BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 318 (January 2023)
Full Title of the Course Laboratory

Final Project Report

Section: A1 Group: 02

Self Balancing Robot

Course Instructors:

Dr. Mohammad Ariful Haque, Professor
Mrinmoy Kundu, Part-Time Lecturer

Signature of Instructor: ____________________

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: Muzakkir **Full Name: Muzakkir Ahmed** **Student ID: 1906002**	**Signature:** [illegible] **Full Name: Md. Rowatul Jannat Rafi** **Student ID: 1906016**
Signature: সাব্বির **Full Name: Md. Sabbir Ahmed** **Student ID: 1906004**	**Signature:** Rifat **Full Name: Md. Rifat Ullah** **Student ID: 1906017**
Signature: Sami **Full Name: Md. Sad Abdullah Sami** **Student ID: 1906011**	**Signature:** Rayhan Siddiq **Full Name: Rayhan Siddiq** **Student ID: 1906022**

Table of Contents

1 Abstract 1

2 Introduction 1

3 Design 2

3.1 Problem Formulation 2

3.1.1 Identification of Scope 2

3.1.2 Literature Review 3

3.1.3 Formulation of Problem 4

3.1.4 Analysis 5

3.2 Design Method 6

3.3 Circuit Diagram 11

3.4 Simulation Model 15

3.5 CAD/Hardware Design 20

3.6 PCB Design 22

3.7 Full Source Code of Firmware 22

4 Implementation 23

4.1 Description 23

4.2 Experiment and Data Collection 23

4.3 Data Analysis 23

4.4 Results 23

5 Design Analysis and Evaluation 24

5.1 Novelty 24

5.2 Design Considerations 24

5.2.1 Considerations to public health and safety 24

5.2.2 Considerations to environment 24

5.2.3 Considerations to cultural and societal needs 25

5.3 Investigations 26

5.3.1 Literature Review 26

5.3.2 Experiment Design 26

5.3.3 Data Analysis and Interpretation 26

5.4 Limitations of Tools 27

5.5 Impact Assessment 27

5.5.1 Assessment of Societal and Cultural Issues 27

5.5.2 Assessment of Health and Safety Issues 27

5.5.3 Assessment of Legal Issues 27
5.6 Sustainability and Environmental Impact Evaluation 28
5.7 Ethical Issues 28
6 Reflection on Individual and Team work 28
6.1 Individual Contribution of Each Member 28
6.2 Mode of TeamWork 28
6.3 Diversity Statement of Team 28
6.4 Log Book of Project Implementation 29
7 Communication 30
7.1 Executive Summary 30
7.2 User Manual 30
8 Project Management and Cost Analysis 31
8.1 Bill of Materials 31
9 Future Work 32
10 References 32

1 Abstract

This project intends to build an open-source DIY Self-Balancing Robot that is user-friendly and instructive. The robot, which resembles Segway-like vehicles, has intricate control mechanisms and practical uses.
Designing, constructing, and programming the robot are the major goals of the project. An inertial measurement unit (IMU), two motorized wheels, and a microcontroller with a Proportional-Integral-Derivative (PID) control algorithm for balance maintenance will be used.
The project offers a step-by-step tutorial for constructing the robot and customizing it, with an emphasis on accessibility, cost, and educational value. The robot's usefulness in personal transportation, environmental monitoring, and assistive technologies is also shown.
This DIY Self-Balancing Robot Project promotes creativity and technical growth by providing a practical, cheap chance to investigate robotics.

2 Introduction

Our project, which is concentrated on building a self-balancing robot similar to the Segway, provides an exciting hands-on introduction to the field of robotics. Our goal is to break down the intricate interaction between sensors, motors, and algorithms that a robot relies on to maintain its equilibrium while moving.
We created our concept with aficionados of different backgrounds in mind. It does not need specialist resources since it uses widely available components and open-source technologies.
We'll provide you a thorough, step-by-step tutorial for constructing and programming the robot along the way, including sensor selection and control algorithm implementation. In addition, we'll look at self-balancing robots' useful applications in the real world, highlighting things like personal transportation, environmental monitoring, and assistive technology.
Balancing the robot is always a difficult task since it requires precise control mechanism. We will use the concept of PID control in our project.
This DIY Self-Balancing Robot Project invites one to dive into the realm of robotics, with an emphasis on accessibility and education. We hope this project inspires your innovation and motivates you to delve into the captivating world of DIY robotics.

3 Design

3.1 Problem Formulation

3.1.1 Identification of Scope

The DIY Self-Balancing Robot Project's scope identifies its main areas of concentration and establishes precise parameters for its activities. It is crucial in determining the project's objectives, outputs, and restrictions. We outline the main aspects of the project's scope here:

Objective:
The main goal of this project is to design, build, and program a self-balancing robot utilizing inexpensive parts and open-source software.

Components and Resources:
The project will rely on readily accessible materials, including motors, sensors, and a microcontroller, ensuring that the resources required for building are within easy reach. Open-source software tools and libraries will be employed for programming, emphasizing accessibility and cost-effectiveness.

Technical Aspects:
The project will delve into the technical intricacies of the robot's construction, encompassing aspects such as sensor selection, assembly procedures, and programming. It will implement a Proportional-Integral-Derivative (PID) control algorithm to achieve and uphold balance.

Customization and Innovation:
This initiative actively encourages participants to personalize and enhance the robot's capabilities, fostering a spirit of customization and innovation. Collaboration within the DIY community is promoted to nurture innovation and knowledge sharing.

Practical Applications:
The project will venture into exploring the practical applications of self-balancing robots, ranging from personal transportation to environmental monitoring and assistive technology. Large scale implementation can be applied in healthcare, agriculture, Security and Surveillance etc.

Limitations:
Project's scope does not encompass the development of entirely bespoke or proprietary components. Furthermore, it may not cover advanced or highly specialized topics that

extend beyond the fundamental aspects of self-balancing robots.
In short, this project holds significant potential and a wide scope for exploration and development.

3.1.2 Literature Review

Self-balancing robots have risen in prominence as an intriguing field within the realm of robotics. This abbreviated review offers an overview of significant developments and educational importance in this area.

Historical Development:
The emergence of DIY self-balancing robots can be traced back to the open-source movement, which drew inspiration from trailblazers like Christian Faber's "Balancing Robot" project. Their roots can be found in Dean Kamen's introduction of the Segway in 2001.

Control Algorithms:
At the core of these robots lies the Proportional-Integral-Derivative (PID) control algorithm, enabling real-time adjustments for balance. Enthusiasts explore variations of PID to enhance performance.

Sensor Integration:
Inertial Measurement Units (IMUs), equipped with accelerometers and gyroscopes, are widely used for motion sensing. Some projects incorporate encoders to provide precise wheel feedback, thus enhancing stability.

Open-Source Community:
Online platforms like GitHub and robotics forums play a pivotal role in facilitating knowledge sharing and collaboration. The open-source philosophy encourages collective learning and active participation in projects.

Practical Applications:
Beyond their educational utility, self-balancing robots find practical applications in realms such as personal transportation, surveillance, and assistive technology. Their adaptability underscores their relevance in real-world scenarios.
DIY self-balancing robots represent a dynamic field with both educational and practical value. These projects democratize access to robotics knowledge and nurture a collaborative community. As technology continues to evolve, and the DIY community thrives, self-balancing robots are well-positioned to drive progress in both research and practical applications.

3.1.3 Formulation of Problem

The DIY Self-Balancing Robot Project is confronted with several significant issues and problem areas:

Achieving and Sustaining Stability:
A fundamental challenge is to ensure that the self-balancing robot maintains a stable and precise equilibrium during operation, even in challenging environments. This involves complex control algorithms and sensor integration.

Component Selection and Cost-Effectiveness:
Selecting appropriate components while keeping the project cost-effective is crucial. The project needs to identify and suggest readily available, affordable sensors, motors, and microcontrollers, making the robot accessible to a broad audience.

Programming and Algorithm Enhancement:
Developing the software and control algorithms for the robot is a substantial challenge. The project must address the intricacies of programming the Proportional-Integral-Derivative (PID) control algorithm for optimal performance and adaptability.

Customization and Innovation:
Encouraging participants to customize and improve the robot's capabilities is a central aspect of the project. The challenge is to foster a community that actively contributes ideas and innovations while staying aligned with the project's core goals.

Practical Use Cases and Field Testing:
Demonstrating the real-world applications of self-balancing robots is crucial. The project should explore practical scenarios like personal transportation, environmental monitoring, or assistive technology and ensure that the robot's performance aligns with these potential applications.

Safety and Dependability:
Addressing safety concerns and enhancing the reliability of self-balancing robots is essential. The project must provide guidelines and recommendations to mitigate risks and ensure the robot's safe operation.
In short, the Self-Balancing Robot Project confronts challenges related to stability, component selection, programming, educational value, customization, practical applications, budget constraints, safety, and reliability. These challenges are pivotal in achieving the project's objectives and creating an inclusive, educational, and innovative self-balancing robot.

3.1.4 Analysis

The DIY Self-Balancing Robot Project combines robotics, education, and innovation. This analysis explores the project's key aspects, its strengths and hurdles, and its broader implications.

Crucial Project Elements:
Educational Focus:
The project excels in its educational value, offering practical learning in robotics, control systems, and programming through clear documentation.

Accessibility and Affordability:
By using available components and open-source tools, the project ensures widespread participation without excessive costs.

Practical Applications:
Demonstrating real-world uses highlights the adaptability of self-balancing robots in personal transportation and assistive technology.

Crucial possible impacts:
Educational Impact:
The project simplifies intricate robotics concepts, improving STEM education for a diverse audience.

Innovation Catalyst:
It sparks innovation by fostering customization and community collaboration, advancing self-balancing robot technology.

Real-World Demonstrations:
By showcasing practical applications, the project motivates participants to explore broader implementation possibilities.

Challenges and Weaknesses:
Complexity:
Building and programming self-balancing robots can be intimidating, especially for beginners.

Resource Limitations:
Component accessibility may be constrained in some regions, necessitating alternative solutions.

Safety Concerns:
The project must prioritize safety guidelines to mitigate potential risks linked to self-balancing robots.

Wider Influence:
The Self-Balancing Robot Project extends beyond its immediate goals, promoting robotics education, open-source collaboration, and innovation. It empowers individuals to delve into robotics, fostering creativity and problem-solving abilities. By demonstrating practical applications, it inspires participants to envision real-world solutions through robotics.

3.2 Design Method

In this section description of components software used in this project, purpose of their use and cause behind choosing these components are discussed.

