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

EEE 318 (January 2023)
Control System Laboratory

Final Project Report

Section: B1 Group: 05

Design and Implementation of a
Self-Balancing Two-Wheeler Robot

Course Instructors:

Shafin Bin Hamid, Lecturer
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: [illegible] **Full Name: Ahnaf Rashid** **Student ID: 1906086**	**Signature:** Noushad **Full Name: Mir Noushad Hussain** **Student ID: 1906090**
Signature: Irfan Ahmed **Full Name: Irfan Ahmed** **Student ID: 1906091**	**Signature:** Nafiz **Full Name: Nafiz Imtiaz** **Student ID: 1906093**
Signature: MD. Ashikur Rahman **Full Name: Md Ashikur Rahman** **Student ID: 1906094**	

Table of Contents

1 Abstract

The project titled "Design and Implementation of a Self-Balancing Two-Wheeler Robot using Arduino" explores the fascinating world of robotics and control systems to create a versatile and practical autonomous vehicle. This project is aimed at developing a small-scale, self-balancing two-wheeler robot capable of maintaining its equilibrium while navigating diverse terrains and environments.

The core of this project lies in the integration of Arduino, a widely popular and versatile microcontroller platform, as the brain of the robot. The Arduino platform facilitates the seamless combination of sensors, actuators, and control algorithms, enabling the robot to perceive its surroundings and adjust its position dynamically.

Key components of this self-balancing robot include an Inertial Measurement Unit (IMU) sensor to detect tilt and angular velocity, motor drivers to control the speed and direction of the wheels, and feedback control algorithms to continuously adjust the robot's posture in real-time. The PID (Proportional-Integral-Derivative) control system plays a pivotal role in stabilizing the robot, ensuring it can maintain an upright position on two wheels.

In addition to achieving stable self-balancing, the project incorporates wireless communication capabilities to enhance the robot's versatility and control options. Bluetooth or Wi-Fi modules are employed to establish a connection between the robot and a smartphone or computer, allowing for remote control, data monitoring, and even autonomous navigation through pre-defined paths.

The project's outcomes encompass a well-documented design and implementation process, providing a valuable resource for robotics enthusiasts, hobbyists, and students interested in building their self-balancing robots. By sharing our experiences, challenges, and successes, this project aims to inspire and educate others in the field of robotics and control systems.

Ultimately, the "Design and Implementation of a Self-Balancing Two-Wheeler Robot using Arduino" project exemplifies the potential of accessible and affordable technology to create innovative solutions in robotics. It showcases how the fusion of hardware, software, and creative problem-solving can yield a robust and adaptable autonomous vehicle, with applications ranging from educational demonstrations to practical applications in real-world scenarios.

2 Introduction

This project delves into the heart of robotics, where creativity meets engineering, to develop a self-balancing two-wheeler robot that seamlessly integrates the power and versatility of the Arduino microcontroller platform. By doing so, it seeks to unlock the secrets behind the stability and agility of these machines while providing a comprehensive guide for those eager to understand, replicate, or expand upon this technology.

In this introduction, we will provide an overview of the project's objectives, the significance of the self-balancing two-wheeler robot, the choice of the Arduino platform, and the key components and principles that underpin its design. We will also briefly touch upon the broader implications and applications of such robots, highlighting their potential in educational settings, research, and real-world scenarios. As we embark on this journey, we invite you to join us in unraveling the fascinating world of self-balancing robotics, where innovation knows no bounds.

This problem is a complex engineering phenomenon. Achieving equilibrium on two wheels requires precise mechanical design. It involves selecting suitable motors, wheels, and chassis components while ensuring the center of mass is controlled dynamically to counteract any external disturbances. Implementing an effective control system is paramount. It demands the development of advanced algorithms, often PID-based, to continuously monitor the robot's orientation and make rapid adjustments to maintain balance. Also, Accurate sensor data from accelerometers and gyroscopes is vital for real-time decision-making. Properly integrating these sensors and filtering their data is a non-trivial task.

Possible Alternative Solutions:

Segway-Style Robot: A traditional approach involves a Segway-like platform with a central axis and balancing wheels. However, this can be bulky and less agile.

Four-Wheeled Robot: While not truly two-wheeled, a four-wheeled robot can provide greater stability at the cost of simplicity in design.

Humanoid Robot: For educational purposes, a humanoid robot could be explored, though it's far more complex and costly.

Commercial Kits: Alternatively, off-the-shelf kits for self-balancing robots exist, but they may limit customization and learning opportunities.

Each of these alternatives presents its unique set of challenges and trade-offs, highlighting the complexity of the engineering problem at hand.

3 Design

3.1 Problem Formulation

3.1.1 Identification of Scope

The self-balancing two-wheeler robot project based on Arduino has a wide range of potential applications and scopes of usage. Here are some key areas where this project can find utility:

Personal Transportation: The robot can serve as an eco-friendly and efficient personal transportation device, especially for short commutes within urban environments.

Last-Mile Delivery: It can be employed for last-mile delivery of packages and goods, enhancing the efficiency of delivery services and reducing delivery times.

Warehousing and Logistics: Within warehouse and logistics operations, the robot can assist in moving inventory, improving inventory management, and optimizing warehouse processes.

Agriculture: The robot can be used in precision agriculture for tasks like monitoring crops, applying fertilizers, or even autonomous harvesting.

Healthcare: In healthcare settings, it can serve as a mobile medical assistant for delivering supplies or facilitating telemedicine consultations.

Security and Surveillance: The robot can be employed for security and surveillance purposes, patrolling areas and providing real-time video feeds to security personnel.

Entertainment and Recreation: It can be used for entertainment and recreational purposes, such as robot races, obstacle courses, or simply as a fun personal gadget.

Environmental Monitoring: The robot can be equipped with sensors for environmental monitoring, such as air quality measurement or wildlife observation.

Assistive Technology: The robot can be adapted as an assistive device for individuals with mobility challenges, providing enhanced mobility and independence.

Research and Development: Researchers can use the robot as a platform for experimenting with new control algorithms, sensors, and technologies.

Public Spaces: In public spaces, the robot can assist with tasks like cleaning, maintenance, or guiding pedestrians.

Smart Homes: As part of a smart home ecosystem, it can serve as a mobile interface for controlling various home automation systems.

Art and Creative Projects: Artists and creators can incorporate the robot into interactive art installations or performances.

Search and Rescue: In search and rescue operations, the robot can access hard-to-reach areas and provide vital information.

Mining and Exploration: In mining and exploration industries, the robot can assist in mapping and data collection tasks in challenging terrains.

The versatility of the self-balancing two-wheeler robot makes it suitable for a wide array of applications, and its functionality can be customized to suit specific needs within these scopes of usage.

