direction to the device being controlled by it. The Wii nunchuk is a kind of joystick which is used mostly as game controllers [26]. It uses x and y axes to read data and instructs accordingly that data to move a being controlled device or paired device in a particular direction with desired specific angles. Fig. 3.10 shows an image of Wii nunchuk used in the self-balancing robot.



Fig. 3.10 The view of Wii nunchuk [34]

Fig. 3.11 shows pin diagram of Wii nunchuk. It has 6 pins, out of which two pins are not used by users and the remaining 4 pins are for power supply, ground connection, data transfer and to provide clock pulses.

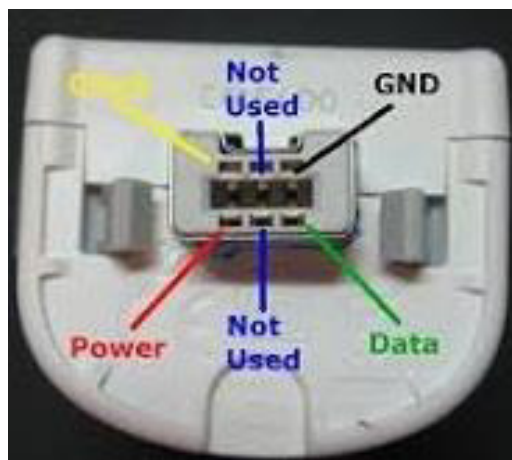


Fig. 3.11 Pin diagram of Wii nunchuk [35]



### 3.2.8 Power source

I have used the ORANGE 11.1V/18800mAh 3S 30C Li-Po (Lithium Polymer) battery pack with XT60 connector. It is rechargeable and designed with heavy duty discharge leads to reduce resistance and maintain high current loads. These are assembled with IR matched cells. It has very good temperature control in spite of high discharge rate. It provides sufficient amount of power supply to run the robot. Fig. 3.12 shows an image of 3 cell 1800 mAh Li-Po battery, and table 3.6 tabulates its specifications.



Fig. 3.12 An image of 1800mAh Li-Po battery [27]

Table 3.6 Specification of battery

Sl. No.	Parameters	Specification
1	Capacity	1800 mAh
2	Output voltage	11.1 V
3	Weight	141g
4	Discharge plug	XT-60
5	Balance plug	JST-XH
6	Size	105 x 34 x 21 mm
7	Charge rate	1-3C

### 3.2.9 Frame

The main point behind selection of the frame of the robot was its availability, versatility and sufficient strength to handle all components' weight used i



the robot. Other frames like plywood, cardboard, etc. are not sufficient to handle the weight of all components used here and also there can be seen some bends or change in size of holes when it falls while doing some trial and error experiments on the robot. So, acrylic sheet is the better option than other frames. I have used white acrylic sheet here. Fig.3.13 shows the view of the robot frame.

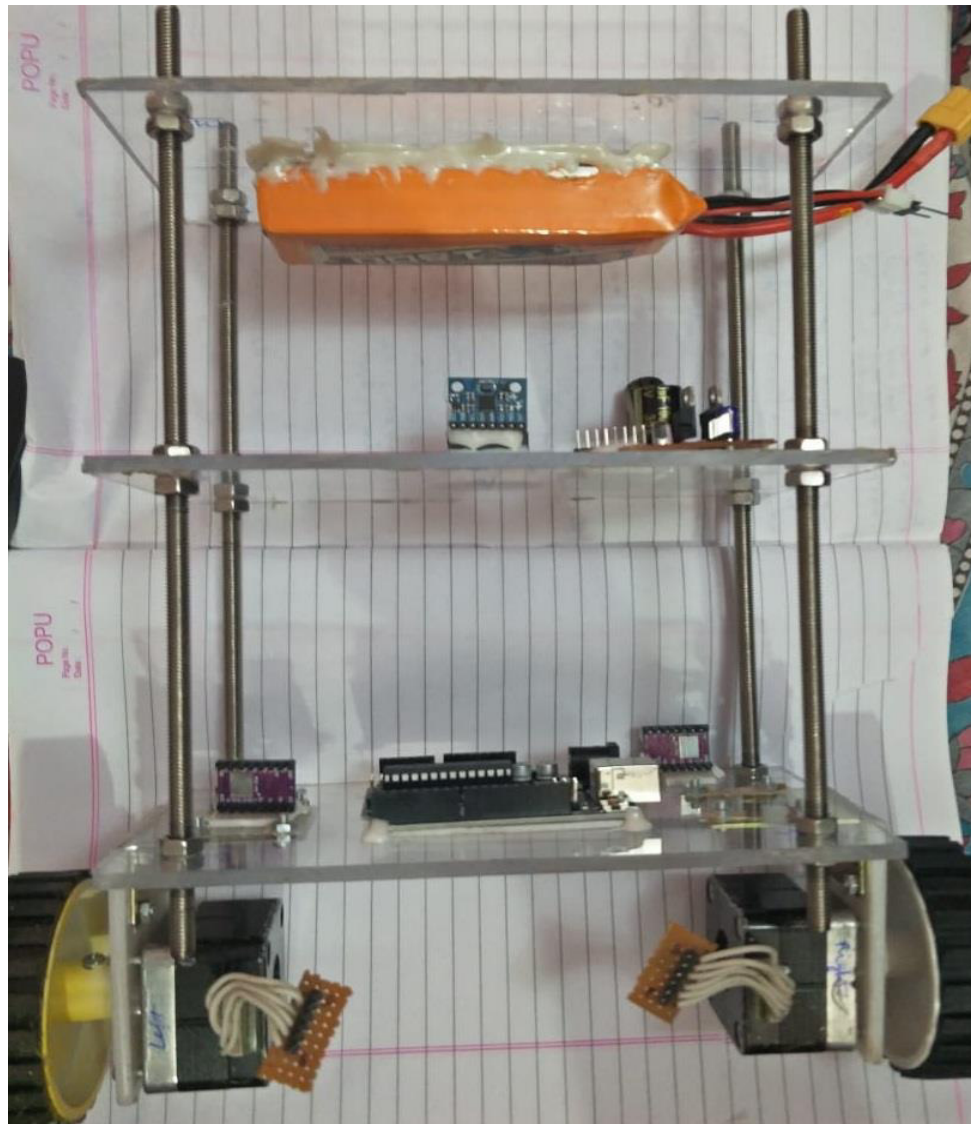


Fig.3.13 View of the robot frame

The advantage of using acrylic sheet is it is easy to cut, drill, install and very cost effective due to having less chance of breakage. Since I have done many hit and trials in my robot to achieve a proper balanced robot so it was essential that the frame needs to be more resistant to breakage. I have used three white acrylic sheets and shaped them in 3 different layers by placing them one above the other to achieve a height of approximately one foot. The size of each acrylic sheet is 18x10 cm. There are two

separate small pieces of acrylic sheets used here to give support to the stepper motors. Each layer is separated by approximately 10cm with the help of 4 iron rods (6 mm diameter and 25 cm length).

### 3.2.10 Wheel

Fig. 3.14 shows a view of wheel used in the robot. The selection of wheels is the important part in development of a proper balanced robot. The wheels should have a good grip over a floor which we are using so that it can efficiently stand with the weight of robot and can reduce the chance of falling down of the robot. I have used two wheels of 6.5cm diameter (with tyre its diameter is 7cm). There is a rubber tyre on the outer circumference of wheel which helps it to get good grip. It was preferred to use moderate size wheel in order to get satisfactory center of mass of the robot and also to properly notice the robot's balancing movement. It is very important that wheels should not skit the floor, otherwise it will be very hard to reach balanced condition of the robot.



Fig. 3.14 Wheel [40]

### 3.3 Software used

Arduino IDE (Integrated Development Environment) is an official Arduino software which is mainly used for writing, compiling and uploading the sketch or code into the Arduino modules. It is an open source software and easily available for various operating systems like Windows, MAC, Linux. It runs on the Java platform which comes with many inbuilt commands and functions that play an important role in editing, compiling and debugging the code in the environment.

Arduino IDE supports both C and C++ languages and the main code written in this software is known as a sketch. IDE generates a HEX File from the sketch and transfer and upload it into the controller of the Arduino board. The IDE mainly consists of two basic parts: Editor which is used for writing the required code and Compiler which is used for compiling and uploading the sketch into the given Arduino device.

The Arduino IDE environment can be mainly divided into three sections:

1. Menu bar
2. Text editor
3. Output pane

Menu bar has five different options i.e. file, edit, sketch, tools and help, and six buttons which are verify, upload, open, new, serial monitor and save. These are used in creating a new file, writing, editing, saving, verifying, compiling and uploading a sketch. Text editor is used to write required sketch and output pane displays the compilation status of the running code i.e. the errors occurred in the sketch and the memory used by the program. It is necessary to fix those errors before burning the hex file into the given Arduino device.

There are many preinstalled libraries provided by Arduino IDE software which adds much extra functionality into the Arduino device. The commands used in the sketch writing in IDE environment are case sensitive. Fig. 3.15 shows a view of Arduino IDE software when we open it after installing. There is also a separate pop-up window that plays an important role in sending and receiving the serial data. It acts as an independent terminal which actually debugs the written sketch and helps us to know how the program is operating.

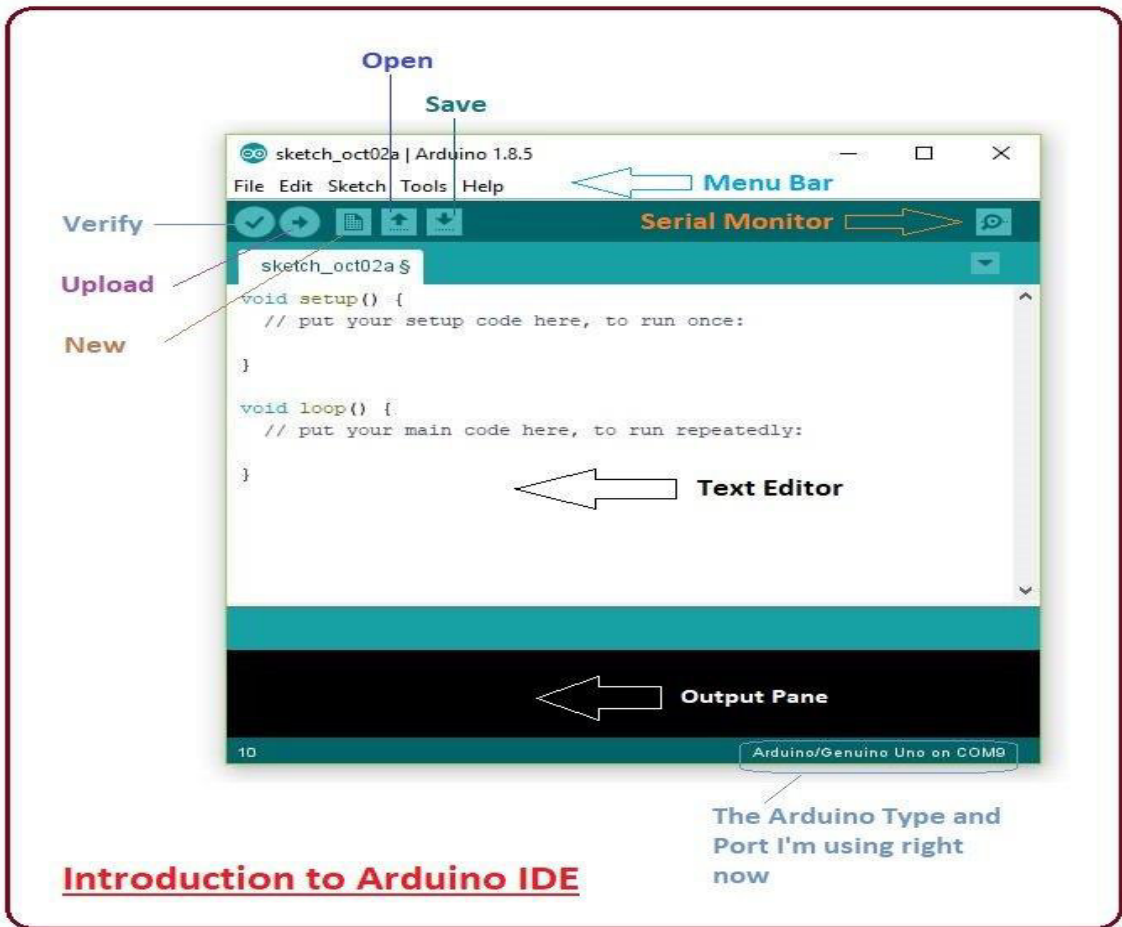


Fig. 3.15 A view of the Arduino IDE software

# CHAPTER 4

## Kinematics and Control Basics

### 4.1 Basics of stability

A two-wheeled robot is basically an inverted pendulum system [33] on a movable cart, with Centre of mass located above its pivot point. Since the pendulum is constrained for movement along sideways, but is free to tilt in forward and backward direction, the system is in a naturally unstable state, with the pendulum having tendency to oscillate and fall down under the influence of gravity or the slightest torque applied on pivot point.