1) Controller

a) Arduino Nano (ATmega328p)

In our project we used breadboard in the main structure of the self-balancing robot since we failed to use veroboard or pcb. Necessarily we need a small, compact and breadboard friendly microcontroller. Arduino Nano meets all the purposes. Consequently we used Arduino Nano as controller in the main structure of the robot.

Figure 3.2.1(a): Arduino Nano

Arduino Nano is one type of microcontroller board with 16 digital pins. It can be built with a microcontroller like Atmega328. It doesn't have any DC jack so that the power supply can be given using a small USB port or directly connected to the pins like VCC & GND.

b)Arduino Uno

To control the movement in different direction of the robot we need another controller circuit. Since to build joystick with Arduino we did not face any constraint of size and the fact that Arduino UNO is flexible, and easy-to-use programmable open-source microcontroller board to meet the purpose we used Arduino Uno.

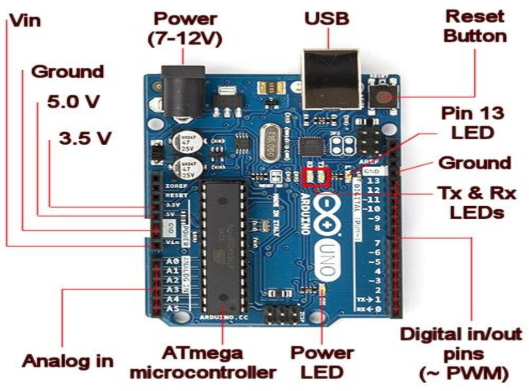


Figure 3.2.1(b): Arduino Uno

2) NEMA 17 Stepper Motor

For choosing motor, precision and accuracy were our priorities since we need a precise and accurate control of rotational speed of the motor. We compared stepper and dc motor in this case and choose NEMA 17 stepper motor to be used in the project. Due to mechanical friction and electrical differentials DC motor shows poor performance than stepper motor. Also stepper motor gives more accurate rotational speed and shows smooth behaviour during speed control.

Technical Specifications

- Step angle accuracy: +/- 5%

  (Full step, no load)

- Resistance accuracy: +/- 10%
- Inductance accuracy: +/- 20%
- Phases:2
- Type: Bipolar Stepper Motor
- Current / Phase:1.8A
- Size: 42x42x48m
- Holding torque 48 N-cm

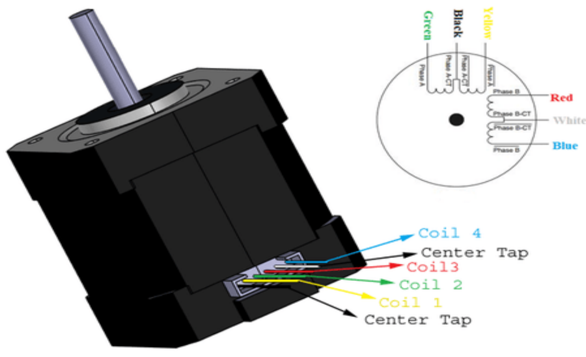


Figure 3.2.2:PindescriptionNEMA17

Stepper motors are electromechanical devices that convert digital pulses into rotational motion. NEMA 17 stepper motor has a 1.8° step angle. There are 200 steps or revolutions per angle. Each phase of the motor type can draw a voltage of 12V when it is in operation.

3) A4988 Stepper Motor Driver Module

We have used two A4988 IC to drive two stepper motors. The A4988 is a bipolar stepper motor driver that is capable of driving stepper motors. Pulse generator, current regulator, micro-stepping indexer, and output driver are the four primary parts of the A4988 stepper motor driver. The micro stepping indexer receives the step pulses required by the stepper motor from the pulse generator. The current flowing through the stepper motor coils is regulated by the current regulator after being controlled by the micro stepping indexer.

Specifications:

- Voltage range: 8V – 35V
- Continuous current per phase: 1 A2
- Maximum current per phase: 2 A
- Logic input voltage: 3V – 5.5V
- Micro step resolutions: full, 1/2, 1/4, 1/8, and 1/16

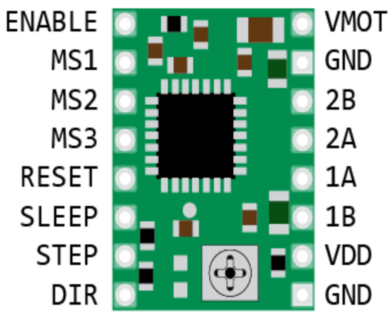
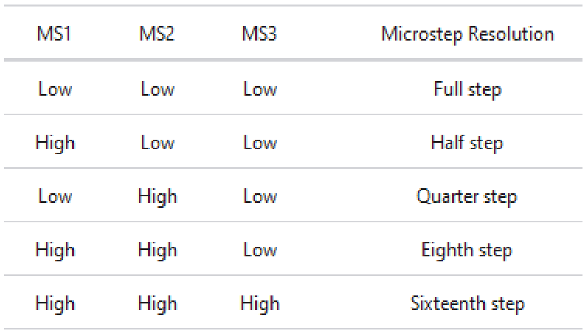


Figure 3.2.3(a): A4988 Stepper Motor Driver Module



MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

Figure3.2.3 (b): Five different step- resolution

This module provides simplified step control with five different step resolutions: full-step, half-step, quarter-step, eighth-step, and sixteenth-step. It also provides current control by setting the maximum current output with a potentiometer, which lets us setting a reference voltage. The A4988 driver IC can give roughly 1A per coil without overheating, while having a maximum current rating of 2A per coil. A heat sink or other cooling technique is needed to generate more than 1A per coil.

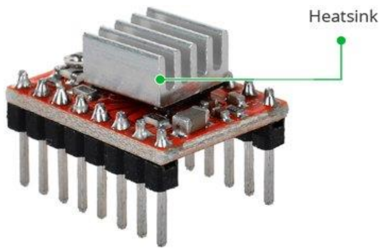


Figure 3.2.3 (c): A4988 IC with heat sink

4) MPU6050 Accelerometer and Gyroscope Module

For our project sensing acceleration and three dimensional position of the robot with substantial accuracy is utmost necessary. The gyroscope used in this case is MPU6050. As the angle of the robot changes, the Arduino sends the signal to the motor driver to direct the motors in such a way that it counters the fall of the robot**.** This in turn helps the robot to balance itself upright.

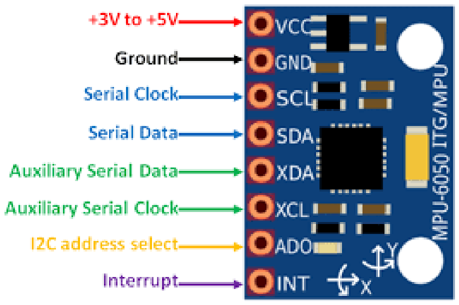


Figure 3.2.4: MPU6050 Accelerometer and Gyroscope Module

5) HC-05 Bluetooth Module

The HC-05 is Bluetooth module designed for transparent wireless serial communication. We have used two Bluetooth module in our project. One HC-05 is used as a master placed at controller (with joystick and Arduino) section and the other as a slave placed at the main body structure of the robot.

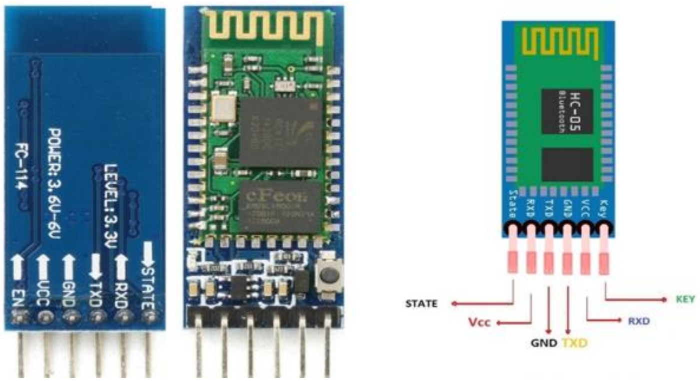


Figure 3.2.5: HC-05 Bluetooth Module

6) Wheels

Two wheels are placed at the shaft of the motors using coupler. We choose the diameter of the wheel so that it is compatible with the height of the body and it can move the structure precisely.

Specifications:

- Material: Brass
- Total length: 18mm
- Hexagonal edge to edge distance: 12mm
- Screw hole diameter: 4mm
- wheel diameter:65mm

Figure 3.2.6: Wheel

7) MDF Board, Threaded Rods, Nuts & Bolts, Angle

To make the structure lighter and strong we use MDF board as its building element. For same reason we use threaded rods (SS) and aluminium angle.

Figure 3.2.7 (a): MDF board

Figure 3.2.7 (b): Threaded Rods

Figure 3.2.7(c): Nuts & Bolts

Figure 3.2.7(d): Aluminium Angle

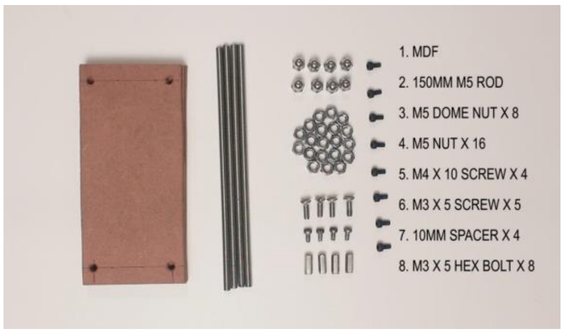


Figure 3.2.7(e**):** MDF Board, Threaded Rods, Nuts & Bolts with Specification

8) Battery

A battery is placed at top of the structure to make a higher inertia and eventually to get a high torque. Battery is chosen as per our requirement.

Description
Capacity:1500mAh.
Voltage:11.1V.
Continuous discharge rate: 25C.
Dimension: 23.5 x 35 x 68 mm
Weight: 112g.
Burst: 40C
Net Weight: 130g
Cell Unit: 3 Cells
.

Figure3.2.8: Lipo Battery

3.3 Circuit Diagram

1) Checking baud rate of HC-05 Bluetooth module

First of all we need to set the baud rate of the HC-05 using Arduino Uno. We keep the baud rate as its default value (9600) and checked it on the serial monitor of Arduino IDE.

