AMA International University BAHRAIN

SALMABAD, KINGDOM OF BAHRAIN

COLLEGE OF ENGINEERING

SELF-BALANCING ROBOT

Design Project Report
Submitted to the Faculty
of the
College of Engineering
in
Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Science
in
Mechatronics Engineering

By:

Faizaan Mohammed Mustafa BH16500026 Bachelor of Science in Mechatronics Engineering

December 2020

AMA INTERNATIONAL UNIVERSITY
KINGDOM OF BAHRAIN

COLLEGE OF ENGINEERING

ACCEPTANCE SHEET

The design project concept entitled **"SELF-BALANCING ROBOT"** has been completed and submitted by the proponent:

Faizaan Mohammed Mustafa **BH16500026**

And in recommended for acceptance by:

DR. ZEYAD MOHAMMED A. ISMAIL, Ph.D.

Advisor

After a thorough review and evaluation of the Engineering Project criteria, the committee for the Engineering Design Project has accepted the title.

Accepted this 1st Trimester, School Year, 2020-2021.

DR.BEDA ALETA, Ph.D.

Dean, College of Engineering

Date of Acceptance: _____

Trimester : 1st Trimester

School Year : 2020-2021

AMA INTERNATIONAL UNIVERSITY
KINGDOM OF BAHRAIN

COLLEGE OF ENGINEERING

APPROVAL FOR ORAL DEFENCE

Faizaan	**Mustafa**	*FAIZAAN*
F. Name	G. Name	Signature

Has successfully passed the Engineering Project Pre-Defence and is recommended for Oral Presentation.

Recommending Approval:

DR. ZEYAD MOHAMMED A. ISMAIL, Ph.D.

Design Project Advisor, Design Project Coordinator

AMA INTERNATIONAL UNIVERSITY
KINGDOM OF BAHRAIN

COLLEGE OF ENGINEERING

APPROVAL FOR ORAL PRESENTATION

In partial fulfillment of the requirements for the Degree of Bachelor of Science in Mechatronics Engineering **Faizaan Mohammed Mustafa** who is hereby recommended for oral presentation has submitted this Engineering Design Project entitled **"Self-Balancing Robot"**.

1-12-2020 — DR. ZEYAD MOHAMMED A. ISMAIL, Ph.D.

Date — Design Project Advisor, Design Project Coordinator

PANEL OF EXAMINATION

Approved in partial fulfillment of the requirements for the degree of Bachelor of Science in Mechatronics Engineering by the Oral Examinations Committee for the Engineering Design Project with the grade of _____ on _____ .

DR. NOAMAN M. NOAMAN, Ph.D. — Chairman

DR. ZEYAD MOHAMMED A. ISMAIL, Ph.D. — Coordinator

DR. RAJKUMAR PALANIAPPAN, Ph.D. — Panel Member

ENG. BILAL AHMED — External Panel Member

Accepted as partial fulfilment of the requirement for the Degree of Bachelor of Science in Mechatronics Engineering (BSME).

Dr. BEDA ALETA, Ph.D.

Dean, College of Engineering

ABSTRACT

In this project, I aimed to build and program a self-balancing robot within the Coppelia Simulation software. By using Coppelia Sim., a new door opens for designing, programming, and constructing any robot we like without any hurdles such as buying and waiting for components during the current scenario where the world is affected by the coronavirus pandemic. The construction and programming of the robot occurred in a virtual scene environment of Coppelia Simulation with the ease of working in any physical environment. At the end of the project, I was able to build a self-balancing robot that withstood external disturbances and balanced itself successfully.

Keywords: self-balancing robot, Coppelia simulation, robot, coronavirus pandemic.

DEDICATION

I would like to dedicate this research to my family and to my professors who have constantly guided me and supported me in all that I desired to do to grow myself as an engineer.

ACKNOWLEDGMENT

I would like to thank my family, friends and my advisor Dr. Zeyad for their constant support throughout the journey despite the difficulties that I had come across during the duration of this project.

Student Name: Faizaan Mohammed Mustafa

Student No.: BH16500026

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

LIST OF EQUATIONS

CHAPTER 1
INTRODUCTION

The world today is surrounded by ideas and methods for solving problems within the fields of engineering, architecture, medicine, and more. One such problem is the classical inverted pendulum problem that involves balancing an inverted pendulum to keep the pendulum upright without falling. Many control theories have been developed to control the unstable system of an inverted pendulum. For the final thesis of the student, it is intended to create a self-balancing robot that can solve the inverted pendulum problem and that can balance itself on a surface without falling on the floor. With the use of motors and sensors, a robot can be designed and programmed to efficiently solve the inverted pendulum problem.