An inverted pendulum system consists of a pendulum of mass  $m$  and moment of inertia  $I$  hinged on a movable cart of mass  $M$  with its center of mass is at a distance  $l$  from hinged point. The pendulum falls with a tilt angle  $\Theta$  from vertical. In order to balance this fall, a force  $F$  has to be applied on cart.

For better understanding of the system and the forces acting on it, fig 4.1 and 4.2 shows inverted pendulum system and its free body diagram respectively. Also, the parameters of the system are tabled in table 4.1.

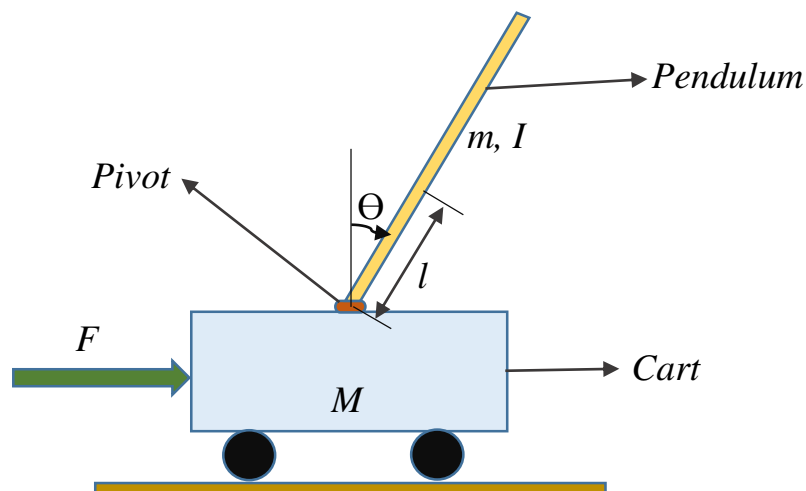


Fig. 4.1 Inverted pendulum

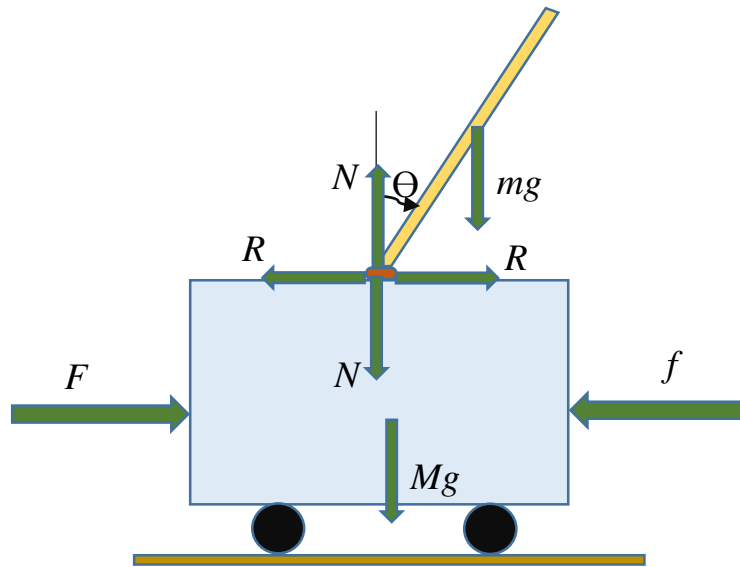


Fig. 4.2 FBD of inverted pendulum

Table 4.1 Parameters of inverted pendulum system

M	Mass of the cart	kg
m	Mass of the pendulum	kg
l	Distance between pivot point and center of mass of pendulum	m
I	Mass moment of inertia of pendulum	kg.m <sup>2</sup>
F	Force applied to the cart	kg.m/s <sup>2</sup>
f	Frictional force on cart	kg.m/s <sup>2</sup>
$\theta$	Angle of tilt	in degree
g	Acceleration due to gravity	9.8 m/s <sup>2</sup>

As the pendulum tries to fall in a direction, the applied force **F** makes the cart move in the opposite direction. It gives upward reaction to the pendulum, which counters the falling motion and stabilizes the system. Now, in order to keep the system stable for a long duration, we have to keep changing the magnitude and direction of force **F** according to the speed of cart, the angle of tilt, the frequency of falling and the direction of fall of pendulum. This is why an embedded control system is used so that with help of sensors and control units, the system can be balanced in a fast and responsive manner.

## 4.2 Types of controllers used

Balancing of robot requires application of closed-loop (Feedback) control systems. Researchers have used different kinds of control methods to study balancing of robots. Some of these controllers/methods are as follows:

- PID (Proportional, Integral, Derivative)
- LQR (Linear Quadratic Regulator)
- LQG (Linear Quadratic Gaussian)
- FSF (Full State Feedback), or, Pole Placement
- Adaptive control
- SMC (Sliding Mode Control)
- Intelligent controls like NN (Neural Network) and Fuzzy logic.

Each of these methods have their own advantages and shortcomings in terms of precision, responsiveness and reliability. So, researchers try to combine two or more control systems so that they can compensate each other's disadvantages and we can get a better performance. We have used PID controller in our project.

## 4.3 PID Controller

### 4.3.1 Overview

The control algorithm used here to control or balance the self-balancing two wheeled robot is a PID controller (Proportional Integral Derivative controller). It is also known as a three term controller. It is a closed loop feedback mechanism which is widely used in the industrial control applications. This controller continuously calculates an error value by comparing the actual value and the desired value of output and because of this it is mostly used in the applications which need continuously monitored control. Fig. 4.3 explains the error as difference between set point and actual tilt angle of the robot. This controller must be executed frequently enough with the controllable range of the system.

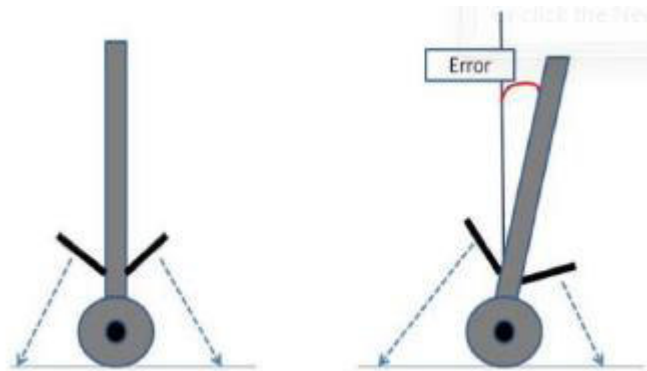


Fig. 4.3 Set point and actual tilt angle of the robot [41]

PID uses three basic control mechanisms for error correction which are explained below:

**P-controller:** P- controller or proportional controller provides output in proportional to the current error  $e$ . It calculated the error as a difference between set point and feedback process value and multiplies that error with a proportional constant ( $K_p$ ) to achieve the output. This controller doesn't reach to the steady state condition so it needs a manual or biasing reset when used alone. When the proportional constant  $K_p$  increases the speed of response also increases. In this controller output can be expressed as

$$\text{Output} = K_p * \text{Error}$$

Where, Error = Actual or measured value - Desired or set point value

**I-controller:** This controller integrates the error over a period of time until error value reaches to zero and multiplies it with integral gain constant  $K_i$  to get the output. It almost eliminates the steady state error. The speed of response increases with decrease in integral gain  $K_i$ . The output of this controller can be expressed as

$$\text{Output} = K_i * (\text{Summation of Error})$$

**D-controller:** Derivative controller is also called anticipatory controller since it predicts future behavior of the error. The output of this controller is given by the rate of change of error with respect to the time, multiplied by the derivative gain constant ( $K_d$ ), i.e.,

$$\text{Output} = K_d * (\text{Error} - \text{Previous Error})$$



The overall output of PID controller can be expressed as a combination of proportional, integral and derivative controller responses, i.e.,

Output PID controller = Output Proportional Term + Output Integral Term + Output Differential Term

Or, Output of PID controller =  $K_p * \text{Error} + K_i * (\text{Summation of Error}) + K_d * (\text{Error} - \text{Previous Error})$

Or, Output of PID controller can be expressed mathematically as

$$u(t) = K_p * e(t) + K_i \int e(t) dt + K_d * de(t)/ dt \quad [31]$$

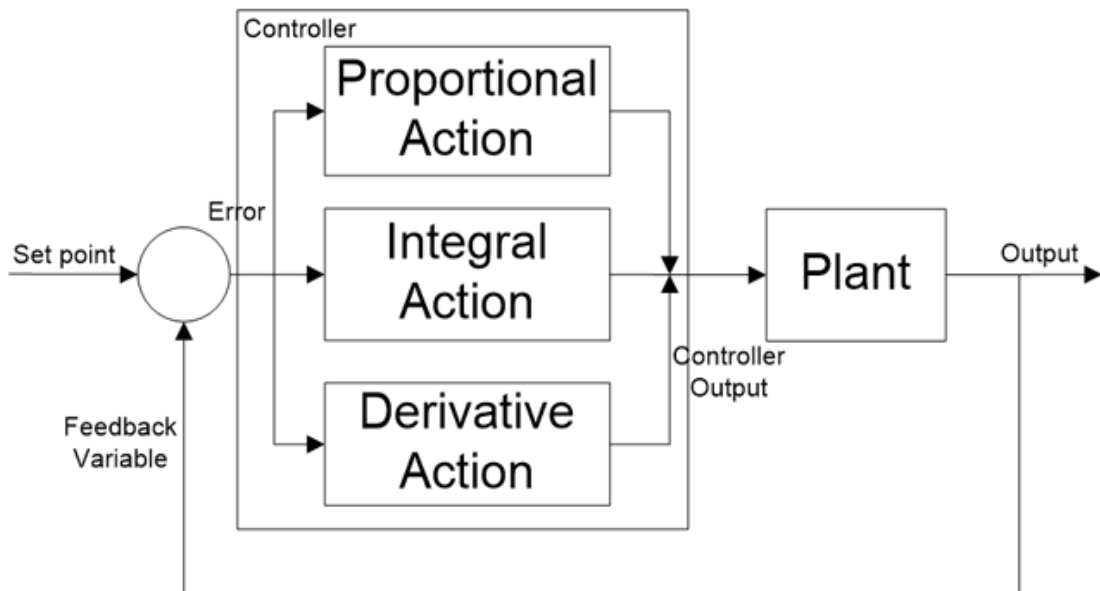


Fig. 4.4 PID Controller block diagram [28]

Fig 4.4 shows PID controller block diagram. In this self-balancing robot instantaneous angle continuously is measured by IMU (Inertial Measuring Unit), which produces digital output signals. This instantaneous angle is the actual angle of the robot and by comparing it with the desired set point, we get the error i.e. difference between the actual angle and desired set point is obtained. The error is then given into PID controller. Then the controller will process, calculate and generate the corresponding instructions to control the speed and direction of rotation of the stepper motor, in order to achieve a proper balanced robot in up right manner.

### 4.3.2 The characteristics of PID controller

The PID controller contains the combined characteristics of three controllers, proportional controller (P-controller), integral controller (I-controller) and derivative controller (D-controller). A P-controller can reduce the rise time and can reduce but unable to eliminate the steady-state error. An I-controller may have the slower transient response but it is effective in eliminating the steady-state error for a step or constant input. A D-controller is effective in reducing the overshoot, increasing the stability of the system and improving the transient response. Table 4.2 summarizes the effect of each controller parameters,  $K_p$ ,  $K_i$  and  $K_d$  on a closed loop system.

Table 4.2 Effects of PID tuning

Response	Rise Time	Settling Time	Overshoot	Steady-state Error
$K_p$	Decrease	Small Change	Increase	Decrease
$K_i$	Decrease	Increase	Increase	Eliminate
$K_d$	Small Change	Decrease	Decrease	No Change

It should be noted here that since these three control gain parameters,  $K_p$ ,  $K_d$  and  $K_i$  are dependent on each other, by changing the value of any one of these variable the effect of other two variables can also be changed.

## CHAPTER 5

### Construction Details

In this section we will discuss in detail about the connections between hardware components of the robot. This part seems easy but very important because the connections should be very proper and accurate otherwise either the robot functionality will not match with the desired objective or its components can be damaged because of wrong connections. The complete project can be divided into two separate mechanical designs namely the robot and the remote control. Fig.5.1 shows a view of the robot after wiring of components.