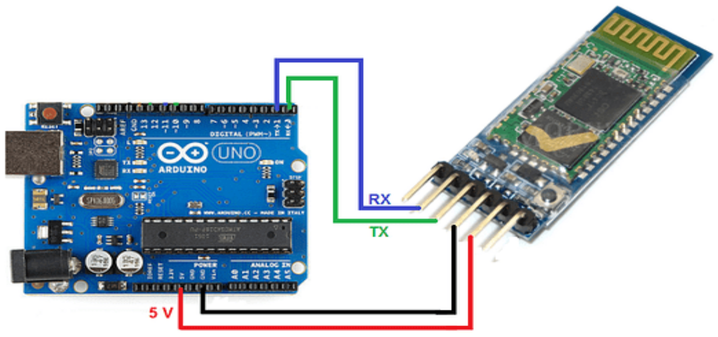


Figure3.3.1: HC-05 connected with Arduino

2) Controller circuit (with Arduino, joystick and HC-05)

This unit is used to control the movement of the robot manually.

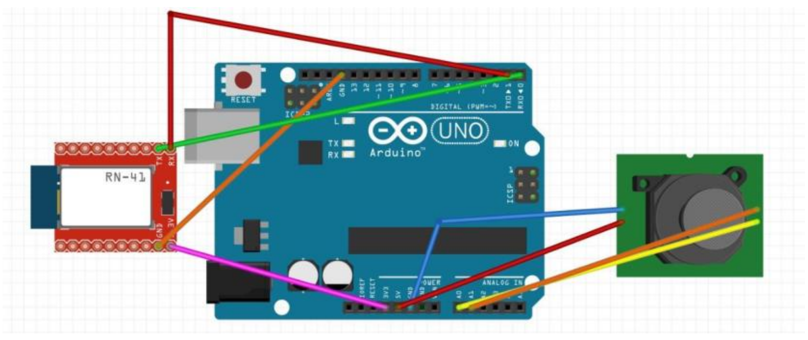


Figure3.3.2: Controller Circuit

3) Testing A4988 and setting Vref

In this subsection we illustrated the circuit diagram for testing and setting the reference voltage of A4988 Driver IC. An A4988 driver is verified to be ok if the motor of Figure3.3.3 (a) runs accordingly.

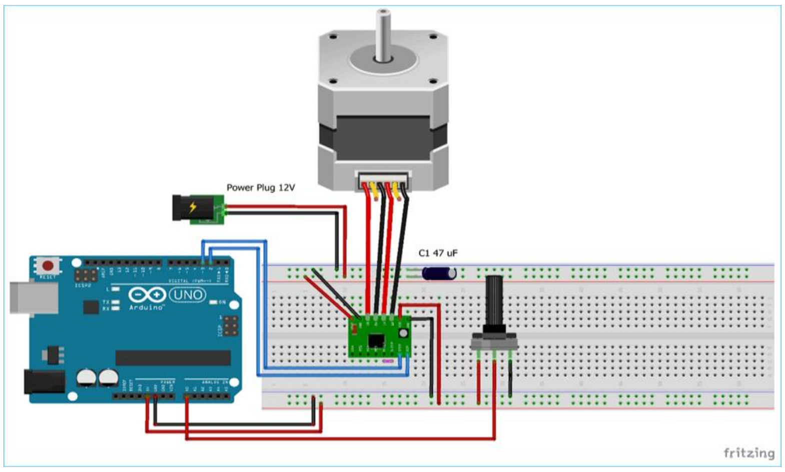


Figure3.3.3 (a): Circuit Diagram for testing A4988 driver IC

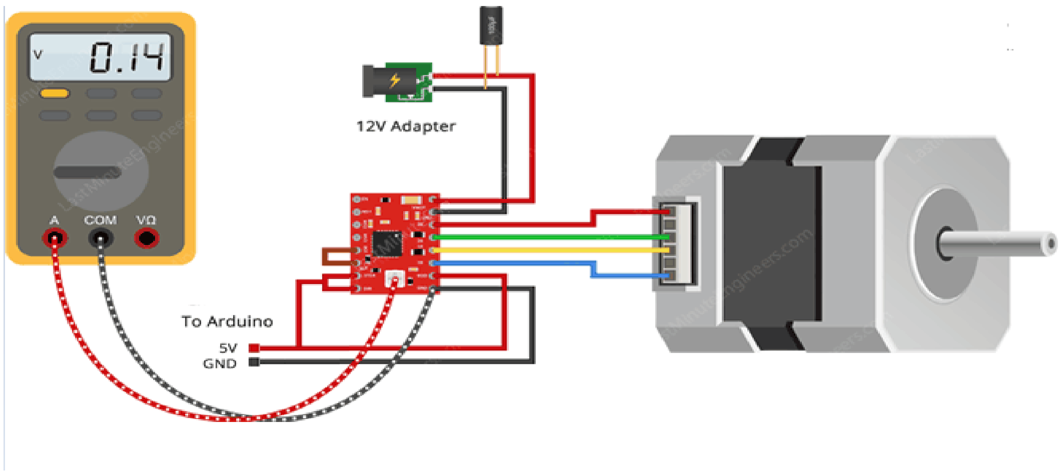


Figure3.3.3 (b): Circuit Diagram for setting Vref of A4988 driver IC

As Figure3.3.3 (b) shows that by placing multimeter we can measure the voltage between the indexer pin and ground in other words the reference voltage for limiting the current.

4) Overall circuit diagram (self-balancing robot)

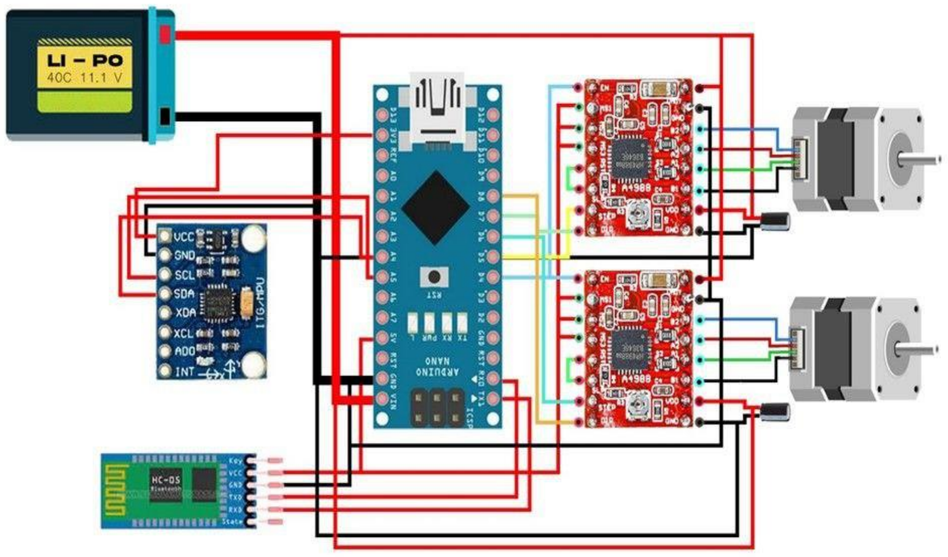


Figure 3.3.4(a): Circuit Diagram for self-balancing Robot

MPU 6050 Connection

The MPU6050 uses the I2C module for communication. The I2C pins are 5V tolerant, hence we use them with the Arduino without any level converters. The SCL pin is connected to the SCL pin (A5) of the Arduino. Similarly, the SDA pin is connected to the SDA pin (A4) of the Arduino.

- **VCC**

Provides power for the module, connected to the 5V pin of the Arduino.

- **A4 – SDA**

Serial Data Used for transferring Data through I2C communication.

- **A5 – SCL**

Serial Clock Used for providing clock pulse for I2C Communication.

A4988 Motor driver and NEMA 17 stepper motor connection:

- **DIR pin (A4988)**
  The DIR pin will control the rotation direction.

- **Step pin(A4988)**
  The micro steps of the motor are controlled by STEP input. The motor is driven by each HIGH pulse delivered to this pin in accordance with the number of micro steps selected by the micro step selection pins.

- **MS (micro step pins)**

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

Connection to Arduino port

- D5 – STEP1 (PORTD 5)
- D6 – STEP2 (PORTD 6)
- D7 – DIR1 (PORTD 7)
- D8 – DIR2 (PORTB 0)
- D4 – ENABLE (for both)

Connection to Stepper Motor

- 2B-Red wire
- 2A-Green
- 1A-Yellow
- 1B-Indigo

A 47 µf capacitor is used to protect the board from voltage spikes.