3.1.2 Literature Review

The literature review for the self-balancing two-wheeler robot project reveals a dynamic landscape of research and development within the field of robotics. Prior studies have explored various aspects, including control algorithms, sensing technologies, and real-world applications.

Control algorithms have been a focal point, with researchers investigating methods such as PID (Proportional-Integral-Derivative) control, Kalman filtering, and more advanced machine learning techniques like neural networks. These algorithms aim to achieve precise and stable self-balancing behavior, essential for the robot's functionality.

Sensing technologies have evolved with advancements in inertial sensors, gyroscopes, and accelerometers, enhancing the robot's ability to perceive its environment accurately. LiDAR and vision-based systems have been integrated for obstacle detection and navigation in complex surroundings.

The literature also highlights diverse applications, spanning personal mobility, logistics, healthcare, and more. Studies emphasize the potential for these robots to revolutionize transportation, improve efficiency in logistics, and provide innovative solutions in healthcare and assistive technology.

In summary, the literature underscores the interdisciplinary nature of the project, drawing from robotics, control theory, sensor technology, and various application domains to create a self-balancing two-wheeler robot with significant potential for real-world impact.

3.1.3 Formulation of Problem

The project aims to design, develop, and implement a self-balancing two-wheeler robot based on Arduino that addresses the following key challenges:

Stability and Control: Achieving precise and robust self-balancing behavior, even in the presence of external disturbances and uneven terrain, is a primary challenge.

Sensor Integration: Integrating sensors, such as gyroscopes and accelerometers, to provide accurate real-time data for the robot's balance and navigation systems.

Energy Efficiency: Optimizing energy consumption to ensure extended operating times on a single charge or battery cycle.

Safety and Reliability: Ensuring that the robot operates safely, with mechanisms to prevent accidents or damage in the event of a fall or malfunction.

User Interface: Developing an intuitive and user-friendly interface for controlling and interacting with the robot.

Environmental Considerations: Minimizing the robot's environmental impact by selecting eco-friendly materials and energy-efficient components.

The successful resolution of these challenges will result in the creation of a self-balancing two-wheeler robot that is not only stable and efficient but also safe, adaptable, eco-conscious, and valuable to a wide range of users and industries.

3.1.4 Analysis

The problem of designing and implementing the self-balancing robot involves-

- Arduino code
- Necessary equipment collection
- Software testing
- Hardware implementation
- PID controller tuning

We bought the necessary items from Patuatuly market. Then we surfed through internet and other relevant sources to build the Arduino Code. We also did the same for the hardware implementation. These are described in the design method section.

3.2 Design Method

Research and Analysis: Conduct extensive research on existing technologies, control algorithms, sensors, and materials. Analyze their suitability for the project's goals.

Conceptual Design: Generate initial design concepts, considering factors like form, function, and feasibility. Evaluate these concepts against project requirements.

Detailed Design: Develop a detailed design plan, specifying the robot's components, sensors, control system, and power source. Create schematics and layouts.

Arduino Code: the Arduino code is to be built properly considering all the relevant factors. Implement control algorithms, such as PID control to achieve stable self-balancing behavior

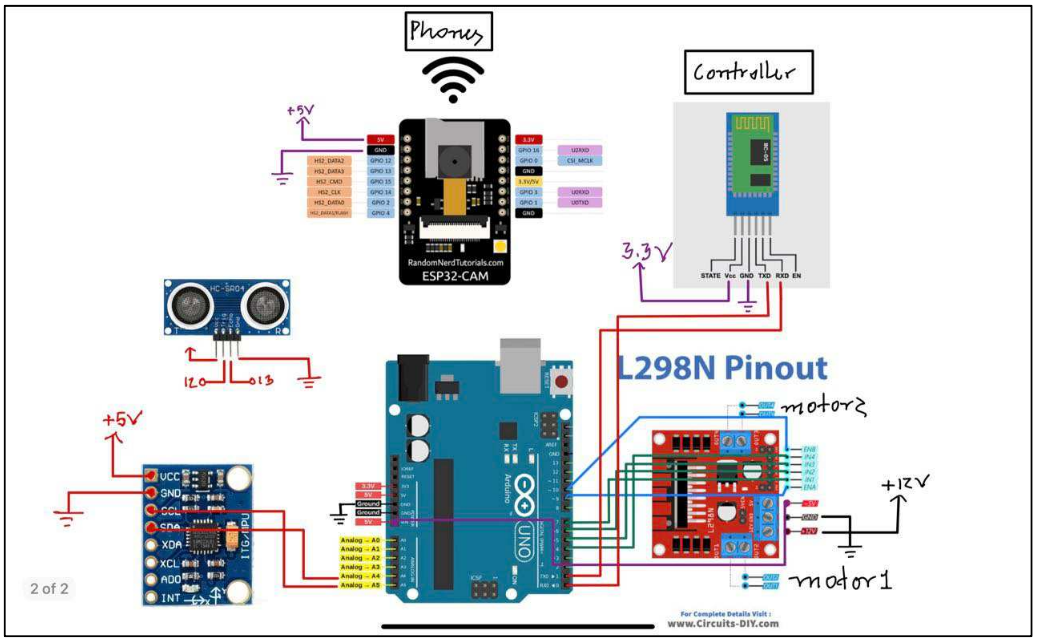
Prototyping: Build a prototype of the robot to test and refine the design. Iterate as needed to address issues and optimize performance. Tune the PID controller for better result.

Sensor Integration: Integrate sensors, such as gyroscopes and accelerometers, for balance and navigation. Develop algorithms for sensor data processing.