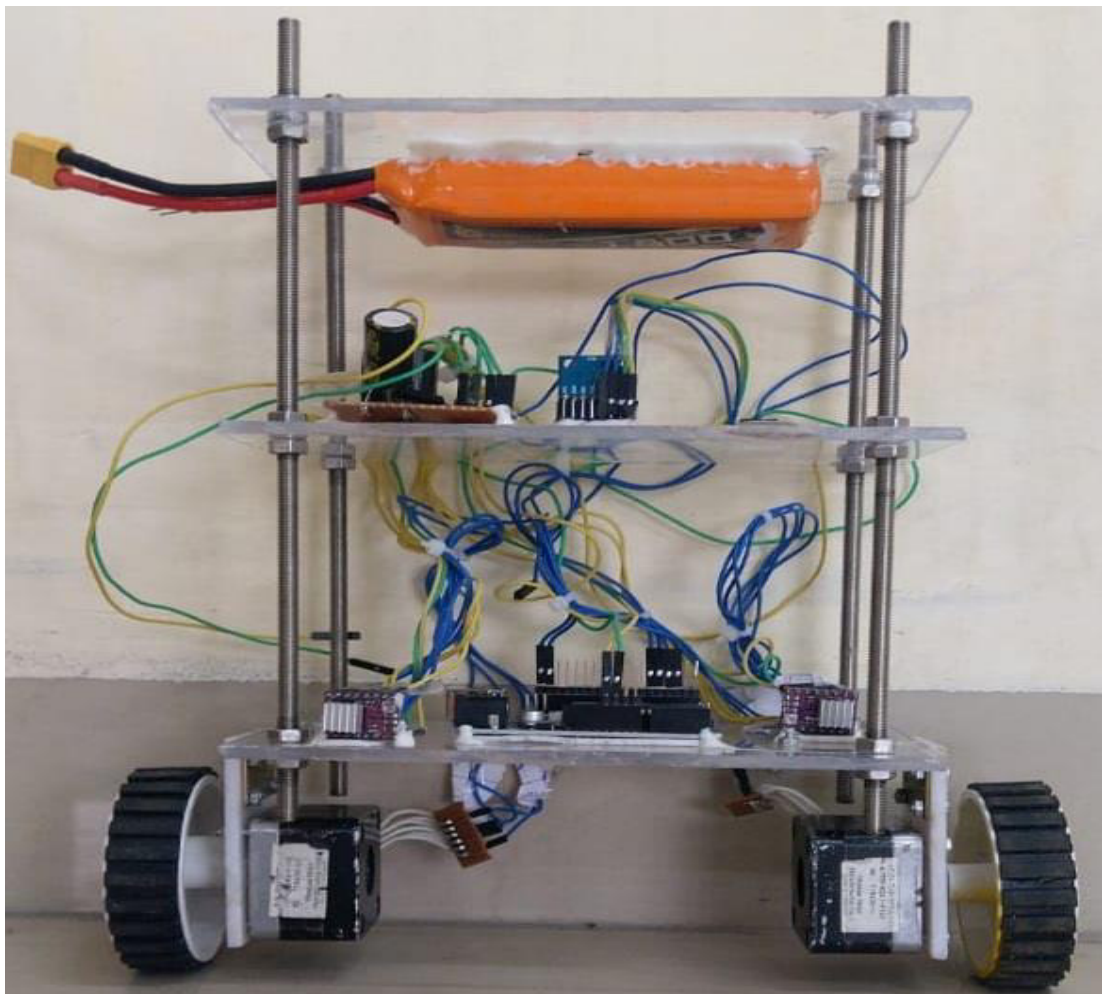


Fig. 5.1 View of the robot after wiring of components

## 5.1 Connections in robot

The power supply to robot is given here by 11.1V Li-Po battery which will be approximately 12V when fully charged. But our components require a supply of maximum 5V so we have used here a voltage regulator that converts the input voltage to 5V. It is a three terminal device. Its input terminal is connected to the positive terminal of the Li-Po battery, middle pin is connected to ground pin of the battery and also with the ground pin of all the components and output pin will be connected to Vcc pin of all electronics components. The I2C communication serial data lines SCL and SDA of IMU sensor is connected with the I2C communication serial pins A4 and A5 of the Arduino UNO. The receiver (Rx) and transmitter (Tx) pins of the Arduino board is connected with the transmitter (Tx) and receiver (Rx) pins of the bluetooth attached in robot respectively. Digital output pins 2, 3, 4 and 5 of the Arduino UNO is joined with the direction and step control pins of the stepper controllers. A1, A2, B1 and B2 pins of stepper motor controller is wired with the stepper motors to give it power supply and instruction of movement. VMOT and GND pins of motor controller are directly connected with the output power pins of the Li-Po battery. M1 pin of the DRV8825 is kept high in order to achieve 1/4<sup>th</sup> step control of the motor. And finally the motors are connected with the wheels for the movement in order to balancing the robot. Figure 5.2 shows these hardware connections between components of robot.

Two 1000uF, 35V capacitors and a 100uF, 25V capacitor are also used before the 12V and 5V power supply respectively which acts as a band filter that helps in maintaining a constant stable current through the device. A semiconductor diode is also attached between the Li-Po battery and voltage regulator so that the electronic components can be protected if accidentally the polarity of the battery gets reversed.

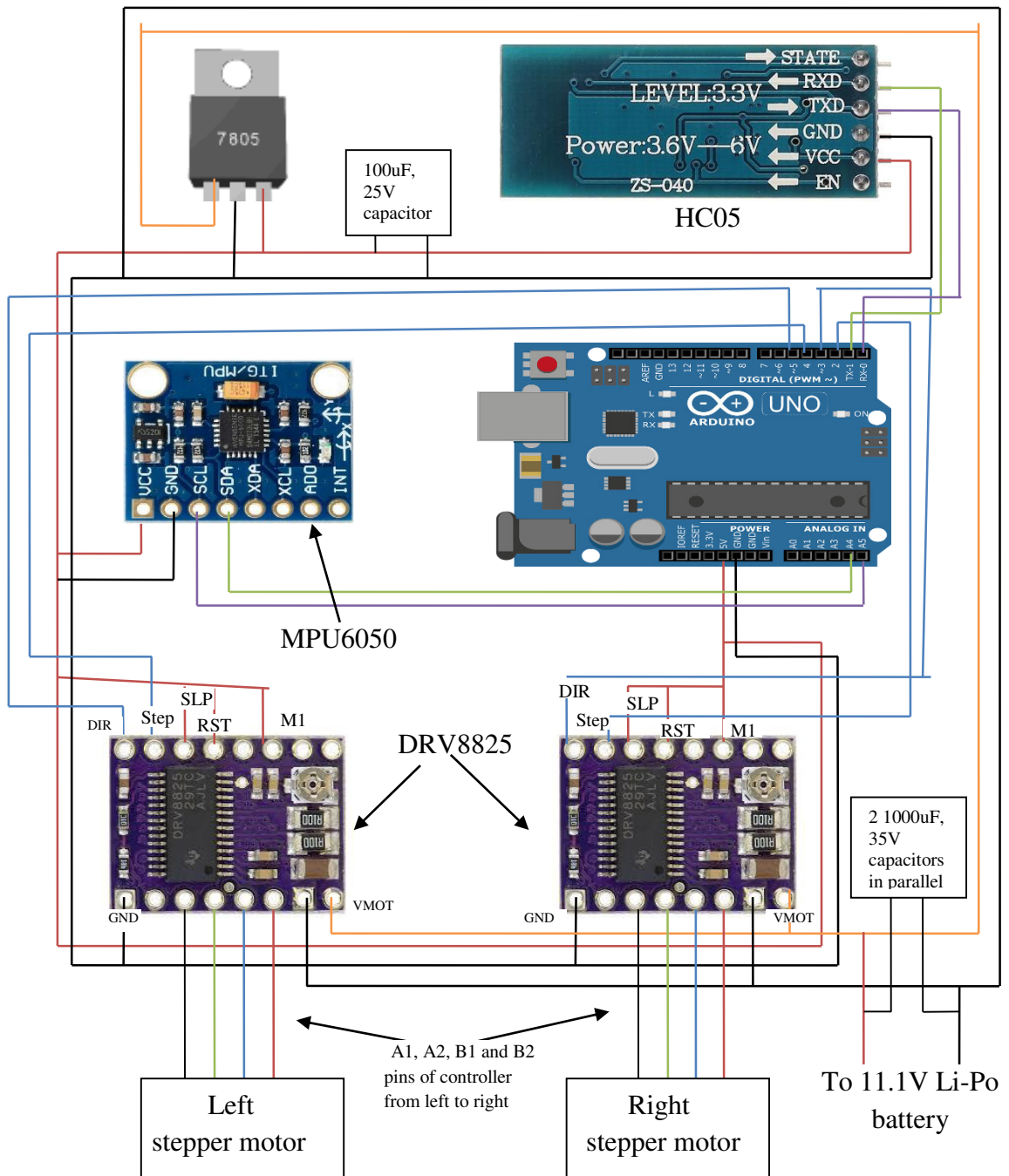


Fig. 5.2 Connection diagram for robot components

## 5.2 Wiring in remote control

The Arduino Nano board is used in remote control in order to make the remote size small so that it can be easily handled and operated. The power to this remote is given directly by the laptop through the Arduino connector. The Arduino board has inbuilt voltage regulator which make sure that the maximum supply to the board doesn't

exceed than 5V. The transmitter (Tx) and receiver (Rx) pins of the Arduino board is connected with the receiver (Rx) and transmitter (Tx) pins of the bluetooth of the remote control respectively. Power supply to bluetooth and joystick is given by Vcc and GND pins of the Arduino. Serial data lines SDA and SCL of joystick are wired with the analog serial pins A4 and A5 of the Arduino Nano in order to enable data transmission using I2C communication. Fig. 5.3 shows hardware connections between remote control components.

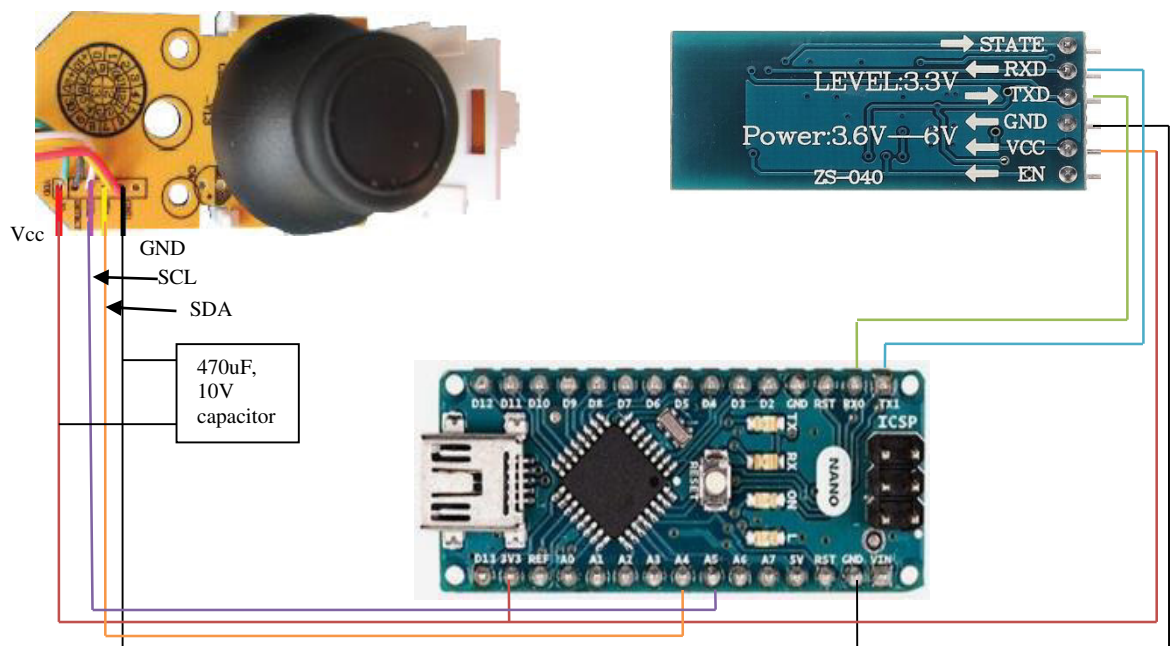


Fig. 5.3 Connection diagram for remote control components

### 5.3 I2C Interface

As shown in the fig. 5.4, Inter Integrated Circuit (I2C pronounced as I-Two-C) is a two-wired serial interface which is used to connect low speed sensors or devices like microcontroller, ADC, DAC, EEPROMs and other ICs for intra-board, short distance communication. It is synchronous, multi-slave, multi-master, packet switched, simple and easy to use interface.