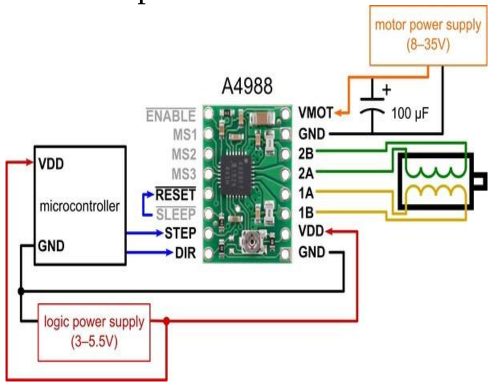
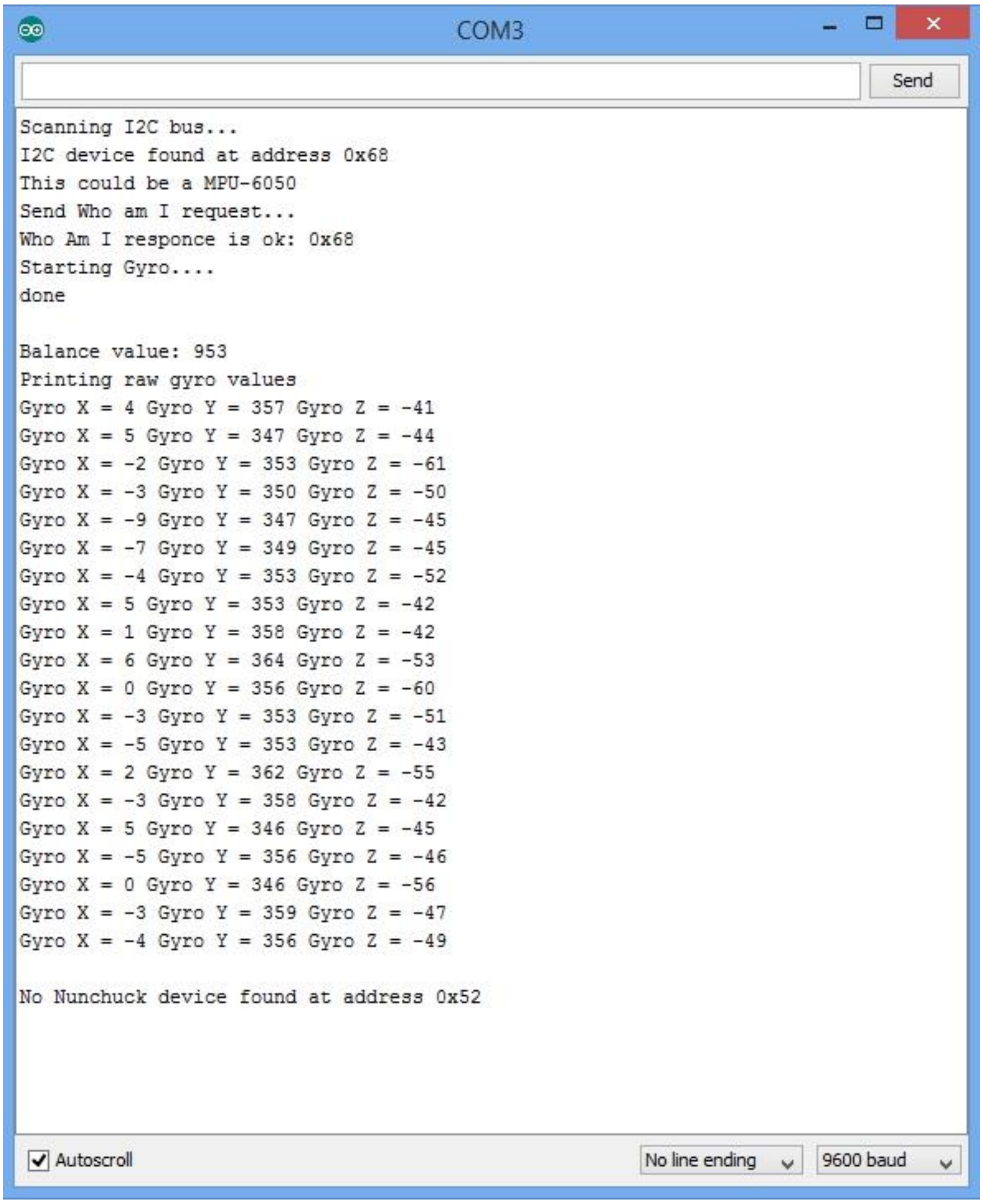


Figure 3.3.4(b): A4988 pin connection for full stepping

3.4 Simulation Model

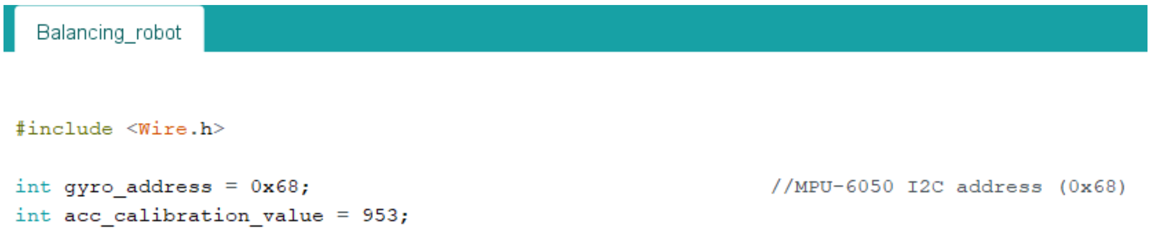
Our model was mainly PID controlled bot. We checked every component through and through by various simulations through our hardware test program that was uploaded to GitHub. (Link provided on section 3.7)

Hardware testing was done for the gyroscope MPU 6050 at first. We got properly responding gyroscope. Here's the snippet from the gyroscope testing. We got a balance value which we will be using in the final robot balancing code for calibration.



```
Scanning I2C bus...
I2C device found at address 0x68
This could be a MPU-6050
Send Who am I request...
Who Am I responce is ok: 0x68
Starting Gyro....
done

Balance value: 953
Printing raw gyro values
Gyro X = 4 Gyro Y = 357 Gyro Z = -41
Gyro X = 5 Gyro Y = 347 Gyro Z = -44
Gyro X = -2 Gyro Y = 353 Gyro Z = -61
Gyro X = -3 Gyro Y = 350 Gyro Z = -50
Gyro X = -9 Gyro Y = 347 Gyro Z = -45
Gyro X = -7 Gyro Y = 349 Gyro Z = -45
Gyro X = -4 Gyro Y = 353 Gyro Z = -52
Gyro X = 5 Gyro Y = 353 Gyro Z = -42
Gyro X = 1 Gyro Y = 358 Gyro Z = -42
Gyro X = 6 Gyro Y = 364 Gyro Z = -53
Gyro X = 0 Gyro Y = 356 Gyro Z = -60
Gyro X = -3 Gyro Y = 353 Gyro Z = -51
Gyro X = -5 Gyro Y = 353 Gyro Z = -43
Gyro X = 2 Gyro Y = 362 Gyro Z = -55
Gyro X = -3 Gyro Y = 358 Gyro Z = -42
Gyro X = 5 Gyro Y = 346 Gyro Z = -45
Gyro X = -5 Gyro Y = 356 Gyro Z = -46
Gyro X = 0 Gyro Y = 346 Gyro Z = -56
Gyro X = -3 Gyro Y = 359 Gyro Z = -47
Gyro X = -4 Gyro Y = 356 Gyro Z = -49

No Nunchuck device found at address 0x52
```

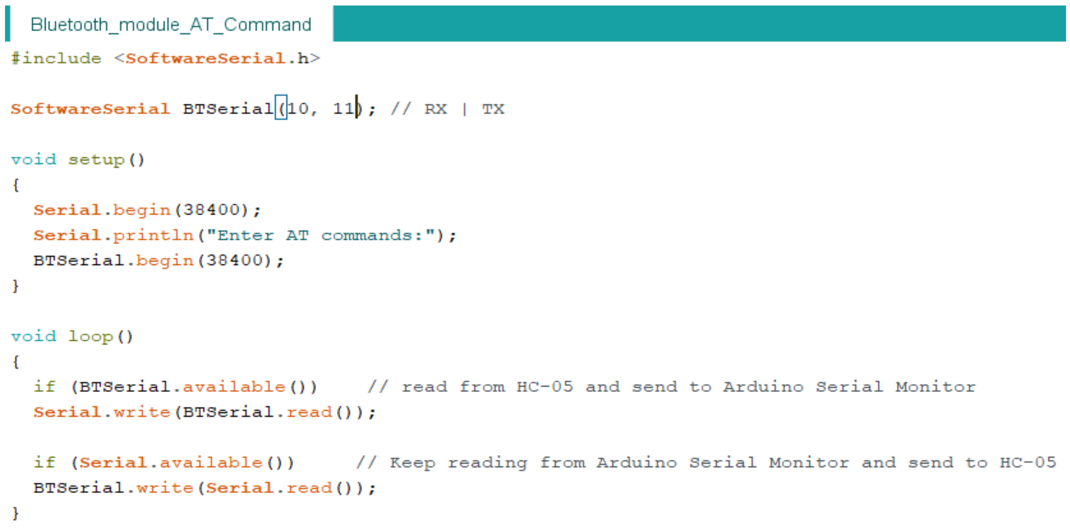
We then uploaded with entering the gyroscope balancing value the self balancing robot code on the nano that will be driving the bot circuit.



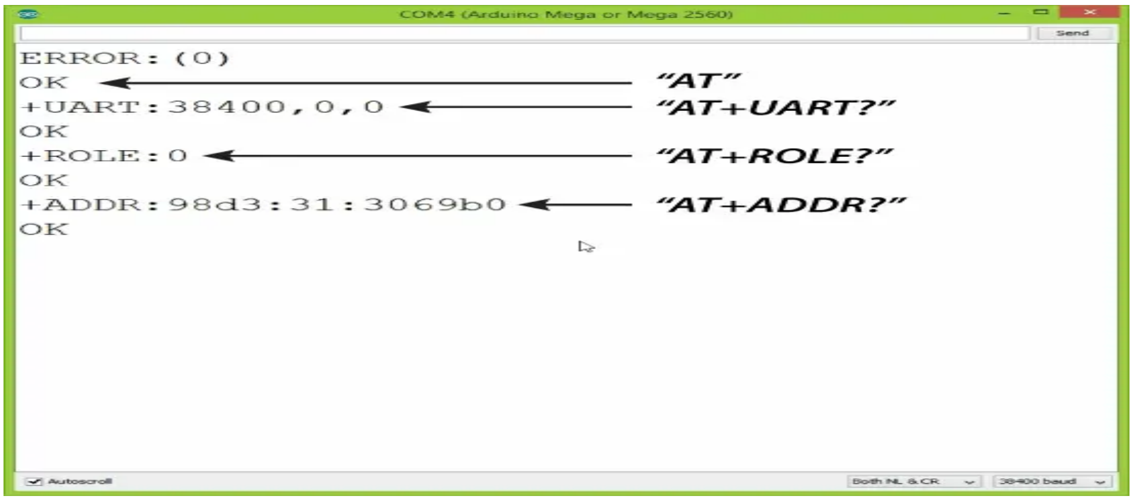
```
#include <Wire.h>

int gyro_address = 0x68;                                    //MPU-6050 I2C address (0x68)
int acc_calibration_value = 953;
```

We also changed our Circuit Board Bluetooth hc-05 module as a slave and the controller hc-05 module as a master by connecting it separately to the Arduino board and pc through our serial monitor of Arduino IDE.

Code looks like this for entering AT command to change the Bluetooth settings. We also changed the baud rate of both Bluetooth module to 9600 to communicate between them.