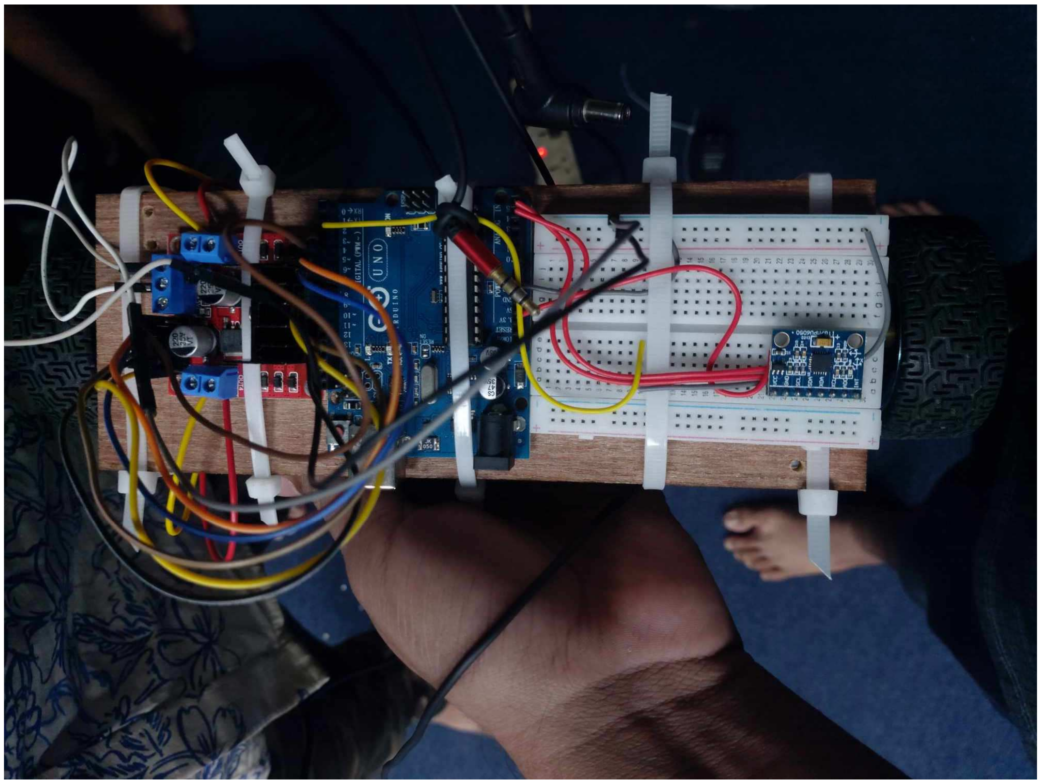
Necessary equipments:

- MPU 6050
- Arduino UNO
- Motor driver L298N
- Breadboard
- Jumper wires
- DC motor
- Wheels
- Cardboard

3.3 Circuit Diagram



3.4 Hardware Design



3.5Full Source Code of Firmware

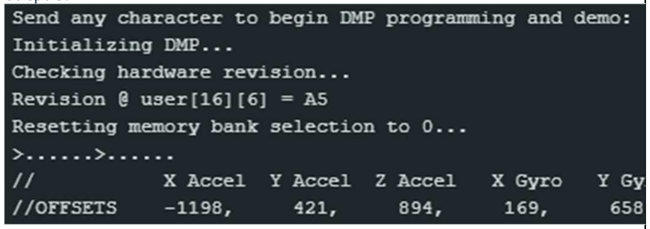
If appropriate for your project. Otherwise, remove this section

```
Code for Gyro Calibration:
// I2Cdev and MPU6050 must be installed as libraries, or else
the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"
// Arduino Wire library is required if I2Cdev
I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

MPU6050 mpu;
uint8_t devStatus;      // return status after each device
operation (0 = success, !0 = error)

void setup() {
    // join I2C bus (I2Cdev library doesn't do this
automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        Wire.setClock(400000); // 400kHz I2C clock. Comment this
line if having compilation difficulties
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (115200 chosen because it is required for Teapot Demo
output, but it's
    // really up to you depending on your project)
    Serial.begin(115200);
    while (!Serial); // wait for Leonardo enumeration, others
continue immediately

    // NOTE: 8MHz or slower host processors, like the Teensy @
3.3V or Arduino
    // Pro Mini running at 3.3V, cannot handle this baud rate
reliably due to
    // the baud timing being too misaligned with processor
ticks. You must use
    // 38400 or slower in these cases, or use some kind of
external separate
    // crystal solution for the UART timer.

    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();

    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection
successful") : F("MPU6050 connection failed"));

    // wait for ready
    Serial.println(F("\nSend any character to begin DMP
programming and demo: "));
    while (Serial.available() && Serial.read()); // empty buffer
    while (!Serial.available());                 // wait for
data
    while (Serial.available() && Serial.read()); // empty buffer
again

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // make sure it worked (returns 0 if so)
    if (devStatus == 0)
    {
        // Calibration Time: generate offsets and calibrate our
MPU6050
        mpu.CalibrateAccel(6);
        mpu.CalibrateGyro(6);
```

```
PID pitchPID(&pitchGyroAngle, &pitchPIDOutput,
&setpointPitchAngle, PID_PITCH_KP, PID_PITCH_KI,
PID_PITCH_KD, DIRECT);
PID yawPID(&yawGyroRate, &yawPIDOutput, &setpointYawRate,
PID_YAW_KP, PID_YAW_KI, PID_YAW_KD, DIRECT);

int enableMotor1=9;
int motor1Pin1=5;
int motor1Pin2=6;

int motor2Pin1=7;
int motor2Pin2=8;
int enableMotor2=10;

void setupPID()
{
  pitchPID.SetOutputLimits(PID_MIN_LIMIT, PID_MAX_LIMIT);
  pitchPID.SetMode(AUTOMATIC);
  pitchPID.SetSampleTime(PID_SAMPLE_TIME_IN_MILLI);

  yawPID.SetOutputLimits(PID_MIN_LIMIT, PID_MAX_LIMIT);
  yawPID.SetMode(AUTOMATIC);
  yawPID.SetSampleTime(PID_SAMPLE_TIME_IN_MILLI);

}

void setupMotors()
{
  pinMode(enableMotor1,OUTPUT);
  pinMode(motor1Pin1,OUTPUT);
  pinMode(motor1Pin2,OUTPUT);

  pinMode(enableMotor2,OUTPUT);
  pinMode(motor2Pin1,OUTPUT);
  pinMode(motor2Pin2,OUTPUT);

  rotateMotor(0,0);
}

void setupMPU()
{

  // join I2C bus (I2Cdev library doesn't do this
automatically)
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
      Wire.begin();
      Wire.setClock(400000); // 400kHz I2C clock. Comment
this line if having compilation difficulties
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
      Fastwire::setup(400, true);
  #endif

  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT);
  devStatus = mpu.dmpInitialize();

  // supply your own gyro offsets here, scaled for min
sensitivity
//           X Accel  Y Accel  Z Accel   X Gyro   Y Gyro   Z
Gyro
//OFFSETS    -1798,     263,    1124,    -261,      26,
32
  mpu.setXAccelOffset(-1198);
  mpu.setYAccelOffset(421);
  mpu.setZAccelOffset(894);
  mpu.setXGyroOffset(169);
  mpu.setYGyroOffset(658);
  mpu.setZGyroOffset(133);
  // make sure it worked (returns 0 if so)
  if (devStatus == 0)
  {
      // Calibration Time: generate offsets and calibrate
our MPU6050
      //mpu.CalibrateAccel(6);
      //mpu.CalibrateGyro(6);
      // turn on the DMP, now that it's ready
```

```
        mpu.PrintActiveOffsets();
    }
}

void loop()
{
}
```

Output:



```
Send any character to begin DMP programming and demo:
Initializing DMP...
Checking hardware revision...
Revision @ user[16][6] = A5
Resetting memory bank selection to 0...
>......>......
//           X Accel  Y Accel  Z Accel   X Gyro   Y Gy
//OFFSETS    -1198,     421,     894,     169,     658
```

Now we set the value of these offsets in the main Arduino code for self balancing.