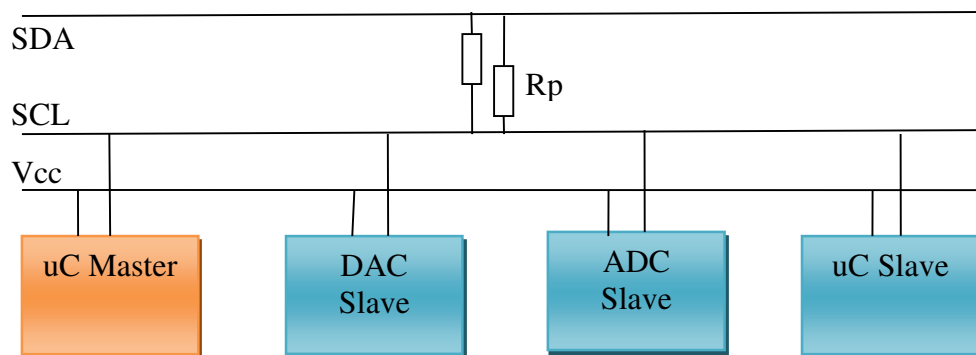


Fig. 5.4 I2C Protocol [29]

In I2C communication, master device doesn't need any address since it generates clock via SCL and addresses various individual I2C slave devices. Each I2C slave device has a 7-bit address which must need to be unique in the interface (bus). Some devices also use 10-bit address depending on their specifications. Out of these 7-bit slave addresses, address is represented by bits 7 to 1 while bit 0 is used to inform writing to or reading from to the device. If bit 0 of slave address byte is set to 1 then slave device will be read by the master device. These addresses are generally fixed for the slave device but sometimes allow it to be changed by fixing one of the pins of the slave device high or low. By this way I2C allows multiple devices on the same bus without any conflict of addresses.

I2C communication uses 8 bits or a byte for data transfer on the bus. In normal mode both SDA and SCL lines remains high. Table 5.1 shows specifications of I2C communication. Both master and slave device can send data over the bus but this is controlled by only the master device by setting the bit 0 of slave addresses to 0 or 1. The master device starts communication by generating a start condition followed by the address of the sensor or slave device. Each slave device compares this address to its own address and if it matches then slave signals an ACK bit (acknowledgement bit) by pulling data line (SDA line) low for one bit. Once master completes all read or write operations a stop condition is generated by it. This informs other devices connected with the bus that the communication has ended and other devices may use the bus, in the case if there is more than one master present on the bus.

Table 5.1 Specifications of I2C communication

Sl. No.	Parameters	Specifications
1	Transmission mode	Bidirectional, half- duplex
2	Wires used	2
3	Synchronous or asynchronous	Synchronous
4	Maximum speed	Standard mode= 100 kbps Fast mode= 400 kbps High speed mode= 3.4 Mbps Ultrafast mode= 5 Mbps
5	Maximum number of masters	Unlimited
6	Maximum number of slaves	1008
7	Serial or parallel	Serial



# CHAPTER 6

## Working and results

### 6.1 Challenges

The biggest challenge that I faced during fabrication and testing of this robot was selection of the motor. Initially I was working with the DC geared motor, but with that motor the balancing action was not accurate. In DC geared motor, there was a performance difference due to mechanical friction and electrical resistance differences. It had no precise movement, so turning the robot to the right or left with DC geared motors was difficult. I preferred stepper motor over DC geared motor because it gives exact positioning with no accumulated error. There is no lag due to the moment of inertia of the motor. It senses position simply by counting PWM signal.

Another challenge that I faced was to find a wheel with good grip. It is very important to use a good griped wheel, otherwise robot can slip over or fall on the ground even in smaller tilt angles. Even if we are tuning PID controller perfectly, we can face problem of unstable, or, not accurately balanced robot. After working with 2 different kind of wheels, finally I selected the wheel with better grip.

### 6.2 Working of the robot

The robot functionality starts with the serial data transmission initialization between the Arduino and IMU sensor. All components of the robot wake up from its inactive or sleep mode when the power supply is switched on. Now IMU sensor starts reading data along side-to-side axis (pitch) and vertical axis (yaw). MPU6050 IMU contains inbuilt DMP that fuses accelerometer and gyroscope data together to reduce error effect of sensor and provides us accurate data. Just after the start robot gyro takes some milliseconds for calibration of offset value. Next step is to set a desired set point in the system. There is an inbuilt encoder in the MPU6050 which continuously reads the position and velocity of the robot, and depending on that it calculates tilt angle of the robot. We just compare these continuously received raw data bytes with the set point

and get the error which we have to eliminate by including a closed loop control system, i.e., PID controller to the device. PID controller has three control gain parameters namely  $K_p$ ,  $K_i$  and  $K_d$  to which we have to assign some random values by trial and error method and test its working with each new value until it is able to eliminate that error. The robot is based on two stepper motors which are controlled by the Arduino. Since the stepper motor is a pulse driven device, if the pulse frequency is increased in the motor then the speed of the motor also gets increased. Since in order to balance the robot, it is moved in the direction of fall so that it can come in upright position. For that the lower part of robot needs to move faster than upper part so that it can come again in balanced vertical position. In order to achieve balance in robot PID controller helps the Arduino to produce PWM output to make robot move with different required speed to reach set point or to come to upright position.

In this project the remote control is being used to steer the robot. When remote control instructs the robot to move left or right, then the PWM values of the output gets changed because in order to take the turn the wheels of the two wheeled robot move with different speeds. For example, if the robot needs to take left turn, then the right wheel connected with the right stepper motor will rotate faster than the left wheel connected with the left stepper motor in order to cover more distance in the same time. Similarly, while taking right turn, the right wheel will rotate slower than the left wheel. Hence, based on the movement instructions from the remote control, the PWM values get decided and finally applied to the stepper motors to achieve the desired objective.

### **6.3 Concept of balancing**

Balancing of the robot is based on the inverted pendulum principle. It is a simple concept and can also be related to the balancing of a rod on a wheeled cart. The main idea of balancing the robot is to move the wheels of the robot in the same direction in which it is falling. In simple manner, we can understand it with the act of our hand movements in order to balance a vertical rod on our palm. Through this action we try to keep the center of mass of the rod just above the contact point of rod and palm. Similarly, the robot has to keep its center of mass above the axis of rotation to maintain itself in a balanced position.

Balancing of a two wheeled robot means keeping the robot upright in vertical position [42]. Since the robot is an unstable system, it will try to fall down either in

forward or backward direction on horizontal plane. The angle of this tilt is measured by IMU sensor which filters this data by DMP. According to angle variations in the robot, the speed of the motor is decided. In order to achieve balance in robot, the motor starts rotating in the same direction in which it tries to fall. It gives an opposite moment to the body of robot which reduces or counters the instability of the system. The speed of motor is proportional to the angle of inclination or tilt of the robot and their relation should be accurately calculated to get better balancing. If the speed of motor is high, it will give a large amount of moment which becomes a reason for a high amplitude vibration and due to this robot can be more unstable. In the other case, even if motor speed is less then it will not give the necessary amount of torque to the robot and it can also make the robot unstable.

The simple method to balance the robot is that for small angle variation or tilt the speed of motor should be less, and the motor speed should be high for high tilt angle of the robot. One more condition should be fulfilled for balancing, that is the torque of the motor should be high enough to control or pull the weight of the complete robot body. When the robot will be powered on, IMU sensor will try to calibrate the offset and for some time the robot will vibrate with large amplitude, but within few seconds it will be balanced.

The robot can't be balanced if the tilt angle is very high because the weight and gravity of the robot will be more dominating over the thrust given by the motor. The tilt angle of the robot must be in a particular zone of forward and backward directions with respect to the vertical balanced position. If in the case the robot will go out of that limited zone, then it will again try to balance itself as quick as possible when the user brings it to the balanced zone. As long as the robot is in upright position, it stays stationary. Even in order to move in the forward or backward direction to the robot, the tilt angle in that direction should be changed by very small values so that it can balance itself and keep moving in the instructed particular direction. If the tilt angle variation will be kept high to make it move in a particular direction, then the speed of robot will also increase by large amount in order to balance that robot. Hence, the robot will become unstable. Also, the center of mass and gravity will dominate the thrust offered by the robot, and then it will fall down.

In this project, after testing the robot with remote control, I analyzed the maximum angle of tilt up to which the robot is able to keep itself in upright position without falling down. I found out that for my robot,  $-25^{\circ}$  to  $+38^{\circ}$  tilt angle is the zone in which the robot is able to balance itself. If in case the tilt angle is more than the balance zone angle, then it will become unstable and will fall down horizontally in either direction. I have also analyzed the direction of wheels of robot at every 5 degree interval of tilt angle. Table 6.1 and 6.2 shows tilt angle for forward and backward directions respectively.

Table 6.1 Angle of tilt of the robot for forward direction

Sl. No.	Angle of Tilt (in degree)	Wheel direction
1	10	Forward
2	15	Forward
3	20	Forward
4	25	Forward
5	30	Forward
6	35	Forward
7	38	Forward

Table 6.2 Angle of tilt of the robot for backward direction

Sl. No.	Angle of Tilt (in degree)	Wheel direction
1	5	Backward
2	0	Backward
3	-5	Backward
4	-10	Backward
5	-15	Backward
6	-20	Backward
7	-25	Backward

Fig. 6.1 shows response of the robot corresponding to tilt angles. The robot will be able to balance itself for a tilt angle between  $-25^\circ$  to  $+38^\circ$ . Beyond that, it will not be able to balance itself and will fall down horizontally.

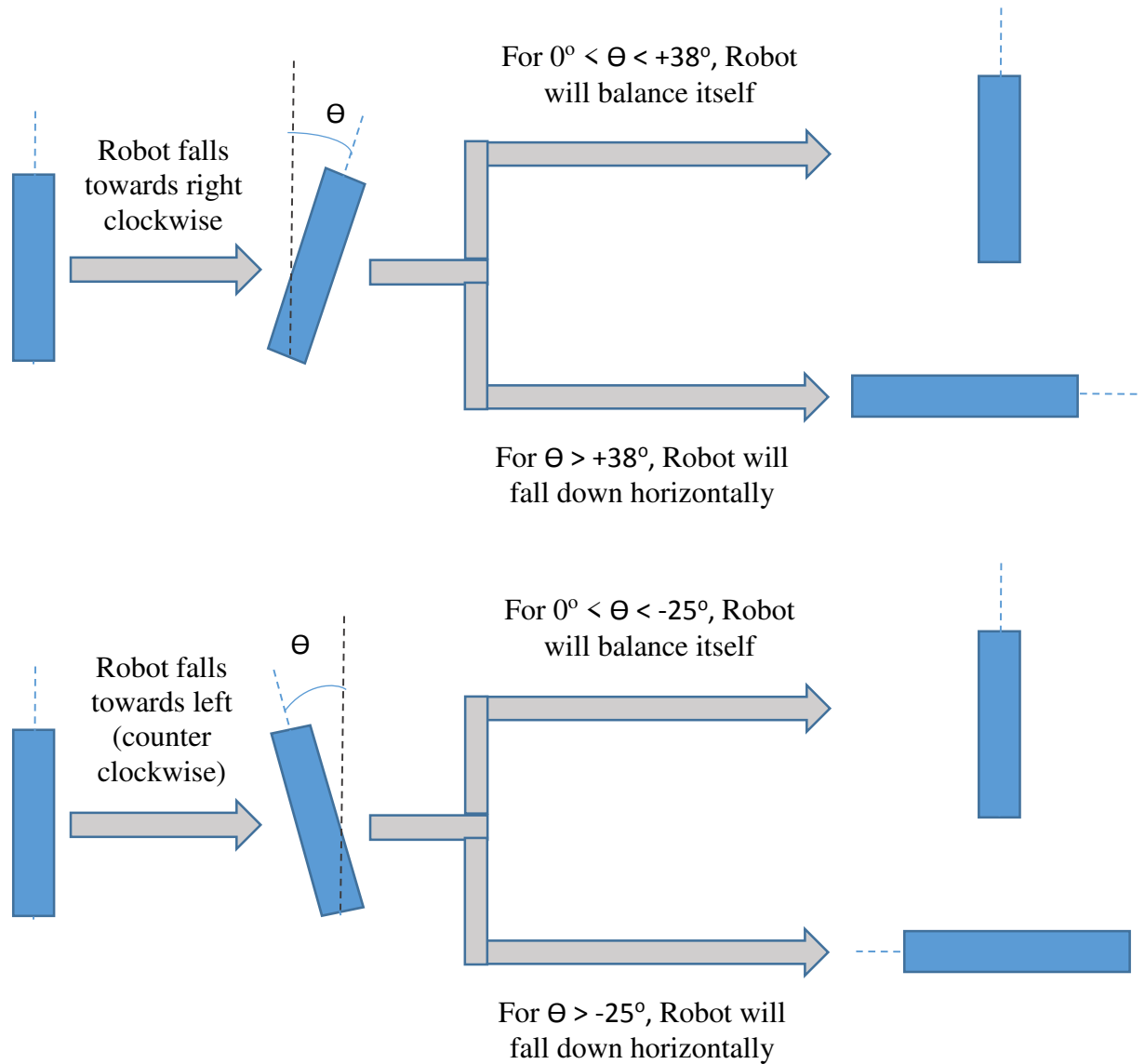


Fig. 6.1 Response of the robot corresponding to tilt angles

## 6.4 PID Tuning

To achieve the desired working of the robot, the PID controller must be tuned to suit with the dynamics of the robot. Tuning is basically the method of changing the values of the controller gain parameters in order to achieve the desired output with respect to the corresponding input given to the system. PID controller tuning is done for disturbance rejection that means if the set point of robot is changed, the controller output should follow the new set point. We need to do PID tuning properly so that controller output will follow variable set point with less damping and less oscillation. The PID controller can be tuned by using various methods like Trial and Error method, Process Reaction Curve technique, or, Zeigler-Nichols method, etc. In my project, I have used Trial and Error method for tuning the PID controller.