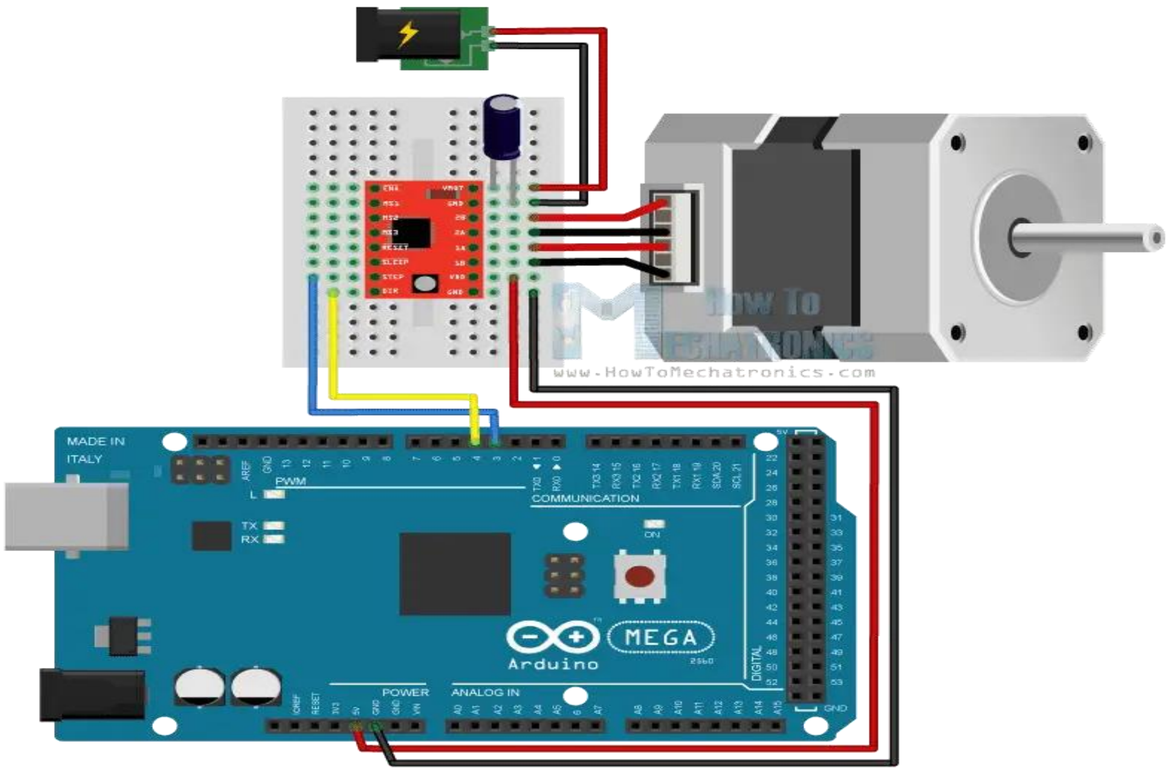


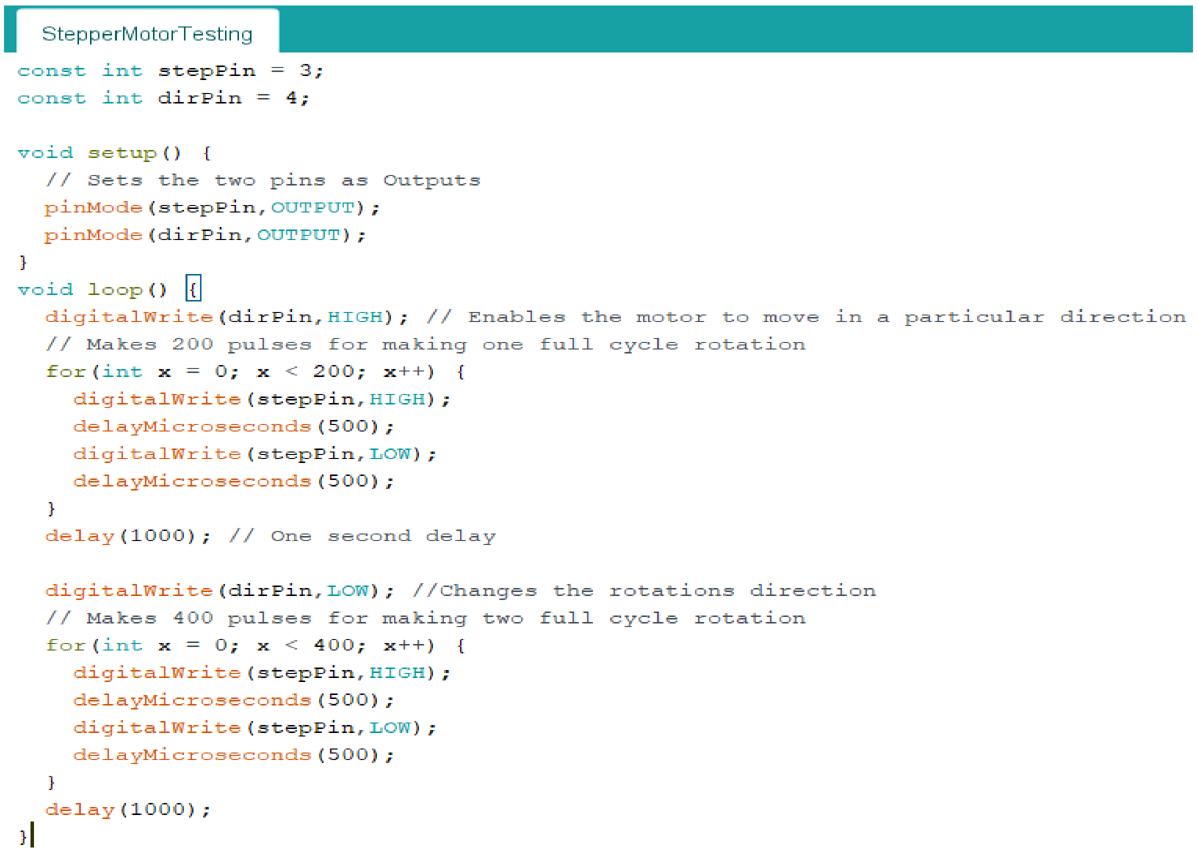
```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
  Serial.begin(38400);
  Serial.println("Enter AT commands:");
  BTSerial.begin(38400);
}

void loop()
{
  if (BTSerial.available())    // read from HC-05 and send to Arduino Serial Monitor
  Serial.write(BTSerial.read());

  if (Serial.available())     // Keep reading from Arduino Serial Monitor and send to HC-05
  BTSerial.write(Serial.read());
}
```



```
ERROR: (0)
OK                                      "AT"
+UART:38400,0,0                         "AT+UART?"
OK
+ROLE:0                                 "AT+ROLE?"
OK
+ADDR:98d3:31:3069b0                    "AT+ADDR?"
OK
```

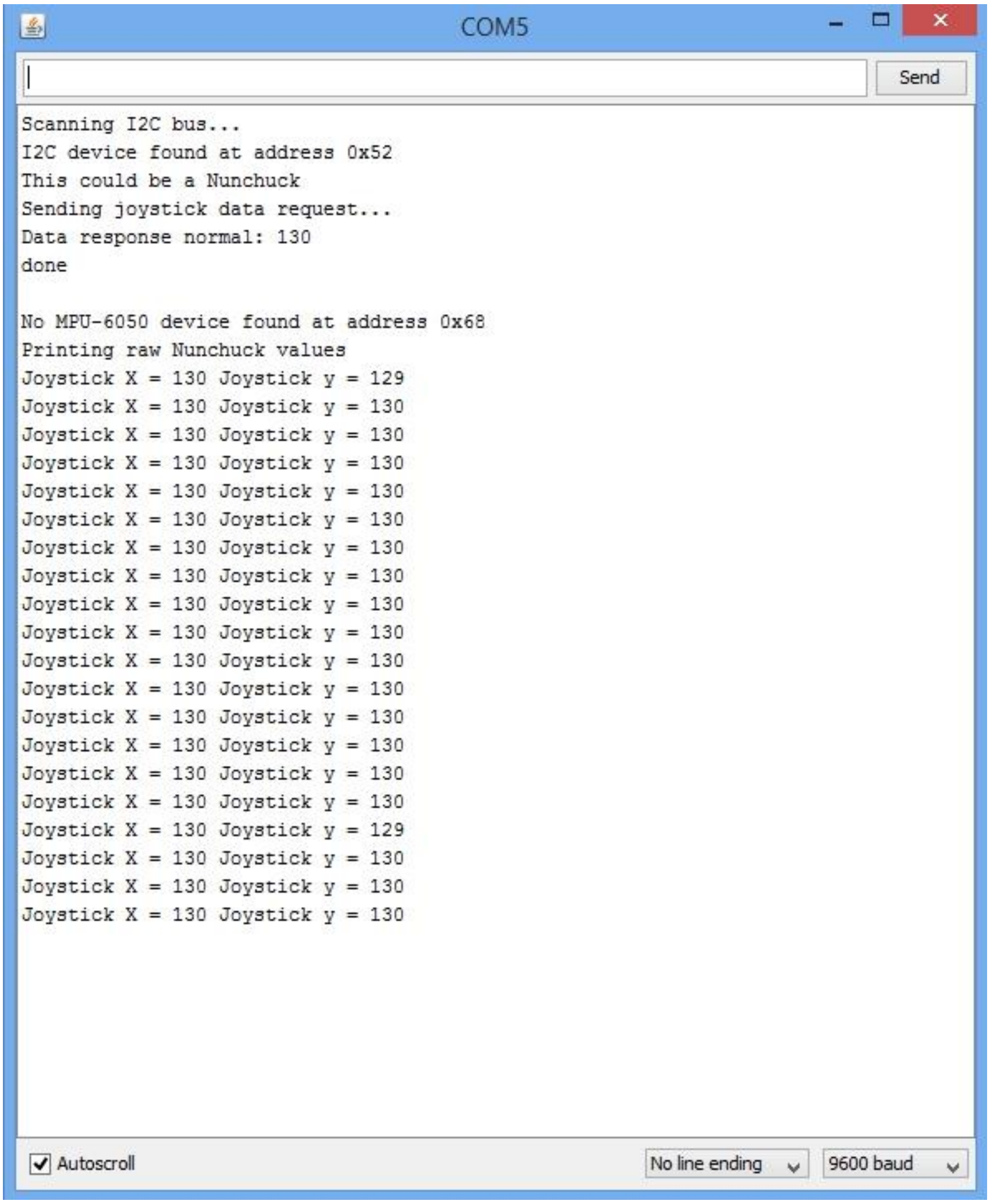
We also tested the Nema 17 Stepper Motor using Arduino separately.





```
const int stepPin = 3;
const int dirPin = 4;

void setup() {
  // Sets the two pins as Outputs
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
}
void loop() {
  digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction
  // Makes 200 pulses for making one full cycle rotation
  for(int x = 0; x < 200; x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);
  }
  delay(1000); // One second delay

  digitalWrite(dirPin,LOW); //Changes the rotations direction
  // Makes 400 pulses for making two full cycle rotation
  for(int x = 0; x < 400; x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);
  }
  delay(1000);
}
```