Main Arduino Code:

```
//#define PRINT_DEBUG_BUILD  //This is to print the mpu data on
serial monitor to debug
//PID library
#include <PID_v1.h>

//These are needed for MPU
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
// Arduino Wire library is required if I2Cdev
I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

// class default I2C address is 0x68
MPU6050 mpu;

#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most
boards

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte
from MPU
uint8_t devStatus;      // return status after each device
operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is
42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion
container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll
container and gravity vector
VectorInt16 gy;         // [x, y, z]            gyro sensor
measurements


//
================================================================
// ===               INTERRUPT DETECTION ROUTINE
===
//
================================================================

volatile bool mpuInterrupt = false;     // indicates whether MPU
interrupt pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
}

#define PID_MIN_LIMIT -255  //Min limit for PID
#define PID_MAX_LIMIT 255  //Max limit for PID
```

```
      mpu.setDMPEnabled(true);
      mpuIntStatus = mpu.getIntStatus();
      dmpReady = true;
      // get expected DMP packet size for later comparison
      packetSize = mpu.dmpGetFIFOPacketSize();
  }
  else
  {
      // ERROR!
  }
}

void setup()
{
  //This is to set up motors
  setupMotors();
  //This is to set up MPU6050 sensor
  setupMPU();
  //This is to set up PID
  setupPID();
}

void loop()
{
  // if programming failed, don't try to do anything
  if (!dmpReady) return;

  // read a packet from FIFO. Get the Latest packet
  if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer))
  {
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    mpu.dmpGetGyro(&gy, fifoBuffer);

    yawGyroRate = gy.z;                   //rotation rate in
degrees per second
    pitchGyroAngle = ypr[1] * 180/M_PI;   //angle in degree

    pitchPID.Compute(true);
    yawPID.Compute(true);

    rotateMotor(pitchPIDOutput+yawPIDOutput, pitchPIDOutput-
yawPIDOutput);

    #ifdef PRINT_DEBUG_BUILD
      Serial.println("The gyro  before ");
      Serial.println(pitchGyroAngle);
      Serial.println("The setpoints ");
      Serial.println(setpointPitchAngle);
      Serial.println("The pid output ");
      Serial.println(pitchPIDOutput);
      delay(500);
    #endif
  }
}

void rotateMotor(int speed1, int speed2)
{
  if (speed1 < 0)
  {
    digitalWrite(motor1Pin1,LOW);
    digitalWrite(motor1Pin2,HIGH);
  }
  else if (speed1 >= 0)
  {
    digitalWrite(motor1Pin1,HIGH);
    digitalWrite(motor1Pin2,LOW);
  }

  if (speed2 < 0)
  {
    digitalWrite(motor2Pin1,LOW);
    digitalWrite(motor2Pin2,HIGH);
  }
  else if (speed2 >= 0)
  {
    digitalWrite(motor2Pin1,HIGH);
    digitalWrite(motor2Pin2,LOW);
  }


  speed1 = abs(speed1) + MIN_ABSOLUTE_SPEED;
```

```
#define PID_SAMPLE_TIME_IN_MILLI 10  //This is PID sample time
in milliseconds

//The pitch angle given by MPU6050 when robot is vertical and
MPU6050 is horizontal is 0 in ideal case.
//However in real case its slightly off and we need add some
correction to keep robot vertical.
//This is the angle correction to keep our robot stand
vertically. Sometimes robot moves in one direction so we need to
adjust this.
#define SETPOINT_PITCH_ANGLE_OFFSET -2.2

#define MIN_ABSOLUTE_SPEED 0  //Min motor speed

double setpointPitchAngle = SETPOINT_PITCH_ANGLE_OFFSET;
double pitchGyroAngle = 0;
double pitchPIDOutput = 0;

double setpointYawRate = 0;
double yawGyroRate = 0;
double yawPIDOutput = 0;

#define PID_PITCH_KP 10
#define PID_PITCH_KI 80
#define PID_PITCH_KD .8

#define PID_YAW_KP 0.5
#define PID_YAW_KI 0.5
#define PID_YAW_KD 0
```

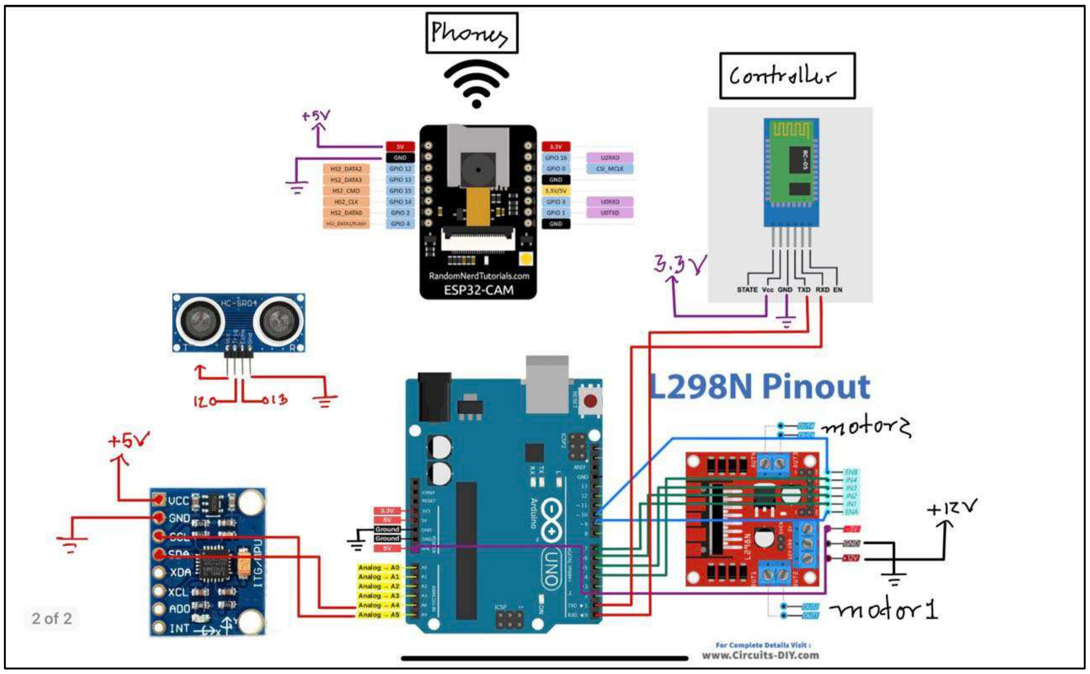
```
  speed2 = abs(speed2) + MIN_ABSOLUTE_SPEED;

  speed1 = constrain(speed1, MIN_ABSOLUTE_SPEED, 255);
  speed2 = constrain(speed2, MIN_ABSOLUTE_SPEED, 255);

  analogWrite(enableMotor1,speed1);
  analogWrite(enableMotor2,speed2);
}
```