Trial and Error method is the simplest method and is also known as manual tuning method. In this method, first we keep the values of  $K_i$  and  $K_d$  to zero and will just increase the value of  $K_p$  until the system or robot reaches to steady oscillating state, but system should not become unstable during this procedure. After this, we will adjust the value of  $K_d$  in order to stop oscillation of the system. We will repeat changing values of  $K_p$  and  $K_d$  until increasing D-Controller gain doesn't stop oscillations. Once the oscillation of the system stops, we will adjust  $K_i$  value in order to get fast response. Hence, by manual method we can successfully achieve a desired control in the robot to keep it in proper balanced state.

In my first trial, I took the value of  $K_p= 10$ ,  $K_i= 0$ , and  $K_d= 0$  for test of working of robot. This time the robot was frequently falling down in either direction. Fig. 6.2 shows the graph to give an idea of response of robot with these gain parameters.

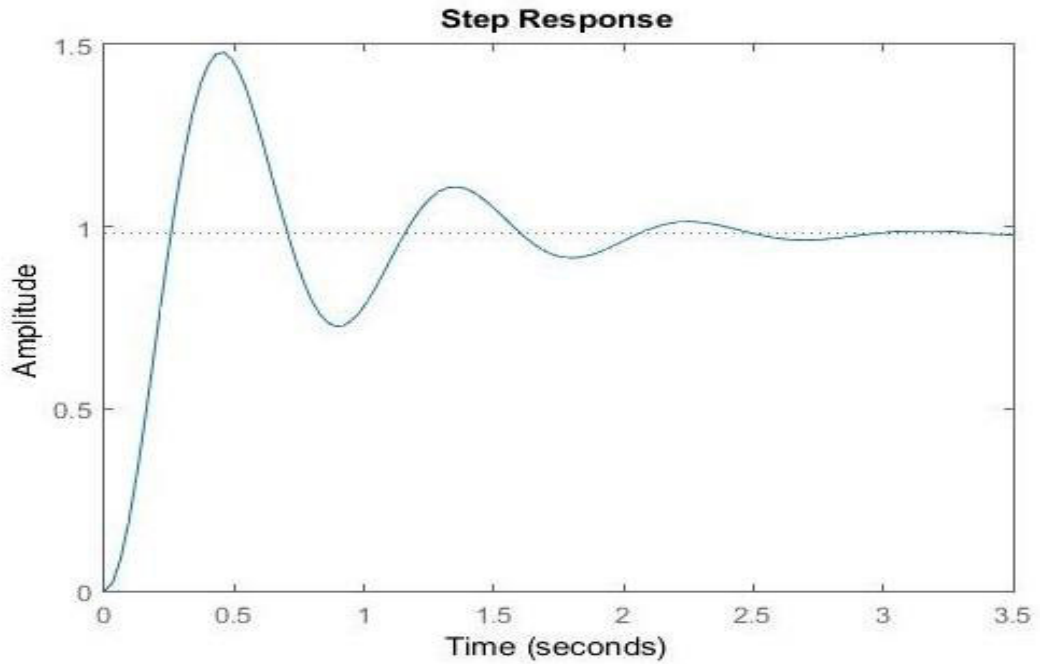


Fig. 6.2 shows step response for  $K_p=10$ ,  $K_d=0$  and  $K_i=0$  [19]

Then I increased the value of  $K_p=15$ , and left the remaining values same as previous. Fig. 6.3 shows the graph to give an idea of response of robot with these values of gain parameters. In this test the robot had steady oscillations across perpendicular axis.

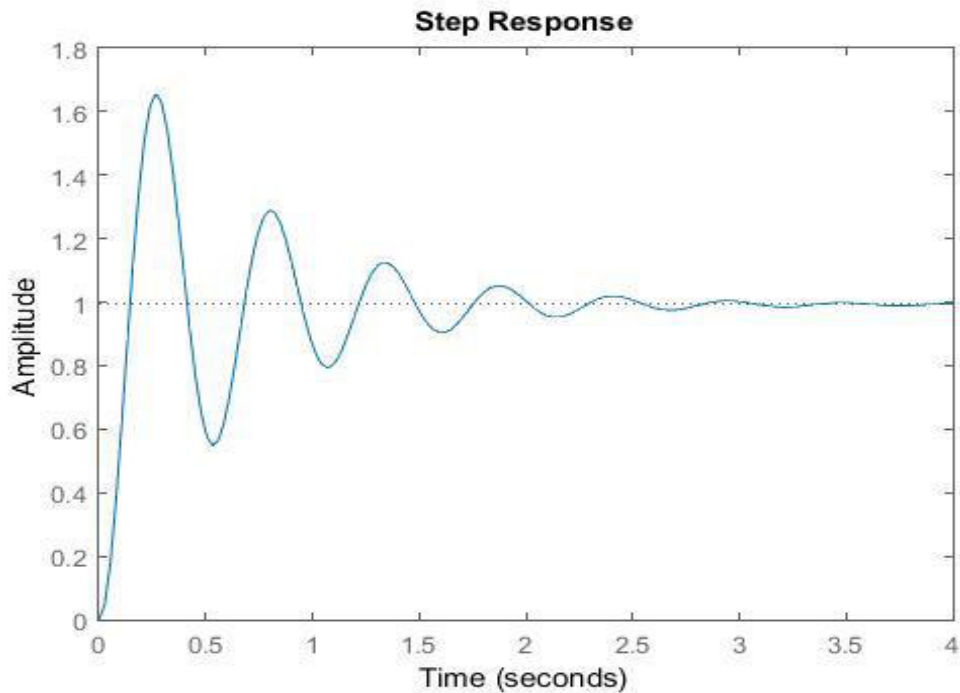


Fig. 6.3 shows step response for  $K_p=15$ ,  $K_d=0$  and  $K_i=0$  [19]

Now, I started increasing the value of  $K_d$  to 10 and left the other two values unchanged. The robot was more stable as compared to the earlier one but still there was visible oscillation in the robot. Fig. 6.4 shows the graph to give an idea of response of robot with these gain parameters.

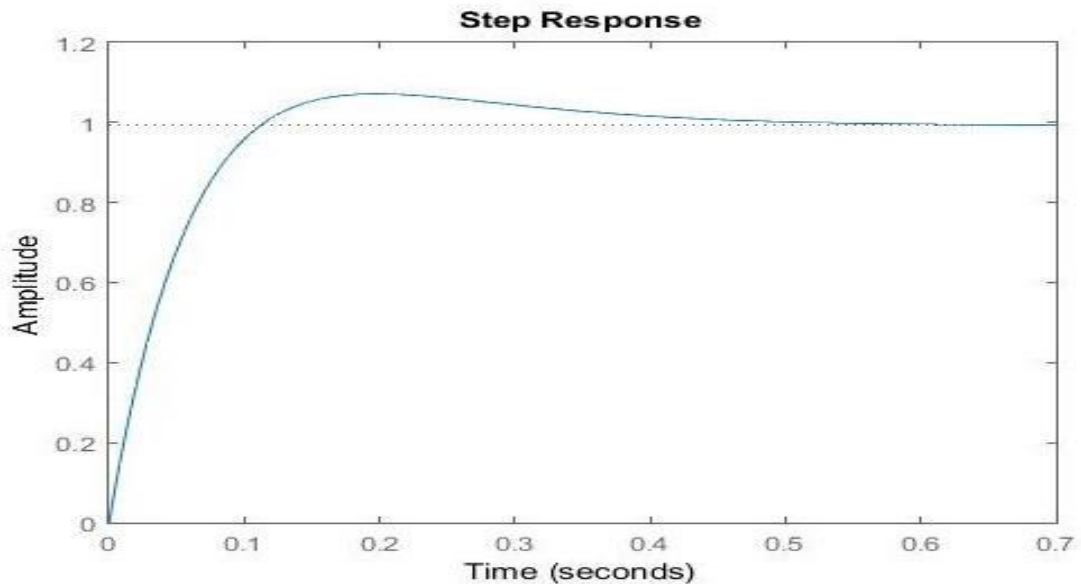


Fig. 6.4 shows step response for  $K_p = 15$ ,  $K_d = 10$  and  $K_i = 0$  [19]

Again, I increased the value of only  $K_d$  to 40, now the oscillation has damped or negligible and system is getting stable in upright direction. Fig. 6.5 shows a graph that gives an idea of response of robot with these values of gain parameters.

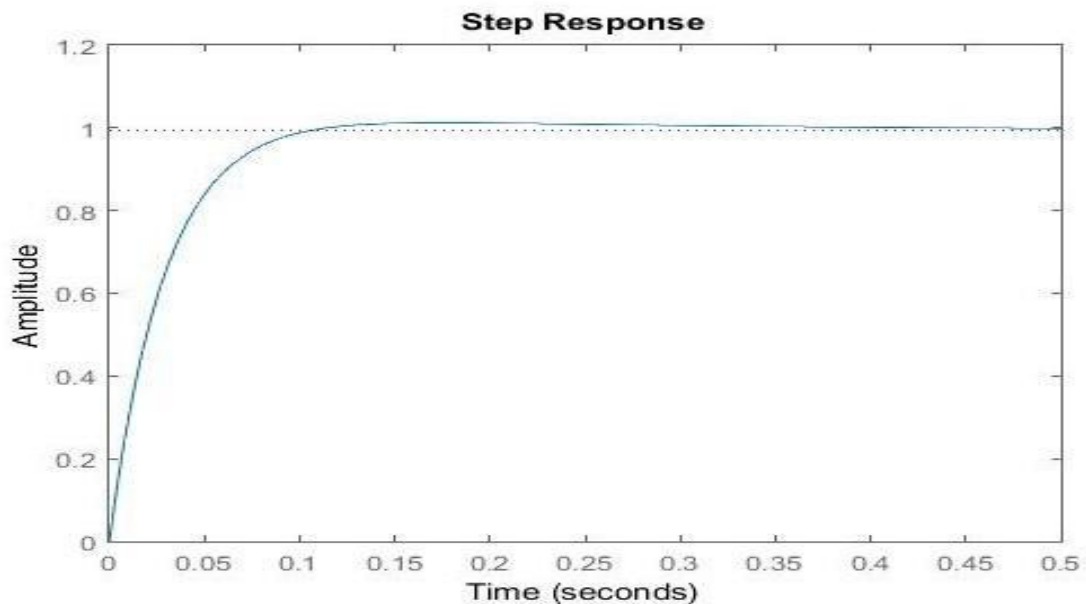


Fig. 6.5 shows step response for  $K_p = 15$ ,  $K_d = 40$  and  $K_i = 0$  [19]



But the time taking to reach this stability is more so in order to reduce the response time we start changing  $K_i$  value and left other values unchanged. This time I set  $K_i$  value to 0.5 and tested the robot. Fig. 6.6 shows a graph that gives an idea of response of robot with these values of gain parameters.