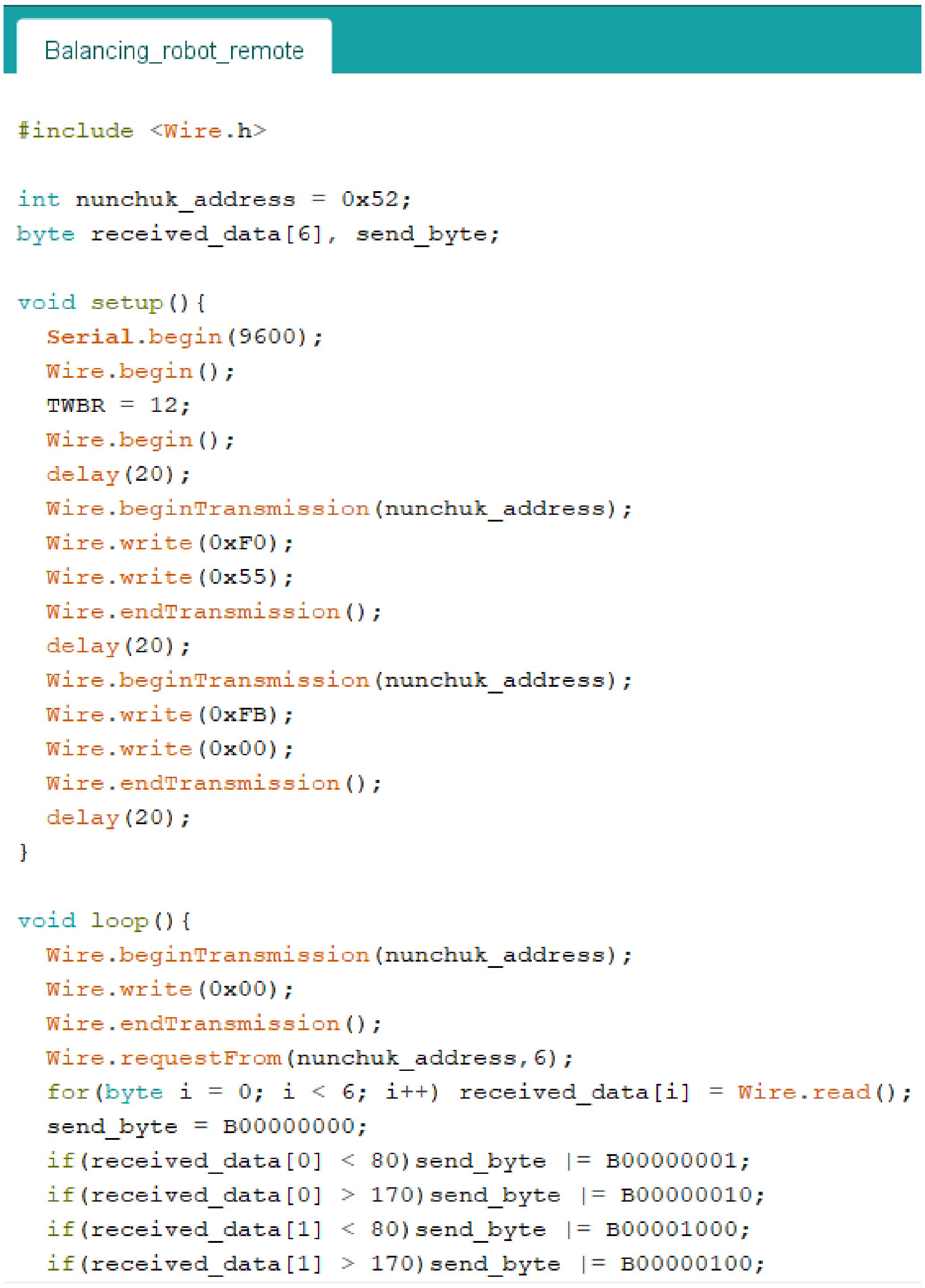
Then we connected the controller and tested the controller. here’s the testing snippet.



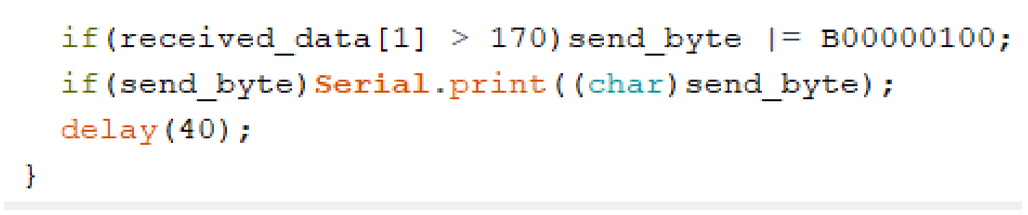
```
Scanning I2C bus...
I2C device found at address 0x52
This could be a Nunchuck
Sending joystick data request...
Data response normal: 130
done

No MPU-6050 device found at address 0x68
Printing raw Nunchuck values
Joystick X = 130 Joystick y = 129
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 129
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
Joystick X = 130 Joystick y = 130
```

☑ Autoscroll — No line ending — 9600 baud

Then we uploaded the controller code. We used this code.



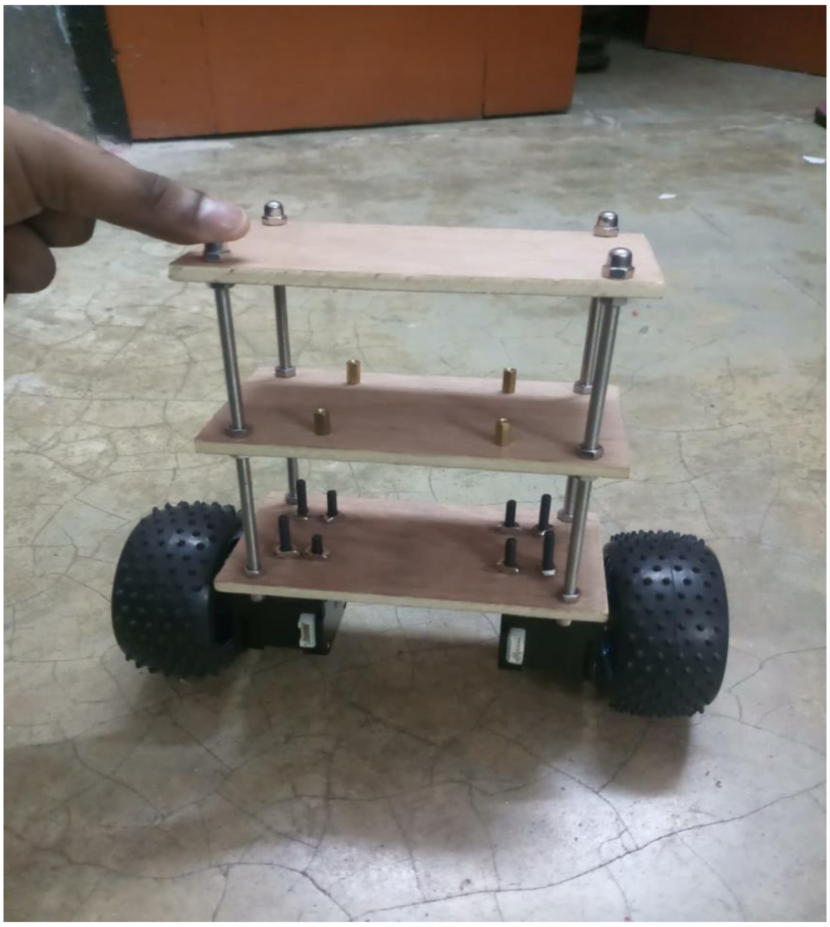
```
#include <Wire.h>

int nunchuk_address = 0x52;
byte received_data[6], send_byte;

void setup(){
  Serial.begin(9600);
  Wire.begin();
  TWBR = 12;
  Wire.begin();
  delay(20);
  Wire.beginTransmission(nunchuk_address);
  Wire.write(0xF0);
  Wire.write(0x55);
  Wire.endTransmission();
  delay(20);
  Wire.beginTransmission(nunchuk_address);
  Wire.write(0xFB);
  Wire.write(0x00);
  Wire.endTransmission();
  delay(20);
}

void loop(){
  Wire.beginTransmission(nunchuk_address);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(nunchuk_address,6);
  for(byte i = 0; i < 6; i++) received_data[i] = Wire.read();
  send_byte = B00000000;
  if(received_data[0] < 80)send_byte |= B00000001;
  if(received_data[0] > 170)send_byte |= B00000010;
  if(received_data[1] < 80)send_byte |= B00001000;
  if(received_data[1] > 170)send_byte |= B00000100;
```



```
  if(received_data[1] > 170)send_byte |= B00000100;
  if(send_byte)Serial.print((char)send_byte);
  delay(40);
}
```

3.5 Hardware Design

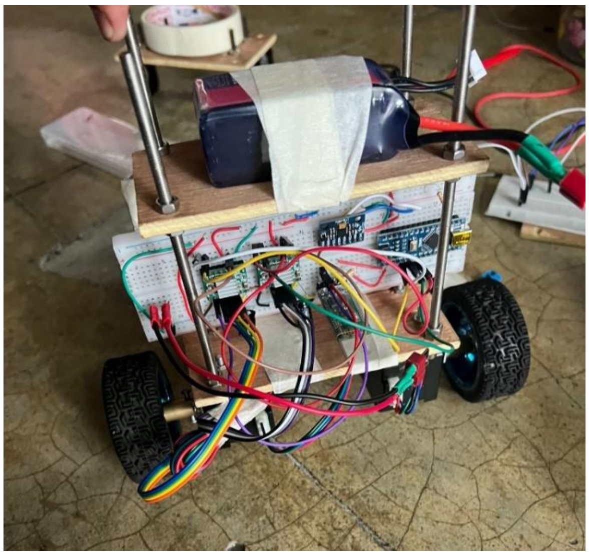
Initial Structure:

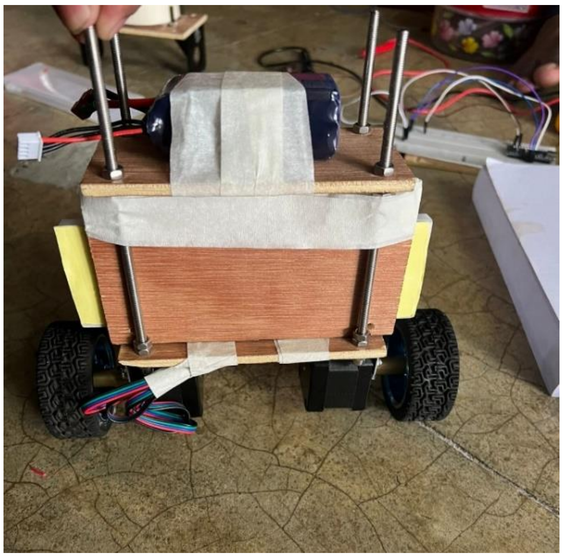


At first, we cut 3 pieces of MDF board size 130×65 mm, cut 4 threaded rods of length 150mm, drilled the board appropriately, and with the help of screws and nuts, we built a symmetric structure as shown in the picture. Our initial structure had 3 layers. In the bottom layer, we intended to place the lipo battery, in the middle layer we created place for the PCB or Veroboard circuit and upper layer was kept free as we wanted to place some object at the top layer after balancing is done.