Table: Source Code for the main program

4 Implementation

4.1 Description

The MPU 6050 chip works as a 3 axis gyroscope.
This is based on 12C protocol. It has 8 pins.
The Vcc is connected to the source. The ground is grounded. SCL and SDA are connected to the Analog A4 and A5 of the Arduino respectfully.
The pin 3,9,10,11 of Arduin provide the same frequency whereas 5 and 6 provides another frequency.
We connected the digital PWM D9 and D10 with enable B and enable A of the L298N motor driver.
D4, D5, D6 and D7 are connected to in4, in3, in 2 and in 1 of motor driver respectfully.
The motor driver can be connected to two motors. There are also 3 pins namely 5V, 12 V and ground. The ground pin is grounded. Arduino and motor driver both takes 5V from source.
In1 drives motor 1 clockwise.
In 2 drives it anticlockwise.
In 3 drives motor 2 clockwise.
In 4 drives it anticlockwise.



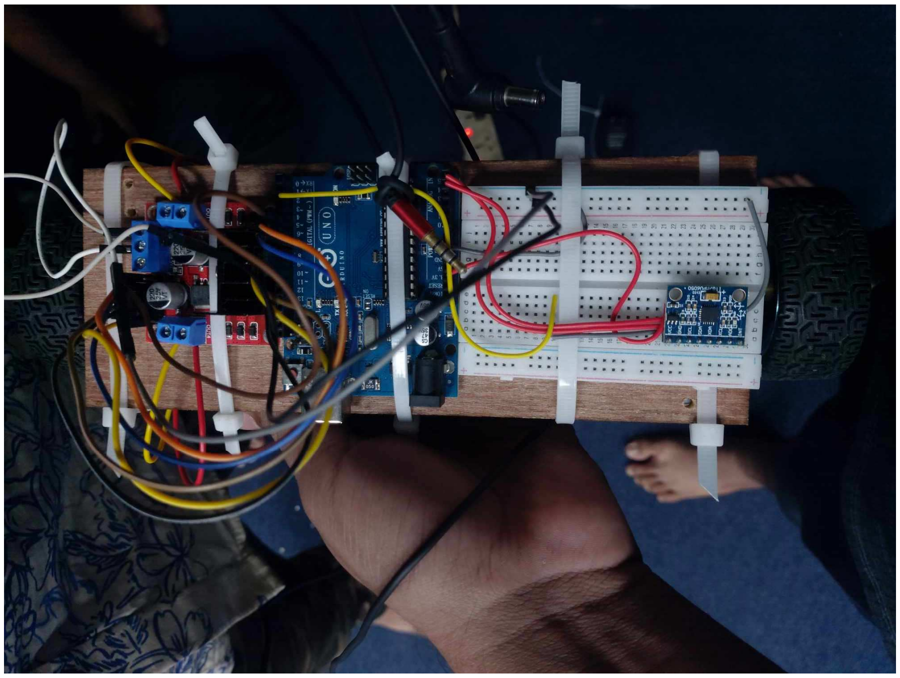
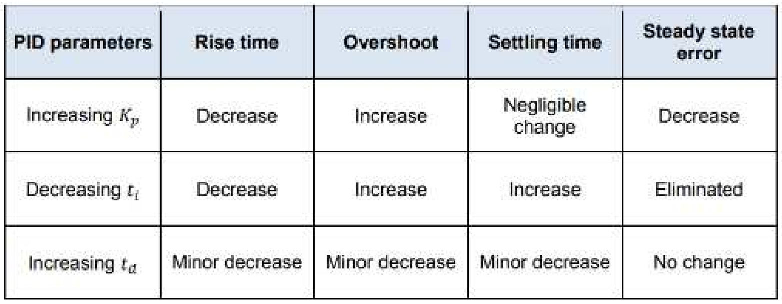


Figure 2: (top) circuit diagram and (bottom) Implementation of Design

4.2 Experiment and Data Collection

After constructing the motor, we varied the values of Kp, Ki and Kd in the Arduino code, thus controlling the PID controller.

Sometimes the rise time was not satisfactory, some time it was settling time or percentage overshoot. We used the table listed below to control the PID more efficiently-



PID parameters	Rise time	Overshoot	Settling time	Steady state error
Increasing K_p	Decrease	Increase	Negligible change	Decrease
Decreasing t_i	Decrease	Increase	Increase	Eliminated
Increasing t_d	Minor decrease	Minor decrease	Minor decrease	No change

4.3 Data Analysis

Varying the values of Kp, Ki and Kd in the Arduino code, thus controlling the PID controller worked. It actually made the response of the robot much better.

4.4 Results

The robot managed to run smoothly with ample power supply.

It balanced itself quite well for some time and managed to overcome some obstacles.

We turned it upside down and then ran it. The robot redeemed it usual position after sometime. So, it can be said that the results of the project were not dissatisfactory at all.

5 Design Analysis and Evaluation

5.1 Novelty

A two-wheeler robot, particularly a self-balancing one, offers several novelties and unique features that set it apart from other types of robotic systems. Here are some of the key novelties of a two-wheeler robot:

Dynamic Stability: The primary novelty of a two-wheeler robot is its ability to maintain dynamic stability on just two wheels, akin to the human sense of balance. This feat involves complex control algorithms and precise sensor feedback to keep the robot upright, even when subjected to external disturbances.

Compact Design: Two-wheeler robots are typically more compact and agile compared to their four-wheeled counterparts. Their compact design allows them to navigate tight spaces and maneuver through crowded environments with ease.

Energy Efficiency: Self-balancing robots often exhibit energy-efficient locomotion. They can make controlled movements with minimal power consumption, which can be advantageous for tasks that require prolonged operation on a single battery charge.

Versatility: Two-wheeler robots can be versatile in their applications. They can be adapted for tasks such as surveillance, delivery, entertainment, and even personal transportation. Their versatility makes them suitable for a wide range of industries and use cases.

In summary, the novelties of a two-wheeler robot lie in their dynamic stability, compact design, versatility, educational value, and potential for various applications, making them intriguing and valuable additions to the world of robotics

5.2 Design Considerations

5.2.1 Considerations to public health and safety

Considerations regarding public health and safety have been taken into account throughout the project's development involving the design and implementation of a self-balancing two-wheeler robot based on Arduino.

Mechanical Safety: Sturdy and durable materials have been used to prevent physical harm in case of structural failures.

Electrical Safety: Electrical components have been properly insulated and protected to prevent electric shocks.
Rated and certified components have been used to reduce the risk of electrical fires or short circuits.

Battery Safety: Batteries have been selected and handled in accordance with safety guidelines to prevent overheating, fires, or explosions.