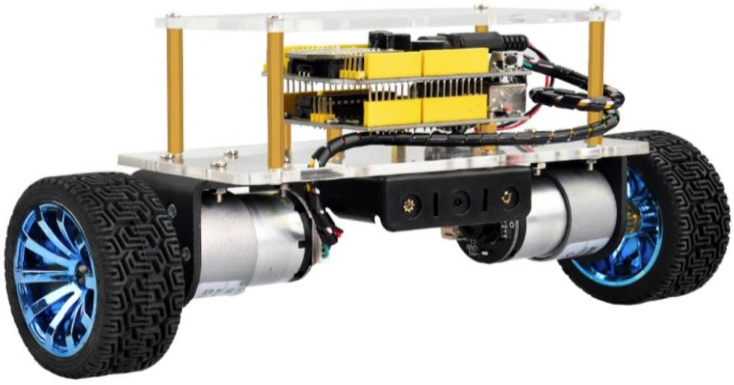


Figure 1.1 Self-Balancing Robot Model(1).

CHAPTER 2

BACKGROUND OF STUDY

A Self-balancing robot has a fundamental characteristic of solving the inverted pendulum problem by understanding the angle at which it is tilting forward and backwards. The inverted pendulum problem involves an inverted pendulum that needs to be balanced with its tilt angle being at 0 degrees. The problem involves correcting the angle of tilt with a suitable closed-loop control architecture that may assist in achieving a balanced system. This project involves exploring the different solutions to the inverted pendulum problem and finding the most suitable one for building a self-balancing robot.

2.1 OBJECTIVES OF THE STUDY:

1. To design a suitable control architecture for solving the inverted pendulum problem.

2. To build a robot that uses the designed control architecture to balance itself with the use of sensors such as a gyroscope and accelerometer.

3. Constraints: For this project, a small and compact robot will be built.

2.2 SIGNIFICANCE OF THE DESIGN PROJECT:

The significance of this project relates to how much the inverted-pendulum problem relates to many worldly applications. These applications include rocket launch applications, where the rocket has to balance its trajectory and keep itself focused on the path to its destination. Another application could be that of a commercial aeroplane which needs to keep itself

steady on its path while withstanding the air pressure of the wind that comes from the front towards the aeroplane. Such applications draw the importance of why designing a stable control architecture is essential for operation in our daily activities. By developing a control architecture, and also a self-balancing robot; the student will learn how to solve complex problems such as the inverted pendulum problem.

2.3 SCOPE AND DELIMITATION:

2.3.1 SCOPE:

This project aims to build a self-balancing robot with the implementation of a control architecture for solving the inverted pendulum problem where the mass of the robot is like an inverted pendulum that needs to be balanced on a surface without falling. The self-balancing robot's mobility and portability make it a very useful object for applications where space is limited. Applications such as delivering light cargo or robots that move around and communicate with other people for giving them directions are a few of the many applications where self-balancing robots can play a significant role.

2.3.2 DELIMITATION:

While the project involves building a self-balancing robot, here are a few limitations that are associated with this project. The first limitation is that in this project, only the self-balancing robot is built to balance the mass of the robot. The second limitation is that we cannot add a lot of weight to the robot as that can make the system unstable. Lastly, the robot will be tested on a flat surface and not on surfaces that are uneven as the latter would cause the robot to struggle in achieving a stable and balanced form.

2.4 DEFINITION OF TERMS:

1. **Gyroscope**- A gyroscope is a sensor that measures the angular accelerations of an object that rotates in the x, y, and z directions. It will be used to find the tilt angle of the robot.
2. **Accelerometer**- An accelerometer is a device that measures linear accelerations in the x, y, and z directions. It will be used to find the tilt angle of the robot.
3. **DC Motor**- A DC motor is a motor that converts electrical energy to mechanical energy. The motor is commonly used in many appliances such as electric vehicles, elevators, hoists, and more.
4. **Stepper Motor**- This motor rotates in steps as a signal is sent to it from the motor controller. It has high holding torque and is used in many devices such as telescopes, hard drives, antennas, and more.
5. **PID Controller-** The PID controller nicked the Proportionality, Integral, and Derivative controllers that can be used for error correction. In this project, the PID controller is utilized for correcting the error in the tilt angle of the robot.
6. **Motor Controller**- The motor controller is used for sending pulses to move the motor and also to control the direction in which the wheel's shaft is rotating. The use of a motor controller will be essential in controlling the movement of the motors connected to the wheels.
7. **Center of Mass-** The centre of mass is the point that is at the centre of the distributed mass. It is the point where if a force is applied, it would cause a linear acceleration in a straight path without any angular acceleration.