We tested our circuit in the breadboard and after that designed a PCB layout for the working circuit (PCB design is included in the next part of the report). But unfortunately, our PCB didn't work as expected. Then we constructed the circuit in the Veroboard (Veroboard design is also given in the next section). Veroboard circuit worked almost perfectly but it gave a small problem of not receiving data through bluetooth module. We had no other option than to implement our circuit to breadboard. We built the circuit in the breadboard and placed it into the 2nd layer of our structure. After calibrating, and powering the circuit, the wheels were moving. But the robot was not balancing. We figured out that for balancing, the center of gravity needs to be in the upper portion of the structure. An upper center of gravity increases the stability of the structure and provides a lower tendency to tip over. Also, the wheels we used were a bit heavy.

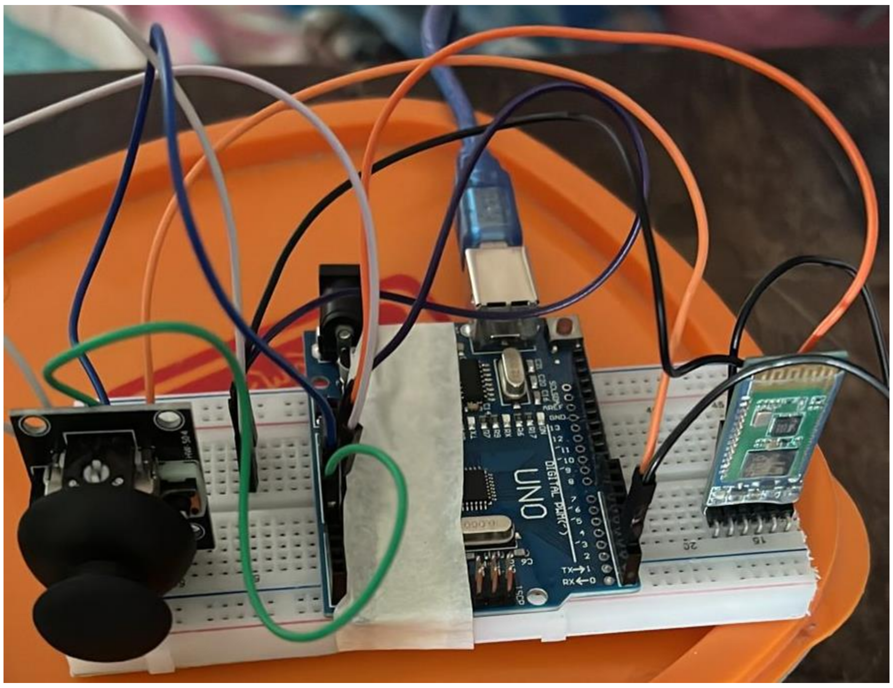
We than changed our structure.
Final Structure:





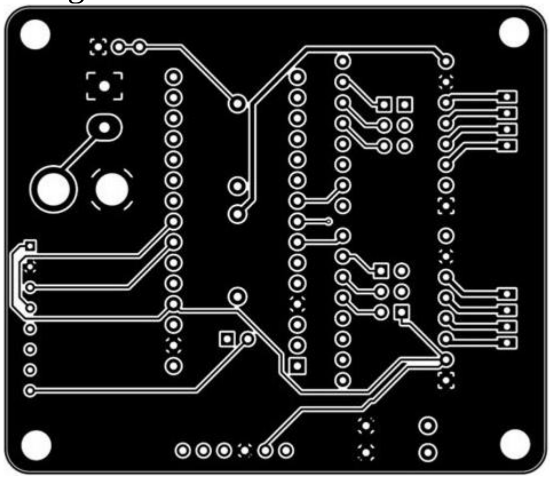
In the final structure, we built a two-layered structure in which battery is placed in the top layer, and circuit is placed vertically as shown in the picture.

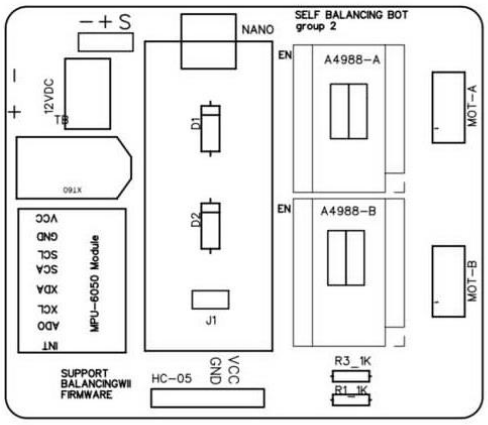
Joystick controller setup:



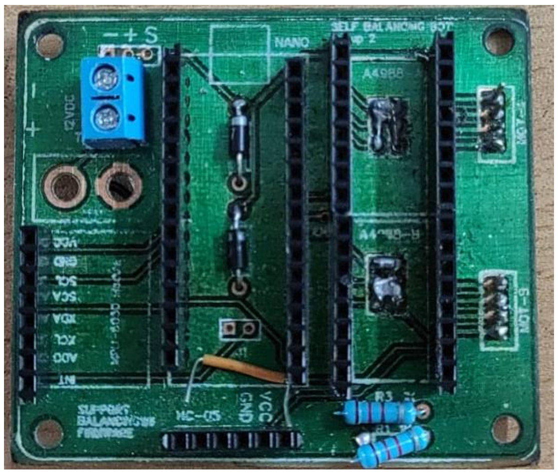
3.6 PCB Design

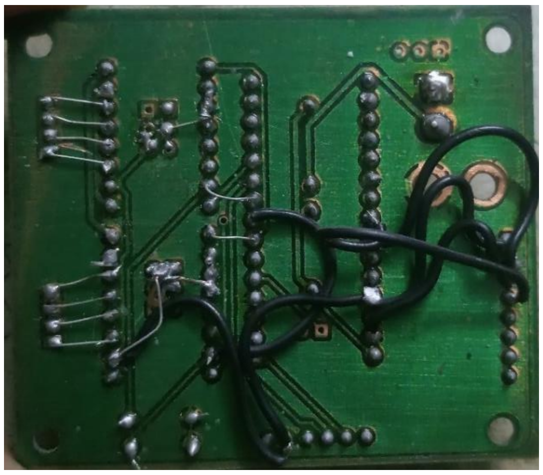
Design:



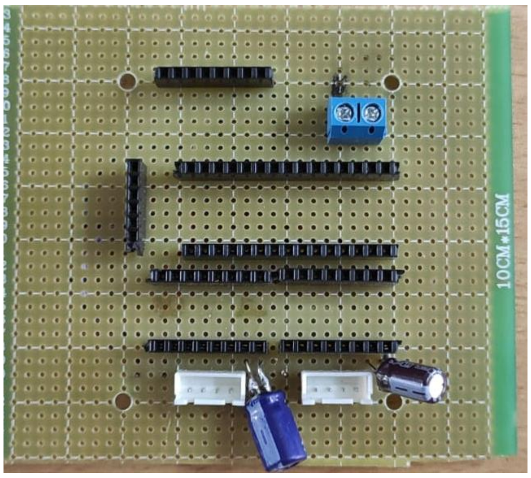


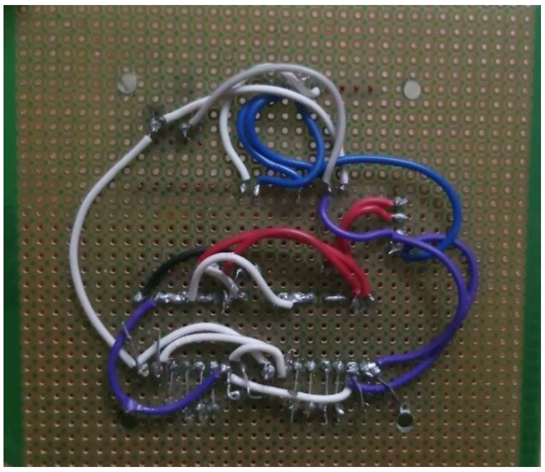
Hardware: PCB





Veroboard





3.7 Full Source Code of Firmware

GitHub Link: https://github.com/rowatulrafi/Self_Balancing_Bot

4 Implementation

4.1Description

We individually tested each of our ICs. Then put it into the initially implemented PCB. But it didn’t work accordingly. Then we implemented we full circuit into Veroboard. But again unfortunately it didn't work either. We finally implemented the circuit in breadboard.

4.2 Experiment and Data Collection

While implementing the circuit on breadboard we tried calibrating circuit components. Specially MPU 6050 and A498 drivers. MPU6050 calibration value is inserted into main program of the Arduino nano which will help on balancing the bot.

A498 drivers Vref is set to 0.76V after calculating Vref = Rated current * 8 * 0.8 * Rcs. Here rated current is found from Nema 17 stepper motor datasheet as 1.7A and Rcs is found from A498 motor driver datasheet. We also tested the motor afterwards with A498 in full step to see the motor actually working.

Finally, our bot had to be controlled through PID controller. And for that we needed to variate the P,I and D-gains in the main program of the Arduino.

4.3 Data Analysis

Our Calibration value for MPU6050 was 953 and this value sufficiently worked. Our stepper driver Vref=0.76V was sufficient enough to power the stepper motor with 1/4th microstepping. Finally, our P-gain =15 , I-gain =1.5 , D-gain =30 which worked sufficiently well. Our Bluetooth module both has 9600bps transmission rate.

4.4 Results

Upon implementing the above data for our project we found a well balanced robot. But unfortunately due to manufacturing issue our stepper motor failed after some running. Because of this happening late, just before the deadline we couldn’t replace our motor to showcase the self balancing robot

5 Design Analysis and Evaluation

5.1 Novelty

We used a 5 pins Joystick Breakout Module to control the direction of our robot. The Joystick module is similar to analog joystick found in gamepads. It is made by mounting two potentiometer at 90 degree angles.