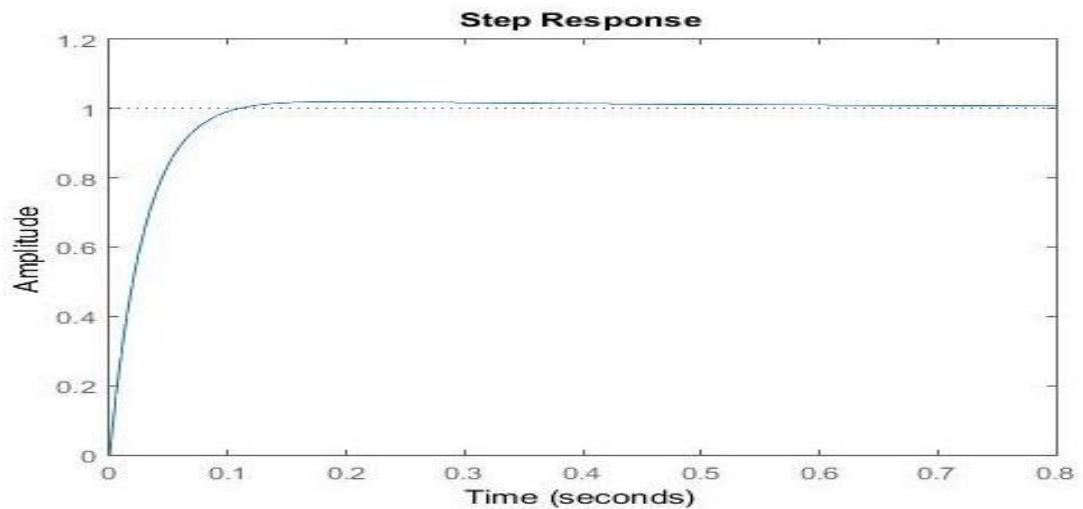


Fig. 6.6 shows step response for  $K_p = 15$ ,  $K_d = 40$  and  $K_i = 0.5$  [19]

I found that its response time is getting improved but still it wasn't much efficient, so I tested the working of robot again and again with different values. With the value of  $K_i = 2.0$ , it was good to see the robot that its response is quite faster to reach in stable position. Fig. 6.7 shows a graph that gives an idea of faster and stable response of robot with these values of gain parameters. Hence by trial and error method we succeed to achieve balanced and stable robot.

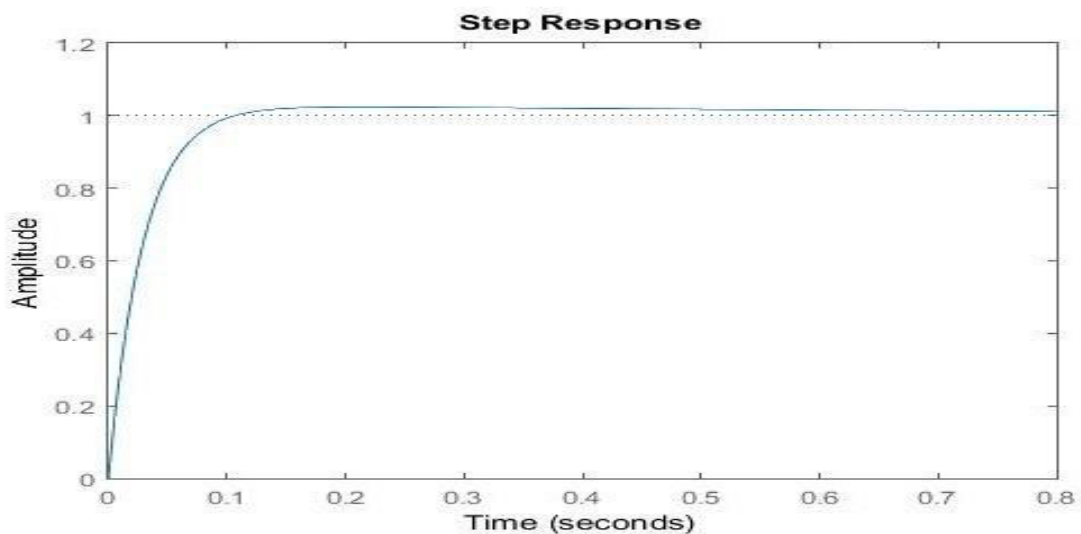


Fig. 6.7 shows step response for  $K_p = 15$ ,  $K_d = 40$  and  $K_i = 2.0$  [19]

There are four metallic supports attached to the robot. These supports are attached to prevent it from falling down after getting unbalanced. The above final values of control gain parameters have the same effect in balancing of robot as earlier even after adding four metallic supports with the robot. If the robot is having tilt angle under its balance zone, then the robot with these four supports will maintain its upright position. Otherwise it will stand on either side of supports instead of falling down to the ground or horizontal plane. Fig. 6.8 shows the view of robot with four supports with no power supply, and fig. 6.9 shows the view of robot with four supports balancing itself when power supply is made available.



Fig. 6.8 A view of robot with four supports with no power supply



Fig. 6.9 A view of robot with four supports when it starts balancing itself

## 6.5 Algorithms for hardware testing

### 6.5.1 Algorithm for MPU6050

At first we take the necessary header files to support the Arduino software.

- Declare some global variables that we need in the program.
- Start I2C communication.
- Set I2C clock speed to 400 kHz.
- Select transmission rate to 9600 bps.
- Start scanning for a device on 7-bit address that means 126 addresses.
- Check if any I2C device is found on any one of the addresses or not.
- If no I2C device is identified there, then print the message on serial monitor as “No I2C device found”.
- In case there is any I2C device found then verify the address of that I2C device whether it is MPU6050 or not.
- If it is a MPU6050 then print a message “starting gyro....” and set all the necessary registers in it.
- Check and print the desired accelerometer set point or balance value of MPU6050 on serial monitor.
- Read the gyro value of IMU sensor across x, y and z-axes and print these on the serial monitor. Fig.6.10 shows MPU6050 testing result on the serial monitor.
- Save the value of set point or balance value to use it in the main code of self-balancing robot.

```
COM11 (Arduino/Genuino Uno)
Scanning I2C bus...
I2C device found at address 0x68
This could be a MPU-6050
Send Who am I request...
Who Am I response is ok: 0x68
Starting Gyro...
done

Balance value: 458
Printing raw gyro values
Gyro X = 373 Gyro Y = -352 Gyro Z = 200
Gyro X = 364 Gyro Y = -323 Gyro Z = 201
Gyro X = 368 Gyro Y = -298 Gyro Z = 216
Gyro X = 375 Gyro Y = -284 Gyro Z = 207
Gyro X = 370 Gyro Y = -288 Gyro Z = 213
Gyro X = 377 Gyro Y = -306 Gyro Z = 210
Gyro X = 372 Gyro Y = -331 Gyro Z = 214
Gyro X = 384 Gyro Y = -355 Gyro Z = 207
Gyro X = 360 Gyro Y = -353 Gyro Z = 213
Gyro X = 365 Gyro Y = -314 Gyro Z = 209
Gyro X = 368 Gyro Y = -304 Gyro Z = 213
Gyro X = 365 Gyro Y = -314 Gyro Z = 202
Gyro X = 376 Gyro Y = -318 Gyro Z = 200
Gyro X = 374 Gyro Y = -333 Gyro Z = 223
Gyro X = 368 Gyro Y = -322 Gyro Z = 201
Gyro X = 383 Gyro Y = -320 Gyro Z = 212
Gyro X = 369 Gyro Y = -320 Gyro Z = 205
Gyro X = 380 Gyro Y = -316 Gyro Z = 213
Gyro X = 370 Gyro Y = -343 Gyro Z = 219
Gyro X = 375 Gyro Y = -340 Gyro Z = 208

No Nunchuck device found at address 0x52
```

Fig. 6.10 Output of MPU6050 testing on the serial monitor

## 6.5.2 Algorithm for Joystick

- Add header file required to support the code.
- Take global variables needed for the sketch.
- Scan the serial port to match the address of device from hex values 1 to 126.
- Verify whether there is any I2C device connected on serial port or not and if it is I2C device then again check if it is a joystick (nunchuk) or not.
- If it is a nunchuk device connected on serial port, then print its raw values across x and y axes on the serial monitor. Figures 6.11, 6.12, 6.13 and 6.14 show outputs displayed on serial monitor when stick moved right, left, up and down respectively.







## 6.6 Algorithm for Bluetooth

### 6.6.1 For slave device

- Upload an empty sketch in the Arduino UNO.
- Open serial monitor and enter AT commands.
- At first type AT in serial monitor window.
- Do factory reset to the bluetooth.
- Enter AT command to remove previous paired device from this device.
- Check the password of the current device.
- Check serial transmission rate of the bluetooth.
- Enter AT command to know the role of the device i.e. it is in slave role or in master role. If it is in master role change it to slave mode. Fig. 6.15 shows Serial monitor output after entering AT commands for slave device.
- Find the address of current device and save it for further use.

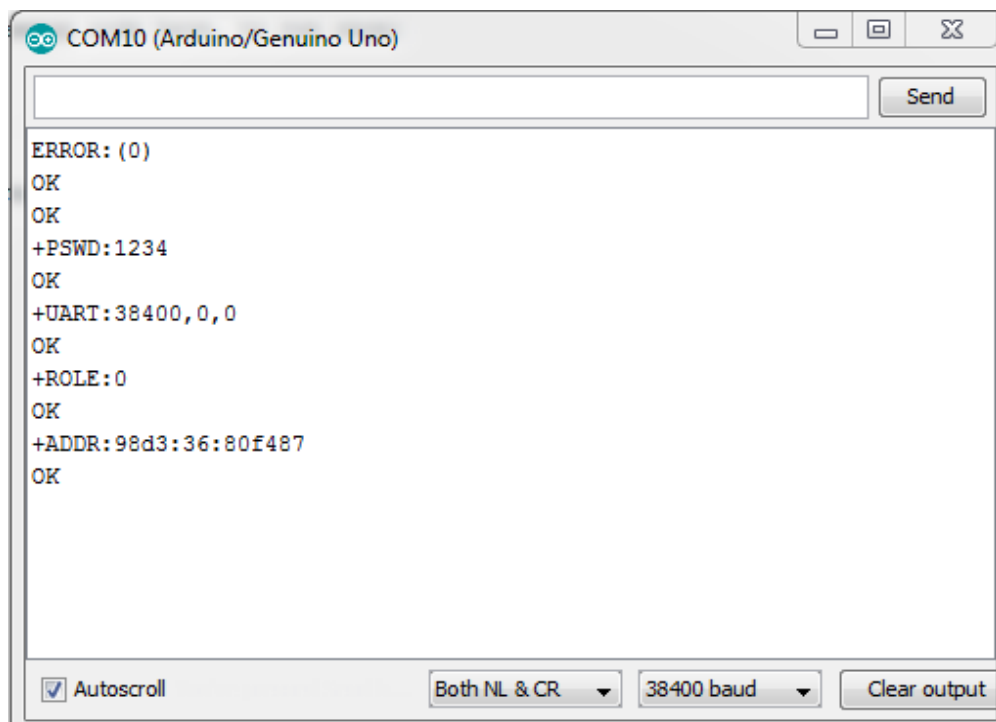
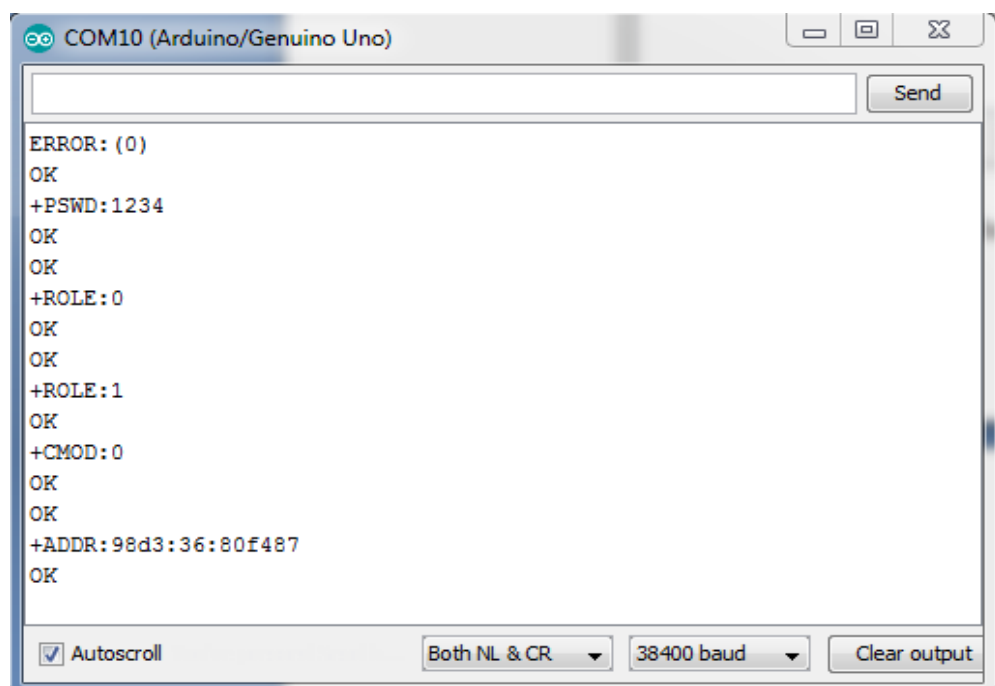


Fig. 6.15 Serial monitor output after entering AT commands for slave device



## 6.6.2 For master device

- Upload an empty sketch in the Arduino.
- Open an independent terminal of serial monitor and enter there AT commands to configure given bluetooth as a master device.
- Enter AT command to factory reset the bluetooth.
- Check the password of given device to make sure master and slave device must have the same password.
- Give command to remove previously paired device.
- Check the role of the device. If it is in slave role (0) then make it a master device (1).
- Give the proper command to make sure this device can communicate only with the paired one.
- Pair it with the desired slave device by writing the address of that slave device in the current bluetooth. Fig 6.16 shows serial monitor output after entering AT commands for master device.



```
COM10 (Arduino/Genuino Uno)
ERROR: (0)
OK
+PSWD:1234
OK
OK
+ROLE:0
OK
OK
+ROLE:1
OK
+CMOD:0
OK
OK
+ADDR:98d3:36:80f487
OK
```

The screenshot shows the Serial Monitor window for COM10 (Arduino/Genuino Uno). The output displays the following AT command responses: ERROR: (0), OK, +PSWD:1234, OK, OK, +ROLE:0, OK, OK, +ROLE:1, OK, +CMOD:0, OK, OK, +ADDR:98d3:36:80f487, and OK. The window includes a 'Send' button at the top right and a 'Clear output' button at the bottom right. The baud rate is set to 38400 and the line ending is set to 'Both NL & CR'.