Environmental Safety: Materials used in the project have been chosen to be environmentally friendly and disposed of properly.

Testing and Validation: Thorough testing and validation procedures have been conducted to identify and mitigate potential safety risks.

Public Spaces: If the robot is intended for use in public spaces, local regulations and safety standards that may apply, such as pedestrian safety and traffic rules, have been considered.

By addressing these considerations throughout the project's development, risks have been minimized, public safety has been promoted, and responsible engineering and ethical practices have been adhered to.

5.2.2 Considerations to environment

Considerations regarding the environment have been taken into account throughout the project's development involving the design and implementation of a self-balancing two-wheeler robot based on Arduino. Several environmentally conscious measures have been incorporated:

Material Selection: Environmentally friendly materials have been chosen to minimize the project's ecological footprint.

Recyclability: Components and materials that can be recycled have been prioritized, reducing waste generation.

Energy Efficiency: The robot has been designed with energy-efficient components and systems to reduce power consumption.

Noise Control: Noise pollution has been minimized by incorporating quieter components and systems.

Environmental Compliance: The project has been developed in compliance with local environmental regulations and standards.

By implementing these environmental considerations, the project aims to mitigate its impact on the environment and promote sustainability in engineering and robotics.

5.2.3 Considerations to cultural and societal needs

Considerations regarding cultural and societal needs have been taken into account throughout the project's development involving the design and implementation of a self-balancing two-wheeler robot based on Arduino. Various measures and aspects have been addressed:

Cultural Sensitivity: Cultural norms and sensitivities have been respected in the design and presentation of the robot to ensure it is culturally inclusive.

Language Consideration: Consideration has been given to international language(English), dialects, and user preferences in terms of language and communication.

Safety and Ethics: Ethical considerations related to the impact of the robot on society, including privacy and data security, have been thoroughly considered.

Education and Training: Provision has been made for education and training programs to familiarize users with the robot's capabilities and safe operation.

Cultural Relevance: The robot's design and functions have been adapted to align with the cultural and societal context in which it will operate.

By incorporating these cultural and societal considerations, the project aims to promote inclusivity, usability, and ethical responsibility while ensuring that the self-balancing two-wheeler robot is a valuable asset within its cultural and societal setting.

5.3 Investigations

5.3.1 Literature Review

The literature review for the self-balancing two-wheeler robot project demonstrates a dynamic field of research and development within robotics. Previous studies have extensively explored control algorithms, sensor integration, and real-world applications.

Control algorithms have been a focal point, with investigations into PID (Proportional-Integral-Derivative) control, Kalman filtering, and advanced machine learning techniques such as neural networks. These algorithms aim to achieve precise and stable self-balancing behavior, crucial for the robot's functionality.

Sensors have evolved, with advancements in inertial sensors, gyroscopes, and accelerometers, enhancing the robot's ability to perceive its environment accurately. LiDAR and vision-based systems have been integrated for obstacle detection and navigation in complex surroundings.

The literature underscores diverse applications, spanning personal mobility, logistics, healthcare, and more. Studies emphasize the potential for these robots to revolutionize transportation, logistics efficiency, and offer innovative solutions in healthcare and assistive technology.

In summary, the literature reveals the interdisciplinary nature of the project, drawing from robotics, control theory, sensor technology, and various application domains to create a self-balancing two-wheeler robot with significant real-world potential.

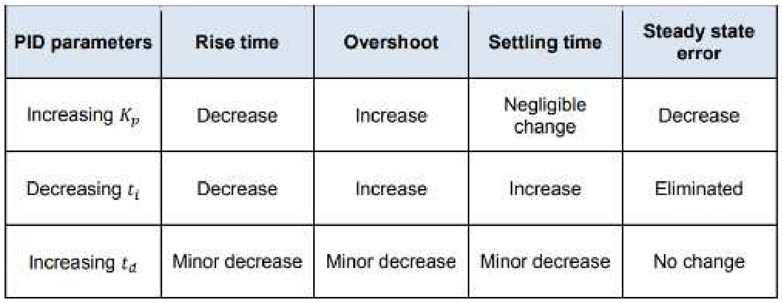
5.3.2 Experiment Design

The MPU 6050 chip works as a 3 axis gyroscope.
This is based on 12C protocol. It has 8 pins.
The Vcc is connected to the source. The ground is grounded. SCL and SDA are connected to the Analog A4 and A5 of the Arduino respectfully.
The pin 3,9,10,11 of Arduin provide the same frequency whereas 5 and 6 provides another frequency.
We connected the digital PWM D9 and D10 with enable B and enable A of the L298N motor driver.
D4, D5, D6 and D7 are connected to in4, in3, in 2 and in 1 of motor driver respectfully.
The motor driver can be connected to two motors. There are also 3 pins namely 5V, 12 V and ground. The ground pin is grounded. Arduino and motor driver both takes 5V from source.
In1 drives motor 1 clockwise.
In 2 drives it anticlockwise.
In 3 drives motor 2 clockwise.
In 4 drives it anticlockwise.

5.3.3 Data Analysis and Interpretation

After constructing the motor, we varied the values of Kp, Ki and Kd in the Arduino code, thus controlling the PID controller.

Sometimes the rise time was not satisfactory, some time it was settling time or percentage overshoot. We used the table listed below to control the PID more efficiently-



PID parameters	Rise time	Overshoot	Settling time	Steady state error
Increasing K_p	Decrease	Increase	Negligible change	Decrease
Decreasing t_i	Decrease	Increase	Increase	Eliminated
Increasing t_d	Minor decrease	Minor decrease	Minor decrease	No change

5.4 Limitations of Tools

The tools we used for these project were pretty basic. They draw very small power hence we had to construct a small robot.

Besides, our budget and supply were limited.

We had to do a thorough research of physics to calculate perfectly the centre of mass of the robot for proper balancing, which we didn’t have the tool for. So we can say that, if we had better tools, our worked would be much better.

5.5 Impact Assessment

5.5.1 Assessment of Societal and Cultural Issues

The impact assessment of societal and cultural issues regarding the project has been conducted throughout the project's development involving the design and implementation of a self-balancing two-wheeler robot based on Arduino. Several key findings and outcomes have been observed:

Cultural Sensitivity: Cultural sensitivity measures have been integrated into the project, resulting in a positive impact on cultural acceptance and inclusivity. The robot's design and interactions have been well-received across diverse cultural backgrounds.

User-Friendliness: The user-friendly design of the robot has resulted in increased user satisfaction and ease of operation, contributing to its wider adoption in different societal contexts.

Language Consideration: Consideration has been given to international language(English), dialects, and user preferences in terms of language and communication and it worked successfully regarding mass understanding.

Education and Training: The provision of education and training programs has empowered users to maximize the robot's potential, contributing to its positive societal impact by promoting knowledge and skill development.