5.2 Design Considerations

5.2.1 Considerations to public health and safety

When working on a DIY self-balancing robot project, it's essential to consider public safety and health. While these robots can be exciting and educational, they can also pose potential risks if not designed, built, and operated with safety in mind. Here are some key considerations:

Electrical Safety:
It was ensured that all electrical components and connections meet safety standards and codes. We followed guidance on how to safely handle batteries and avoid short circuits or overheating.

Mechanical Safety:
Mechanical safeguards have been implemented to prevent users from coming in contact with moving parts or sharp edges. We ensured that the robot's balance system is well-calibrated to prevent sudden falls or uncontrolled movements.

Fire Safety:
Fire-resistant materials were where appropriate, especially when working with batteries and electrical components.

Testing and Quality Assurance:
We intend to thoroughly test their robots in controlled environments before operating them in public spaces.
By prioritizing safety and health considerations throughout the DIY self-balancing robot project, you can ensure that it serves as an educational and innovative endeavor without compromising the well-being of builders, users, or the public.

5.2.2 Considerations to environment

When undertaking a DIY self-balancing robot project, it's essential to take into account environmental concerns and aim to reduce any adverse effects on the environment. Here are some key environmental factors to think about in such a project:

Material Choice:
We opted for eco-friendly materials with a minimal ecological impact, avoiding hazardous or non-recyclable options whenever feasible.

Energy Efficiency:
We tried to design the robot to consume energy efficiently by selecting greener power sources like rechargeable batteries or energy-efficient components.

Recycling and Disposal:
We tried to devise plan for responsible recycling or disposal of electronic components and materials when the robot's life cycle ends, and ensure proper battery disposal.

End-of-Life Planning:
We consider the robot's fate when it reaches the end of its usefulness; explore options for repurposing, recycling, or environmentally responsible disposal of its parts.

Local Impact:
We assessed the local environmental impact of your project, especially if we plan to showcase the robot in public areas, ensuring it does not harm local ecosystems or wildlife.
By integrating these environmental considerations into your DIY self-balancing robot project, you can work toward reducing its environmental footprint and advocating responsible practices within the fields of robotics and engineering.

5.2.3 Considerations to cultural and societal needs

When engaging in a DIY self-balancing robot project, it's we took into account cultural and social considerations to ensure that the project is inclusive for all. Here are some key points to think about:

Accessibility:
We intend to make sure that the robot and project materials are accessible to people with various abilities, including those with disabilities. We have considered features like clear instructions and options for alternative control methods.

Community Engagement:
We intend to involve the local community or communities with specific interests in the project. Seek their input and feedback to ensure that the project meets their needs and interests.

Concerns Considerations:
We addressed concerns that may arise from the project's impact on society, such as issues related to privacy, surveillance, or potential job displacement due to automation.

Community Benefits:
We intend to highlight how the project can benefit the community, such as by offering new educational opportunities or addressing specific local needs.
By taking cultural and social needs into consideration in your DIY self-balancing robot project, we can create a more inclusive and socially responsible endeavor that resonates with a broader range of individuals and communities while respecting their unique cultural and social backgrounds.

5.3 Investigations

5.3.1 Literature Review

Gyroscope measures the deviation of the robot from the calibrated position. It then sends the signal to Arduino nano. Arduino generates required control signal and operates the motors with the help of motor controller A4988. When robot tips in the forward direction, wheels rotate even faster in the forward direction to prevent the fall of the robot. Similarly, when robot tips in the backward direction, wheels rotate faster in the backward direction preventing falling of the robot.

5.3.2 Experiment Design

We have used 1 arduino nano, 1 Gyroscope MPU 6050, 2 motor driver module A4988, 2 NEMA 17 stepper motor, 11.1V 1500mAH Lipo battery, 1 arduino uno, 1 joystick.

5.3.3 Data Analysis and Interpretation

Robot is calibrated and reference data from gyroscope is set in the program and then P,I,D values are changed in trial and error method to obtain balancing. Reference voltage in the driver module A4988 is set according to the specification and necessary formula.

5.4 Limitations of Tools

The first Arduino nano we used found to have some error in it. The Vin pin of that Arduino was not working so we had to change the Arduino. The HC-05 Bluetooth module we used was not responding initially. So we had to buy anther 2 HC-05 module to successfully receive and transmit data. The first 2 motor driver A4988 we bought, found to be damaged. We had to rebuy the pair of A4988 driver modules. Finally the motor was damaged just before the demonstration of our project. The maximum current rating of the motors were 1.3A, but accidentally current flow in the motor exceeded the limit and reached around 1.5A. Stepper motor got mechanically damaged by this.

5.5 Impact Assessment

5.5.1 Assessment of Societal and Cultural Issues

By using this self-balancing robot, we can contribute greatly to societal and cultural issues.
This project is easily accessible for all walks of the people in the society. The materials are sufficiently available and easy to understand manual will be given.
One of the goal of the project was to attract interested people irrespective of age, race, gender.
Different concerns like security, privacy of the project was analyzed and solved accordingly and people can use it without any concern.
This project can make a significant impact in the manufacturing sector where mass use of this project would reduce risk, increase efficiency.

5.5.2 Assessment of Health and Safety Issues

This project didn’t use any fossil fuel, nor did it emit any gas at all. So harmful element isn’t a concern in this project.
In this project no physical contact is required so there was given serious consideration to reduce the electrical shock probability. Insulating material was used high voltage places where human contact is required.
We implemented mechanical safeguards to prevent users from coming in contact with moving parts or sharp edges. We ensured that the robot's balance system is well-calibrated to prevent sudden falls or uncontrolled movements.

5.5.3 Assessment of Legal Issues

According to our research there was no legal issue during the completion of our project.

5.6 Sustainability and Environmental Impact Evaluation

This project doesn't any fast-decaying component. And the operations are done on low voltage. So the decaying of components will be minimal and the components will be able to operate for a very long time. And the components are also recyclable.

This project doesn't emit any harmful element for the environment. By using this DIY self-balancing robot efficiency is maximized. This in turn ensures that less energy is required for this and so less contribution to emission of carbon-di-oxide and other gases.

5.7Ethical Issues

We didn't face any moral dilemma while doing this project because there is nothing unethical about this project. We can proudly say that this project is fully ethical.

6 Reflection on Individual and Team work

6.1 Individual Contribution of Each Member

1906002 – Contributed in analysis, assembling, building structure

1906004 – Contributed in assembling, controlling, debugging, building structure

1906011 – Contributed in buying components, analysis

1906016 – Contributed in coding, assembling, debugging

1906017 – Contributed in buying components, analysis

1906022 – Contributed in assembling, debugging, building structure

6.2 Mode of TeamWork

Our team took a collaborative approach in doing things. We tried to work as a team while completing individual responsibilities. For completing a particular **test** multiple member tried to work together (at least two) so that errors could be minimized. Decision making was done through consensus.

6.3 Diversity Statement of Team

In all the decision-making from preparing proposal to writing final report every member shared their opinion and each opinion was evaluated with care. This inclusiveness, this environment where every opinion was heard created a diverse environment.

6.4 Log Book of Project Implementation

Date	Milestone achieved	Individual Role	Team Role	Comments
Week 1-3	Project proposal preparation	Each member suggested four different ideas	Selected most Suitable two for proposal	We worked as a team and completed all the tests, experiments
Week 4	Preparing detailed workflow of the selected project	Idea of the accepted project was given by the one who suggested it	Accepted the workflow and suggested some minor change where needed	
Week 5-7	Analysing the theories related to project, collecting the material for hardware implementation	Each member done their analysis from different website	Most suitable analysis and design idea was accepted	
Week 8-10	Building structure	Collect parts	Assembling	
Week 11-12	Coding part	Testing and analysing	Testing	
Week 13	Debugging	Debugging, report write	Completing project	

7 Communication

7.1 Executive Summary

[xyz city, 12/09/23] — We're thrilled to unveil our latest innovation, the DIY Self-Balancing Robot kit, designed for everyone, from tech enthusiasts to curious minds! This project offers an exciting opportunity to build your very own self-balancing robot from scratch.
With user-friendly instructions and all necessary components included, you'll embark on a captivating journey into robotics. Whether you're a beginner or a seasoned DIY enthusiast, our kit simplifies the complex, making it fun and educational.
Join us in embracing the future of robotics – create, learn, and amaze with your very own DIY Self-Balancing Robot! Get yours today and dive into the world of innovation.
For more information, visit [www.xyzcontrolbalancingrobot.com]] or contact [01763110902].
[XYZ Control Limited]

7.2 User Manual

Turn on the Controller by connecting Controller Arduino board to a pc. Then turn on the bot by connecting the battery power cable to the bot board connector. Lie down the robot for 5 seconds to let the bot calibrate its position. Then manually make the robot stand. The robot should keep itself balance from then on. Now you can also control the robot by the controller.

8 Project Management and Cost Analysis

8.1 Bill of Materials

Component	Quantity	Unit Price	Total Price
11.1 V 1500mah 35C 3S Lipo Battery	1	1750	1750
Hc-05 Bluetooth Module	2	340	680
MPU-6050	1	190	190
NEMA 17 Stepper Motor	2	990	1980
A4988 Stepper Motor Driver Module	2	125	250
Male-Male Jumper Wires	120	3	360
Male-Female Jumper Wires	40	3	120
6MM Gear motor Coupler	2	175	350
M4 20mm Hex Socket Cap Screw	12	9	108
M3 Stainless Steel Nuts	5	7	35
M4 20 Series Slot T-nut	12	11	132
Breadboard	3	145	435
Arduino Uno R3	1	660	660
Arduino Nano ATmega328	3	460	1400
Digital Clamp Multimeter	1	460	460
10*15 cm Veroboard	2	130	260
MDF Board	1	90	90
Double Layer PCB	2	60	120
Car Robot Wheel	2	399	798
Nema 17 Motor Mount Steel Breaker	2	295	590
		Total	11000

9 Future Work

For the future development of the balancing robot, the robot system modelling and simulation with different control theory is highly recommended.
Low noise reference voltage circuit board can be used to regulate the conversion by providing an accurate voltage reference to the microcontroller. It will reduce the unstable reference voltage and noise in data conversion process.

Stepper motor with encoder is recommended as encoder motor generally has higher torque and speed specification compared to any dc gear motor. The encoder can be used gain feedback for determining and control the motor speed very accurately

10 References

https://www.instructables.com/Arduino-Self-Balancing-Robot-1/

https://www.flyrobo.in/blog/self-balancing-robot