Fig. 6.16 Serial monitor output after entering AT commands for master device

## 6.7 Algorithm for self-balancing robot

- Take the necessary header file and global variables to support the program in Arduino IDE software.
- Write the set point or accelerometer calibration value which we get from hardware testing of MPU6050.
- Configure PID constant and speed variables.
- Set transmission rate and I2C clock frequency in setup loop after starting I2C communication.
- Create a timer that will execute a particular interrupt section of code after every 20 microseconds to create a variable pulse to control the stepper motors. For this customize the registers of 8-bit Timer 2 of the Arduino timer.
- Do the necessary changes in MPU6050 registers to wake it up from sleep mode and set to required full scale of gyro and accelerometer.
- Connect the Arduino with stepper motor controllers by declaring some of its pins as output pins.
- Take some raw data from IMU sensor and calculate its average to get offset values of the sensor, in order to calibrate it.
- The Arduino receives serial data from joystick in each loop and depending on that it instructs stepper motor to move in a particular direction.
- The Arduino reads analog data of voltage or power of Li-Po battery in order to save device from any loss in case of low battery or insufficient power supply.
- According to the angle change accelerometer set point or offset value is updated to the new precision in each loop in order to balance the robot.
- PID controller, stepper motor speed and direction and motor pulse calculations are done to achieve a stable self-balancing robot movement.

## 6.8 Algorithm to steer the robot by remote control

- Take the necessary header file and global variables to support the program in Arduino IDE software.
- Initialize I2C communication and set necessary serial transmission rate and I2C clock frequency for the device.
- After starting communication of the Arduino with the nunchuk customize the registers of it in order to make it suitable for remote controlling the robot.
- Read bytes from the nunchuk in order give direction instructions to the robot.
- On depending upon some calculation on the received data bytes from nunchuk, it sends 4 different kinds of data bytes to the robot via bluetooth to make it move according to the instructions given by nunchuk operator. Fig. 6.17 and 6.18 shows final working model of self-balancing robot without four supports and with four supports respectively.

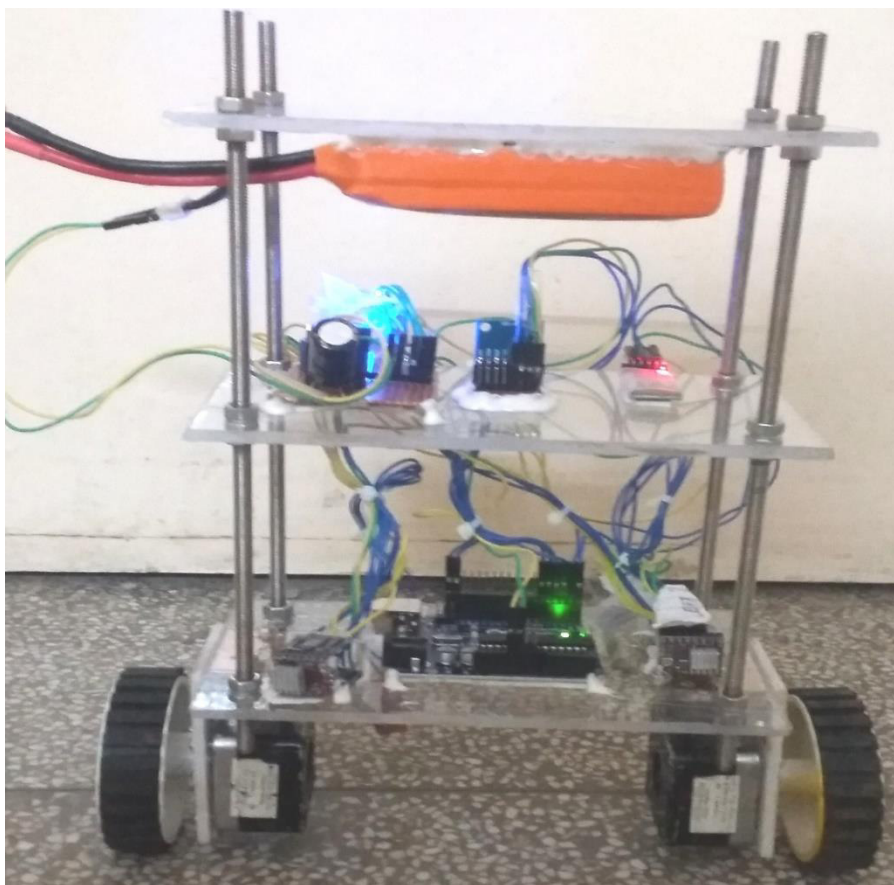


Fig. 6.17 Final working model of the robot without four supports

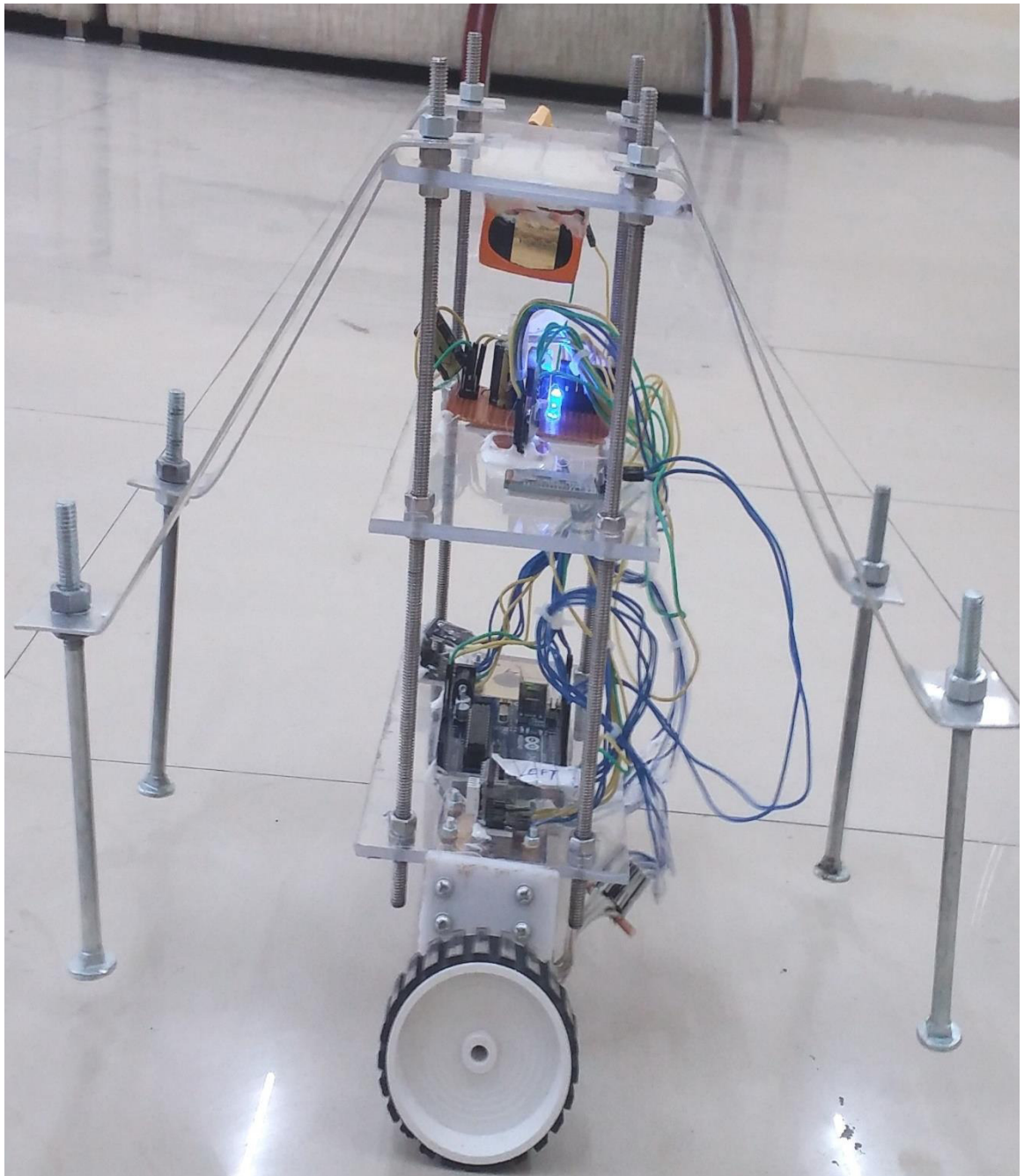


Fig. 6.18 Final working model of the robot with four supports

# CHAPTER 7

## Conclusion and Future Work

### 7.1 Conclusion

This project was aimed to fabricate and to test working of the self-balancing robot. The design and fabrication of the robot is correct with the fixed position its components even at maximum amplitude of vibration. The robot body is able to tolerate the sudden force to a good extent in the case of its falling down horizontally on the ground when it goes out of limited balancing zone.

The power transmission source is sufficient enough to supply the required amount of power to the robot. In the project, rechargeable battery is used so it has benefits in testing robot's working any number of times without thinking about the replacement of the discharged battery.

To make this robot, I did refer to the robot Bimbo [4], and have attempted to fabricate a better robot with improved stability. The final robot is successfully balancing itself for different tilt angles, and has some additional features such as larger frame and remote controlled operation. The robot is giving proper response to the instructions given by the remote control. The wireless communication between the robot and its remote control is established without any failure.

The maximum zone ( $-25^{\circ}$  to  $+38^{\circ}$  degree tilt angle) in which the robot is able to balance itself without any external help, is analyzed properly.

### 7.2 Future work

- It can be further enhanced to work with vehicle mechanics. For example, it can be added as a safety feature in two-wheelers. These vehicles can balance themselves sideways whenever the rider loses his balance, therefore can significantly decrease the road accidents rate.

- The robotic arms can be attached with it for pick and place purposes in the warehouses, hotels and malls.
- It can be used to develop self-balancing wheelchairs for disabled patients.
- In future, it can be modified in a legged robot where each movable layers are balanced by using different control systems.
- Some modification and analysis can be further done in this robot in order to increase the balancing zone.
- Future work can be done on this robot to make it move and balance itself in different terrains.

## REFERENCES

- [1] Kamen, D., “Segway PT”, 2001. Available online on: [www.segway.com](http://www.segway.com).
- [2] Junoh, S. A. B., “Two-wheeled balancing robot controller designed using PID”, 2015. Universiti Tun Hussein Onn, Malaysia.
- [3] Anderson, D. P., “nBot”, 2003. Available online on: [www.geology.smu.edu](http://www.geology.smu.edu).
- [4] Martins, R. S., Nunes, F., “Control system for a self-balancing robot”, 2017. International conference, University of Algarve, Faro, Portugal.
- [5] Junfeng, W., Wanying, Z., “Research on control method of two-wheeled self-balancing robot”, 2011. 4<sup>th</sup> International conference, Intelligent Computation Technology and Automation.
- [6] Feng, T., Wang, X., Liu, T., Xu, Z., Zhang, M., Han, S. C., “Modeling and implementation of two-wheel self-balancing robot equipped with supporting arms”, 2011. IEEE, 6th Conference, Industrial Electronics and Applications.
- [7] Wu, J., Liang, Y., Wang, Z., “A robust control method of two-wheeled self-balancing robot”, 2011. IEEE, 6th International Forum, Strategic Technology.
- [8] Ferdinando, H., Khoswanto, H., Tjokro, S., “Design and evaluation of two-wheeled balancing robot chassis”, 2010. Petra Christian University, Indonesia.
- [9] Juang, H. S., Lum, K. Y., “Design and control of a two-wheel self-balancing robot using the Arduino microcontroller board”, 2013. IEEE, 10th International Conference, Control and Automation (ICCA), China.
- [10] Sun, C., Lu, T., Yuan, K., “Balance control of two-wheeled self-balancing robot based on Linear Quadratic Regulator and Neural Network”, 2013. IEEE, 4th

International Conference, Intelligent Control and Information Processing (ICICIP), China.