CHAPTER 3

REVIEW OF LITERATURE

This chapter discusses the literature that is published on developing the self-balancing robot in different ways.

3.1 CONCEPTUAL LITERATURE:

The research that was done was mostly related to analyzing how the inverted pendulum problem can be solved and how other researchers have attempted their ideas in getting the robot to balance itself. The literature is mostly related to the different controllers that have been used for calculating the tilt angle of the robot. This is essential as the angle of tilt dictates the direction which the motors need to move towards to balance the robot. The efficiency of all the controllers that had been witnessed was compared and analyzed to find the most suitable controller for the current project.

3.2 RESEARCH LITERATURE:

- **FOREIGN STUDIES:**

(1) Sun Wenxia and Chen Wei from Qingdao University of Science & Technology had published their paper titled “Simulation and debugging of LQR control for two-wheeled self-balanced robot” in the IEEE Xplore library where they used the Newton Euler method for linearizing the inverted pendulum problem by building a model that included the mass of the robot and the mass of the wheels, the torque of the wheel, acceleration, and the angular accelerations of the motors(2). It can be noted from this research that the

LQR(Linear Quadratic Regulator) controller was able to provide good results as the tilt angle of the robot returned to 0 degrees in almost 1.6 seconds. This shows that the LQR controller is another controller that can be used in controlling the two-wheeled robot.

(2) Xiaogang Ruan, Jianxian Cai, and Jing Cheno from the Beijing University of Technology, Institute of Artificial Intelligence and Robotics published a paper titled "Learning to Control Two-Wheeled Self-Balancing Robot Using Reinforcement Learning Rules and Fuzzy Neural Networks" in the IEEE Xplore library where they developed a novel approach to balance the two-wheeled by using Reinforcement learning(RL) and Fuzzy Neural Networks(FNN)(3). The concept of Q-learning, where a system learns from its environment and develops itself to earn rewards by correcting itself is also implemented by them as they try to solve the inverted-pendulum problem. They compared their FNN and RL models with the classic PID model and discovered that the FNN and RL model was able to stabilize the robot efficiently with fewer oscillations when compared to the PID controller.

(3) A Youtuber named Joop Brokking developed a PID based self-balancing robot that had a complementary filter included for more accurate measurements of the tilt angle(4). His model consisted of a robot that would start balancing itself as the robot is lifted to a certain tilt angle. The robot used stepper motors which kept the robot fixed at a specific position when the motors weren't moving. The system was very stable and was also successful in solving the inverted pendulum problem.

(4) A. N. K. Nasir, M. A. Ahmad, and R. M. T. Raja Ismail are authors of a paper titled "The Control of a Highly Nonlinear Two-wheels Balancing Robot: A Comparative Assessment between LQR and PID-PID Control Schemes" published in the World Academy of Science, Engineering and Technology Journals (Vol:4, No:10) in 2010 where they reported their development of LQR and PID controllers and measured the efficiency of both controllers(5). They presented the control structure of both controllers and found out that both controllers were capable enough to balance the robot but the LQR was slightly better in balancing the nonlinear system of the self-balancing robot.

(5) Victor Vicente Abreu from the University of Madrid developed a balancing robot that was modelled in the MATLAB software using the LQR controller(6). The robot used a PID controller with the inclusion of a Kalman filter for more accurate measurements of the tilt angle. From their tests, they noted as they added more mass to the top-centre position, the robot became more unstable and struggled in balancing itself as the centre of mass had moved away from the vertical axis.

- **LOCAL STUDIES:**

(6) Amin Alqudah and Asseel Qasaimeh from Yarmouk University, Irbid, Jordan reported a journal where they discussed how they enhanced and tested the PID controller(7). They combined fuzzy logic with the controller to enhance the output as the fuzzy logic allowed control over the gain and time constant during different operation times.

(7) Emerson Arabia's DeltaV controllers are controllers that are used in oil and gas plants for plant operating procedures(8). The controllers make use of self-tuning, neural networks, and fuzzy logic with the inclusion of PID controllers for efficient control of parameters such as gas level, or the amount of oil that is passed through a pipe. The controllers establish a plant control standard and also provide a library of control templates, designed for the plant to be used or reused.