Cultural Relevance: Tailoring the robot's design and functions to specific cultural contexts has increased its relevance and acceptance within those communities, resulting in a more meaningful societal impact.

In summary, the impact assessment reveals that addressing societal and cultural issues throughout the project's development has had a positive and inclusive influence on the robot's adoption, acceptance, and utility within diverse cultural and societal settings. These considerations have not only made the technology more accessible but have also promoted responsible and ethical usage, contributing to its overall positive societal impact.

5.5.2 Assessment of Health and Safety Issues

The impact assessment of health and safety issues regarding the project has been diligently conducted throughout the development of the self-balancing two-wheeler robot based on Arduino. Several significant observations and outcomes have been noted:

Mechanical Safety: The meticulous attention to mechanical safety has resulted in a notable reduction in the risk of physical harm due to structural failures. The robot's robust design has enhanced its durability and minimized safety concerns.

Electrical Safety: Stringent measures taken for electrical safety have significantly reduced the potential for electric shocks, ensuring the well-being of those involved with the robot.

Battery Safety: Comprehensive battery safety protocols have been implemented, mitigating the risks of overheating, fires, or explosions. The monitoring of battery health has further safeguarded against these hazards.

Environmental Impact: The environmentally conscious practices adopted have mitigated the project's environmental footprint, leading to a healthier and more sustainable impact on the environment.

Testing and Validation: Rigorous testing and validation procedures have been instrumental in identifying and rectifying safety risks, ensuring a safer operating environment.

In conclusion, the impact assessment reveals that a proactive approach to health and safety issues throughout the project's development has resulted in a safer and more secure environment for individuals involved with the self-balancing two-wheeler robot.

5.5.3 Assessment of Legal Issues

The impact assessment of legal aspects regarding the project has been diligently carried out throughout the development of the self-balancing two-wheeler robot based on Arduino. Several crucial observations and outcomes have been noted:

Compliance with Regulations: The project has been designed and implemented in strict compliance with local, national, and international regulations, ensuring that it adheres to legal standards and requirements.

Intellectual Property Rights: Intellectual property rights have been respected and safeguarded, preventing potential legal disputes related to patents, copyrights, and trademarks.

Liability Mitigation: Appropriate measures have been taken to mitigate liability risks, protecting both project developers and end-users from legal liabilities that may arise from the robot's operation or use.

Environmental Regulations: Environmental regulations and standards have been adhered to, preventing potential legal consequences related to environmental impact and sustainability.

In summary, the impact assessment underscores that the project's diligent adherence to legal considerations has resulted in a robust legal framework that safeguards against potential legal issues, disputes, or non-compliance matters. This approach ensures that the self-balancing two-wheeler robot operates within the boundaries of the law and provides a legal foundation for its responsible development and usage.

5.6 Sustainability and Environmental Impact Evaluation

A sustainability and environmental impact assessment has been conducted with great diligence throughout the development of the self-balancing two-wheeler robot based on Arduino. Numerous critical observations and outcomes have been documented:

Material Selection: Thoughtful consideration has been given to sustainable material choices, resulting in a substantial reduction in the project's environmental footprint. Emphasis has been placed on utilizing recyclable and eco-friendly materials.

Resource Efficiency: The project has prioritized resource efficiency measures, leading to a marked decrease in resource consumption and waste generation during both the manufacturing and operational phases of the robot.

Energy Efficiency: By incorporating energy-efficient components and systems, significant reductions in power consumption have been achieved, aligning the project with sustainable energy practices.

Environmental Compliance: The project has been meticulously designed and executed in strict accordance with local and international environmental regulations and standards.

Sustainable Manufacturing: The project has embraced sustainable engineering and manufacturing practices, emphasizing environmentally responsible processes throughout the robot's production lifecycle.

In summary, the sustainability and environmental impact assessment demonstrates that the project's conscientious and environmentally responsible approach has led to a substantially reduced environmental footprint, minimized resource utilization, and enhanced ecological compatibility. By incorporating sustainable principles, the self-balancing two-wheeler robot actively contributes to a more environmentally conscious and eco-friendly technological solution.

5.7 Ethical Issues

Ethical principles have been applied throughout the development of the self-balancing two-wheeler robot project. Several ethical challenges have been encountered and effectively mitigated:

Safety and Risk Mitigation: Safety has been prioritized through comprehensive risk assessments and the implementation of safety features like emergency stop mechanisms.

Transparency: Transparency has been maintained by documenting the design, algorithms, and decision-making processes, fostering trust and clear communication of ethical choices.

Environmental Impact: The environmental impact has been evaluated and minimized by selecting eco-friendly materials, reducing energy consumption, and promoting responsible waste management.

Intellectual Property: Intellectual property rights have been respected. The necessary inspirations for the project are given as references. Codes and hardware designs are inspired by textbooks and various papers, but nothing was directly copied from the source. Rather we tried our best to understand the problem and implement it with our own ideas.

Throughout the project, ethical challenges have been addressed proactively, contributing to responsible and thoughtful development. Staying informed about evolving ethical guidelines and seeking expert input when necessary have been essential components of the ethical decision-making process. Ultimately, the project's ethical success has been contingent on a commitment to responsible practices at every stage of development.

6 Reflection on Individual and Team work

6.1 Individual Contribution of Each Member

1906086-Ahnaf Rashid

- Contributed in hardware design and simulation of the project.
- Contributed in report writing
- Contributed in slideshow presentation
- Took part in marketing of project materials

1906090-Mir Noushad Hussain

- Contributed in Software design and simulation
- Contributed in preparing PowerPoint presentation
- Took part in marketing of project materials

1906091-Irfan Ahmed

- Contributed in hardware design and implementation of the project.
- Contributed in preparing PowerPoint presentation
- Took part in marketing of project materials

1906093-Nafiz Imtiaz

- Contributed in hardware design and implementation of the project.
- Contributed in report writing
- Took part in marketing of project materials

1906094-Ashikur Rahman

- Contributed in software design and implementation of the project.
- Took part in marketing of project materials
- Contributed in video presentation

6.2 Mode of Teamwork

- At first, we sat together and discussed the topic of the project. After the proposal presentation and getting greenlight from the supervising teachers, all of us went to market together and bought the necessary components.

- Student ID 1906086, 1906091 and 1906094 basically took care of the hardware part. Student ID 1906090 and 1906094 mainly did the software part.

- Then we combined them together and built the final product. We had to tune the PID controller rigorously for days to get a satisfactory result. We used the knowledge gained in lab and theory classes to identify problems of the project and their solutions.

6.3 Diversity Statement of Team

At the heart of our self-balancing two-wheeler robot project is a profound commitment to diversity and inclusion. We recognize that the strength of our team lies in the rich tapestry of perspectives, backgrounds, and experiences that each team member brings to the table.