- [11] An, W., Li, Y., “Simulation and control of a two-wheeled self-balancing robot”, 2013. Proceeding of IEEE, International Conference, Robotics and Biometrics (ROBIO), China.
- [12] Jamil, O., Jamil, M., Ayaz, Y., Ahmad, K., “Modeling, Control of a two-wheeled self-balancing robot”, 2014. IEEE, International Conference, Robotics and Emerging Allied Technologies in Engineering (iCREATE), Pakistan.
- [13] Prasetio, B. H., “Ensemble Kalman filter and PID controller implementation on self-balancing robot”, 2015. IEEE, International Electronics Symposium (IES).
- [14] Prakash, K., Thomas, K., “Study of controllers for a two-wheeled self-balancing robot”, 2016. IEEE, International Conference, Next Generation Intelligent Systems (ICNGIS).
- [15] Ali, M. I., Hossen, M. M., “A two-wheeled self-balancing robot with dynamics model”, 2017. IEEE, Proceedings of 4<sup>th</sup> International Conference, Advances in Electrical Engineering (ICAEE), Bangladesh.
- [16] Arduino Interrupt - <https://www.teachmemicro.com/arduino-timer-interrupt-tutorial/>
- [17] Sketch reference- [http://www.brokking.net/yabr\\_main.html](http://www.brokking.net/yabr_main.html)
- [18] DMP and Euler angle concept- <http://www.geekmomprojects.com/mpu-6050-dmp-data-from-i2cdevlib/>
- [19] PID Controller response graph for step input-  
[https://www.youtube.com/watch?v=cJ7q4EltyE0&list=LLG1061JFGoUEe\\_o6ljXEu6Q&index=9&t=0s](https://www.youtube.com/watch?v=cJ7q4EltyE0&list=LLG1061JFGoUEe_o6ljXEu6Q&index=9&t=0s)



- [20] Arduino UNO-  
<https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-uno.html>
- [21] Stepper motor- <https://www.pololu.com/product/2267>
- [22] Stepper motor driver module (DRV8825)-  
<https://lastminuteengineers.com/drv8825-stepper-motor-driver-arduino-tutorial/>
- [23] IMU sensor (MPU6050)- <https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module>
- [24] Bluetooth module (HC05)- <https://www.electronicwings.com/sensors-modules/bluetooth-module-hc-05->
- [25] Arduino NANO- <https://store.arduino.cc/usa/arduino-nano>
- [26] Wii nunchuk-<https://www.instructables.com/id/Wireless-Wii-Nunchuck-control-of-Arduino-projects/>
- [27] ORANGE 11.1V/18800mAh 3S 30C Li-Po battery-  
<https://robu.in/product/orange-11-1v-1800mah-3s-30c-lipo-battery-pack-xt60-connector/>
- [28] PID Controller block diagram- <https://circuitdigest.com/article/what-is-pid-controller-working-structure-applications>
- [29] I2C interface- <https://www.engineersgarage.com/tutorials/twi-i2c-interface>
- [30] Piezoelectric effect- <https://www.nanomotion.com/piezo-ceramic-motor-technology/piezoelectric-effect/>
- [31] PID controller working- <https://www.elprocus.com/the-working-of-a-pid-controller/>

- [32] Arduino IDE Software Introduction-  
<https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html>
- [33] Inverted pendulum concept-  
<http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling>
- [34] Joystick image- <https://www.techexpress.co.nz/products/nunchuk-game-controller-for-nintendo-wii>
- [35] Nunchuk pin view image- <https://forum.arduino.cc/index.php?topic=67962.0>
- [36] Arduino Nano image- <https://www.amazon.com/OSOYOO-Arduino-ATMEGA328P-Microcontroller-Without/dp/B00UACD13Q>
- [37] Stepper motor image- <https://www.amazon.com/Stepper-Motor-Bipolar-64oz-Printer/dp/B00PNEQI7W>
- [38] MPU6050 image- <https://www.elementsonline.com/mpu6050-gy-521-3-axis-analog-gyro-sensors-accelerometer-module>
- [39] HC05 image- <https://www.dhgate.com/product/hc05-hc-05-master-slave-6pin-jy-mcu-anti/402173542.html>
- [40] Wheel image- <https://www.deltakit.net/product/big-wheel/>
- [41] Set point and actual tilt angle of the robot image-  
<https://www.semanticscholar.org/paper/i-TWO-WHEELED-BALANCING-ROBOT-CONTROLLER-Onn-diperlukan/5a5b3b44c6456ae231162ce013a8e493dc1bc6db>
- [42] Padhan, M. K., “Fabrication, balancing and analysis of two wheeled robot”, 2015. National institute of technology, Rourkela, Odisha.

# PLAGIARISM REPORT

## ORIGINALITY REPORT

**21%**

SIMILARITY INDEX

**7%**

INTERNET SOURCES

**3%**

PUBLICATIONS

**20%**

STUDENT PAPERS

## PRIMARY SOURCES

<b>1</b>	<b>Submitted to Malaviya National Institute of Technology</b> Student Paper	<b>3%</b>
<b>2</b>	<b>Submitted to National Institute of Technology, Rourkela</b> Student Paper	<b>3%</b>
<b>3</b>	<b>eprints.uthm.edu.my</b> Internet Source	<b>2%</b>
<b>4</b>	<b>Submitted to De Montfort University</b> Student Paper	<b>1%</b>
<b>5</b>	<b>Submitted to University Tun Hussein Onn Malaysia</b> Student Paper	<b>1%</b>
<b>6</b>	<b>www.theengineeringprojects.com</b> Internet Source	<b>1%</b>
<b>7</b>	<b>Submitted to Universiti Teknologi Malaysia</b> Student Paper	<b>1%</b>
<b>8</b>	<b>Submitted to Universiti Teknologi MARA</b> Student Paper	<b>1%</b>

9	<b>Submitted to Thapar University, Patiala</b> Student Paper	<1%
10	<b>Submitted to Higher Education Commission Pakistan</b> Student Paper	<1%
11	<b>Submitted to Universiti Teknikal Malaysia Melaka</b> Student Paper	<1%
12	<b>Submitted to Segi University College</b> Student Paper	<1%
13	<b>Submitted to Centre for Development of Advanced Computing</b> Student Paper	<1%
14	<b>Submitted to Asian Institute of Technology</b> Student Paper	<1%
15	<b>Submitted to University of Nottingham</b> Student Paper	<1%
16	<b>Submitted to BITS, Pilani-Dubai</b> Student Paper	<1%
17	<b>circuitdigest.com</b> Internet Source	<1%
18	<b>Submitted to Birla Institute of Technology and Science Pilani</b> Student Paper	<1%

19

**Submitted to University of Michigan, Dearborn**

Student Paper

<1%

20

**Submitted to University of Leeds**

Student Paper

<1%

21

**Submitted to Universiti Tenaga Nasional**

Student Paper

<1%

22

**Submitted to The University of Manchester**

Student Paper

<1%

23

**Submitted to College of Science and  
Technology, Bhutan**

Student Paper

<1%

24

**Submitted to City University**

Student Paper

<1%

25

**playground.arduino.cc**

Internet Source

<1%

26

**Gonzalo Maldonado-Guzmán, Jose Arturo  
Garza-Reyes, Lizeth Itziguery Solano-Romo.  
"System Architecture", Emerald, 2019**

Publication

<1%

27

**Pongtorn Chunchacha. "Parameters tuning  
effects in the model predictive control of an  
inverted pendulum", TENCON 2011 - 2011  
IEEE Region 10 Conference, 11/2011**

Publication

<1%

**Submitted to The Hong Kong Polytechnic**

<b>28</b>	<b>University</b> Student Paper	<1%
<b>29</b>	<b>Submitted to American College of the Middle East</b> Student Paper	<1%
<b>30</b>	<b>Submitted to University of Sheffield</b> Student Paper	<1%
<b>31</b>	<b>3dprintnerd.com</b> Internet Source	<1%
<b>32</b>	<b>Submitted to California Lutheran University</b> Student Paper	<1%
<b>33</b>	<b>www.circuitbasics.com</b> Internet Source	<1%
<b>34</b>	<b>Submitted to University of Salford</b> Student Paper	<1%
<b>35</b>	<b>Submitted to Siddaganga Institute of Technology</b> Student Paper	<1%
<b>36</b>	<b>Submitted to Multimedia University</b> Student Paper	<1%
<b>37</b>	<b>Submitted to Universiti Teknologi Petronas</b> Student Paper	<1%
<b>38</b>	<b>Submitted to Ateneo de Manila University</b> Student Paper	<1%

39	<b>Submitted to The London College UCK</b> Student Paper	<1%
40	<b>www.coursehero.com</b> Internet Source	<1%
41	<b>Submitted to University of Newcastle upon Tyne</b> Student Paper	<1%
42	<b>www.elprocus.com</b> Internet Source	<1%
43	<b>Submitted to Universiti Malaysia Perlis</b> Student Paper	<1%
44	<b>Submitted to Engineers Australia</b> Student Paper	<1%
45	<b>www.irjet.net</b> Internet Source	<1%
46	<b>Submitted to RMIT University</b> Student Paper	<1%
47	<b>Submitted to Rajarambapu Institute of Technology</b> Student Paper	<1%
48	<b>Submitted to Singapore Institute of Technology</b> Student Paper	<1%
49	<b>Submitted to Institute of Graduate Studies, UiTM</b>	<1%

**50** **Submitted to Baskent University** <1%  
Student Paper

---

**51** **www.mikroe.com** <1%  
Internet Source

---

**52** **repository.tudelft.nl** <1%  
Internet Source

---

**53** **Submitted to Victoria University College** <1%  
Student Paper

---

**54** **Submitted to University of Birmingham** <1%  
Student Paper

---

**55** **Submitted to University of Greenwich** <1%  
Student Paper

---

**56** **Submitted to Atilim University** <1%  
Student Paper

---

**57** **Submitted to Madhav Institute of Technology & Science** <1%  
Student Paper

---

**58** **Submitted to Emirates Aviation College, Aerospace & Academic Studies** <1%  
Student Paper

---

**59** **Submitted to Laureate Education Inc.** <1%  
Student Paper

---

**60** **M. Saif, B. Ebrahimi, M. Vali. "Terminal sliding**



mode control of Z-axis MEMS gyroscope with  
observer based rotation rate estimation",  
Proceedings of the 2011 American Control  
Conference, 2011

Publication

<1%

61

[repository.up.ac.za](http://repository.up.ac.za)

Int ernet Source

<1%

62

Submitted to University of Bristol

Student Paper

<1%

63

Submitted to iGroup

Student Paper

<1%

64

Submitted to CSU, Chico

Student Paper

<1%

65

Submitted to Lebanese International University

Student Paper

<1%

66

Submitted to Coventry University

Student Paper

<1%

67

[era.library.ualberta.ca](http://era.library.ualberta.ca)

Int ernet Source

<1%

68

Submitted to NIT Imphal

Student Paper

<1%

69

[tii.ieee-ies.org](http://tii.ieee-ies.org)

Int ernet Source

<1%

70

Submitted to University of Sunderland

Student Paper

<1%

---

<b>71</b>	<b>Submitted to Heriot-Watt University</b> Student Paper	<1%
<b>72</b>	<b>Submitted to University of Surrey</b> Student Paper	<1%
<b>73</b>	<b>Submitted to University of Durham</b> Student Paper	<1%
<b>74</b>	<b>fr.scribd.com</b> Internet Source	<1%
<b>75</b>	<b>The Anh Mai, Dmitry Nikolaevich Anisimov, Thai Son Dang, Ekaterina Fedorova. "Fuzzy-PID Controller for Two Wheels Balancing Robot Based on STM32 Microcontroller", 2019 International Conference on Engineering Technologies and Computer Science (EnT), 2019</b> Publication	<1%
<b>76</b>	<b>Submitted to University of South Australia</b> Student Paper	<1%
<b>77</b>	<b>Submitted to Glyndwr University</b> Student Paper	<1%
<b>78</b>	<b>Submitted to University of Lincoln</b> Student Paper	<1%
<b>79</b>	<b>Submitted to Napier University</b> Student Paper	<1%

---