(8) H. M. Omar, A. M. Elalawy, and H. H. Ammar published their paper titled "Two-wheeled Self-balancing robot Modeling and Control using Artificial Neural Networks (ANN)," in the IEEE Xplore platform where a mathematical model was developed and the state space was developed to model the plant of the system(9). Since the system of a Self-balancing is non-linear, the used Nonlinear Autoregressive Exogenous (NARX) Neural Network model with the use of recorded data architecture. They were able to conclude the NARX model was better than the mathematical model. For controlling the self-balancing robot, two controlling architectures including the PID Controller and the Model Reference Adaptive Control(MRAC) were considered. The two controllers were tested with the result being that both controllers are suitable for the control problem but MRAC gave the best response when random inputs were given with good disturbance rejection.

(9) Ahmed, O., El-Amin, E., Salem, A., and Mohammed, A. of the Sudan University of Science and Technology College of Engineering School of Electrical and Nuclear Engineering developed a mathematical model of a two-wheeled self-balance robot system using physical and electrical laws(10). After linearizing the model, they were able to simplify the model. By manually tuning the PID controller of their robot, they were able to

determine the Kp, Ki, and Kd constants for the best system response. The best response was found at: Kp= 40, Ki= 1 ,and Kd= 1. They observed high oscillation around the setpoint with the robot not settling down. After passing a certain angle, their system would become unstable with the robot not regaining balance. Yet, the results obtained from the manual tuning was acceptable and had met their expectation.

(10) Yunus Celik and Mahit Güneş from Karamanoğlu Mehmetbey University, Turkey developed a robot that can track colourful objects while balancing itself on two wheels(11). The robot was balanced using a PID controller with the optimal values for the controller found using the trial and error method. They used the Kalman filter to make the robot stable and less noisy and made use of a colour detection algorithm where a blue object was chosen as the object to track while the robot balanced itself.

3.3 SYNTHESIS:

From the research, it was learned that there are various methods of solving the inverted pendulum problem with the PID controller, the LQR controller, and methods that integrate or customize both PID and LQR controllers. The controller that I have chosen is a PID controller as I have used parts of the PID controller for my previous projects.

It was also noted that mass distribution is essential for the stability of the robot as it tries to balance itself. The best option for the mass distribution would be to place most of the weight near the wheels at the centre of the robot frame to achieve a vertical centre of gravity. This would reduce the chances of instability in motion as the robot tries to balance itself.

CHAPTER 4

DESIGN SPECIFICATIONS

4.1 RESEARCH PARADIGM:

The research paradigm includes the three phases for developing the prototype including the Project Development phase, the Assembly phase, and the Prototype Testing phase.

Phase 1: Project Development	**Phase 2: Assembly**	**Phase 3: Prototype Testing**
Knowledge Requirements - PID Control theory. - Inverted Pendulum Problem. - Arduino Programming. - Coppelia Simulation Scene building and programming. - Mechanical design ***Hardware Requirements:*** - Arduino Microcontroller - Ultrasonic Sensor ***Software Requirements:*** - Coppelia Simulation - Arduino IDE	***Part Assembly*** - Building robot frames within the Coppelia Scene environment. - Programming the ultrasonic sensor using an Arduino - Program the robot in the Coppelia simulation software.	- Test the self-balancing robot.

Figure 4.1. Research Paradigm in Phases.

The project development phase requires the acquisition of fundamental knowledge and resources that can aid in the next phase of the project. An understanding of each requirement is essential for development as every component and resource are interdependent for the final result.

4.2 PROJECT DEVELOPMENT:

The project began with the idea of developing a self-balancing robot with a suitable control algorithm. The research began by understanding the inverted pendulum problem and how it can be solved with the use of different control theories. From personal experience, the PID controller has been well-versed to me and thus, I have chosen the PID based control architecture. Since I needed inputs on how the robot is balancing on its own, I chose to use a gyroscope and an accelerometer to measure the tilt angle of the robot. Initially, the plan was to develop an Arduino-based prototype of the self-balancing robot but the plan had to change due to the COVID-19 pandemic.

4.2.1 PROJECT ISSUES:

Due to the Coronavirus pandemic, a lot of challenges and risks were introduced, especially the risk of receiving the components late and also not having access to an in-person consultation with my project advisor. Also, the closure of the university caused me to not access the university resources on previous thesis papers of students who had worked on a control problem such as the Inverted Pendulum Problem. The solution to these challenges and risks came with distant consultation with my project advisor and online research. My plan changed with the decision to build my robot virtually on a virtual platform known as

Coppelia Sim. and interfacing that robot with a physical Arduino microcontroller for external sensory inputs.