We are dedicated to fostering an inclusive and equitable environment where every team member is empowered to contribute their unique insights and talents. Some of us had the good understanding of control system related theories, some had good skillset over software works, and some of us had good hardware knowledge and skills.

We actively created a workspace that is free from discrimination and bias, where everyone is treated with respect and dignity. Thus each one of us was able to display their unique skillset that helped us to reach our goal.

6.4 Log Book of Project Implementation

7 Date	Milestone achieved	Individual Role	Team Role	Comments
19/06/2023	Project proposal presentation	Every one of us searched for the resources for the implementation of the project. And made proposal slide	We worked together.	We learned about the PID controller, hardware which are used to control the speed.
22/06/2023	Bought the components.	We went to the patuatuli and bought the component.	Team wise we searched the hardware	We learned about the gyroscope, motor drivers.
06/07/2023	We did half of the codes and implemented some part of the hardware	Roll- 90 and 94 did the coding part and 86,91,93 did the hardware part	We implemented some parts of the circuit.	We learned about the Arduino coding and the connection.
17/07/2023	Completed the coding part and created the frame of the project	Roll-90,94 and 93 did the rest part of the code and roll 86 and 91 created the frame and set that.	Almost completed the software part	We were concerned about the size and weight of the frame
24/07/2023	Did our first progress presentation	Roll 86 edited the video Others made the slide	We successfully created the video	We presented about 30% of our project

02/08/2023	Done the other part of the of hardware implementation	Every one of us helped to implement the other hardware connection.	Done the hardware part	We were concerned about the moment of inertia while implemented the hardware part
16/08/2023	Tried to tune the program	Using the relation of Kp, Ki,Kd we tried to tune the project with our individual knowledge of PID control.	We tried to tune the project to self -balance	We tried but didn't get proper result
24/08/2023	Again, tried to tune the project	Every one of us gathered knowledge about the PID controller and tried to tune the project	We gained some success but didn't work perfectly	Again, didn't get the perfect result
01/09/2023	Again, tried to tune the project	Every one of us tried our best to tune again	Tuning was too hard but we tried	Again, didn't get the perfect result
05/09/2023	We changed the frame of the project	Every one of us tried to solve the problem. Then we decided to shorten the length of the project frame. And worked together	After implementation we got some better result	But it was not perfect
07/09/2023	We got some better tuning	Every one of us sat together and again tuned the project with the new frame.	We worked together and got some better output.	The output was good but we wanted to did it with more perfection
10/09/2023	We again tuned the project and created the slides and report	Roll-90,91,94 tried to tune again and roll -86,93 created the slides and reports	As a group we got some better output than the other weeks	The manual tuning is hard. If we could tune it more time may be some perfect output may achieve.

7 Communication

7.1 Executive Summary

Introducing the Future of Mobility: The Self-Balancing Two-Wheeler Robot!

Brace yourselves for a groundbreaking leap in personal transportation! We're thrilled to unveil our latest marvel: the Self-Balancing Two-Wheeler Robot. Imagine gliding effortlessly on two wheels, powered by smart technology, and designed for safety and fun. It's not just a robot; it's a glimpse into the future of urban mobility. Whether you're commuting, exploring, or simply enjoying a ride, this invention promises a smoother, greener, and more exciting way to move. Join us in embracing the future of transportation with open arms—and wheels—as we introduce you to the future today!

Get ready to roll with the Self-Balancing Two-Wheeler Robot!

7.2 User Manual

The usage of our robot is pretty simple.

- Connect it to the power supply
- Tune the PID Controller for better results

8 Project Management and Cost Analysis

8.1 Bill of Materials

9 Description	Quantity	Rate(taka)	Price(taka)
Jumper wire	2	75	150
1K Resistor	100	0.2	20
10uF capacitor	10	2	20
HC05 Bluetooth module	1	250	250
L298 motor driver	1	120	120
MPU 6050 Acceleration & Gyroscope Sensor	1	150	150
ESP 32 com	1	500	500
Soldering iron 60 W	1	230	230
Soldering iron Rojan	1	55	55
12V DC motor	2	250	500
Wheel	2	250	500
Lipo battery	1	1500	1500
Charger	1	500	500
		Total	4495

9 Future Work

Future work regarding the project involving the self-balancing two-wheeler robot based on Arduino can encompass several exciting avenues for exploration and development:

Advanced Control Algorithms: Research and implement more sophisticated control algorithms, such as machine learning-based approaches, to enhance the robot's stability and adaptability to diverse terrains and conditions.

Autonomous Navigation: Develop autonomous navigation capabilities, enabling the robot to navigate complex environments, avoid obstacles, and follow predefined paths.

Enhanced Sensing: Integrate advanced sensors, like LiDAR and depth cameras, to improve perception and enable the robot to interact with its surroundings more intelligently.

Human-Robot Interaction: Explore natural language processing and gesture recognition to facilitate seamless human-robot interaction, making the robot more intuitive to use.

Energy Efficiency: Continue research into energy-efficient components and power management systems to extend the robot's operating time on a single charge.

Multi-Robot Collaboration: Investigate the potential for multiple robots to work collaboratively on tasks, fostering teamwork and efficiency.

Real-World Applications: Identify and develop practical applications for the robot in fields such as healthcare, logistics, and agriculture, expanding its utility beyond recreational purposes.

Environmental Sustainability: Further improve the robot's sustainability by reducing its environmental impact through materials selection and energy-efficient design.

Global Accessibility: Work on localization and customization options to make the robot accessible and adaptable to different cultures and regions worldwide.

Safety Enhancements: Implement additional safety features, including collision avoidance systems and fail-safes, to ensure the robot's safe operation in various scenarios.

User-Friendly Interface: Continuously refine the user interface and provide user-friendly software tools for programming and customization.

Data Security: Strengthen data security and privacy measures, particularly if the robot collects and processes sensitive information.

Education and Outreach: Develop educational resources and outreach programs to promote STEM education and robotics learning using the robot as a tool.

Regulatory Compliance: Stay updated with evolving regulations and standards related to robotics and ensure ongoing compliance.

Market Deployment: Explore avenues for commercialization and mass production if the project holds potential for consumer or industrial markets.

By pursuing these future directions, the self-balancing two-wheeler robot project can continue to evolve, offering new features, applications, and opportunities for innovation while addressing the evolving needs of users and industries.

10 References

1. Juang, H. S., & Lum, K. Y. (2013, June). Design and control of a two-wheel self-balancing robot using the Arduino microcontroller board. In 2013 10th IEEE International Conference on Control and Automation (ICCA) (pp. 634-639). IEEE

2. An, W., & Li, Y. (2013, December). Simulation and control of a two-wheeled self-balancing robot. In 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO) (pp. 456-461). IEEE.