Figure 4.2 Coppelia Sim(12).

Coppelia Robotics is a small Swiss SME active in the field of robotics simulation(13). This is a simulation software where a robot of any kind can be designed, programmed, and tested. Various pre-built robots are already present within the software such as line-following robots, humanoids, and more. In addition to the virtual capabilities that are present, this software can interact with external hardware such as Arduino Uno for receiving sensory inputs. Therefore, such capabilities allowed me to bring changes to my project by turning the focus towards the interaction between reality and virtual reality. Such interaction became possible by adding an external ultrasonic sensor that would measure the displacement of my hand to create a force that is applied to the virtual self-balancing robot within the Coppelia Sim. Software. By doing this, we can verify the robot's resistance to external impact as it tries to balance itself virtually. The Lua programming language has been chosen to program the robot.

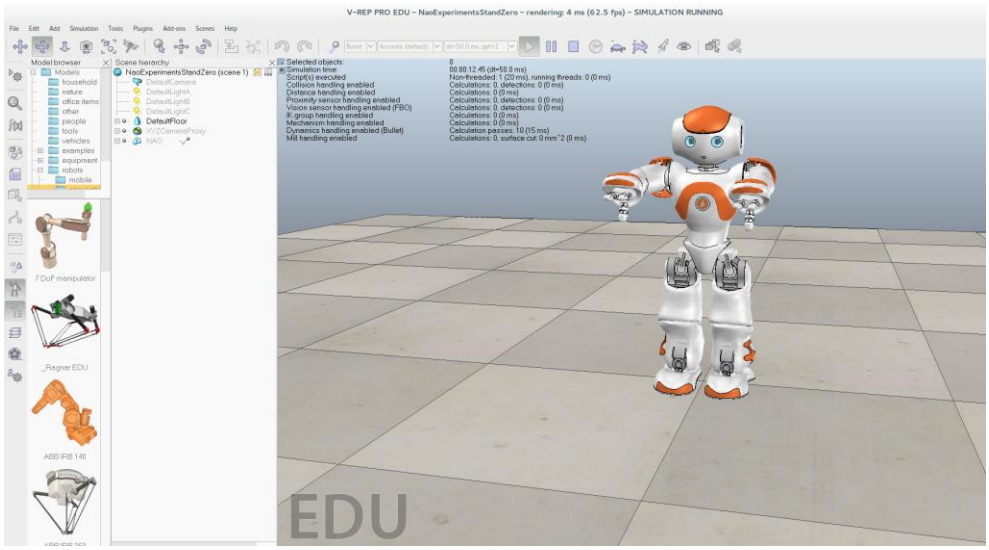


Figure 4.3 A Virtual Humanoid Robot within Coppelia Sim(14).

4.2.2 MATHEMATICAL MODEL:

Before designing the actual robot, it was important to develop a mathematical model by deriving the equations of motion for the inverted pendulum model to understand the non-linear dynamics of this model. The diagram below shows an ideal mass with an inverted pendulum.

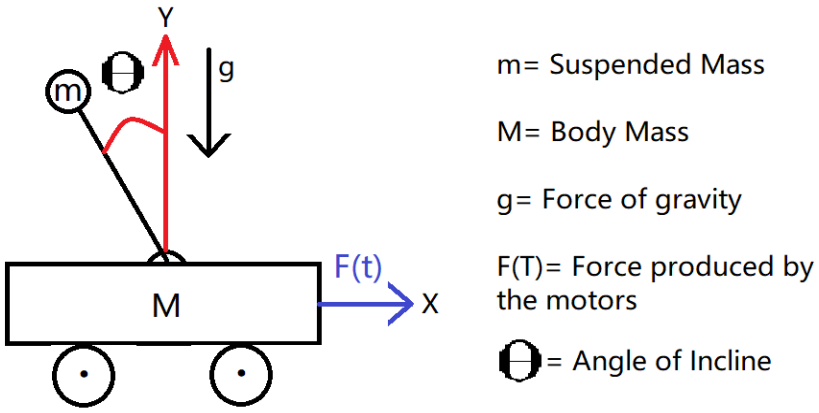


Figure 4.4 Inverted Pendulum Model

Initially, the model is assumed to be on a frictionless surface with a mass “M” that has a force “F" applied to it. The pendulum is made up of a mass “m " with a rigid massless rod attached to the same mass. The force of gravity ‘g’ is present and acts on the model. Using Lagrange’s Equations, the equations of motion are derived for the inverted pendulum model(15). Lagrange’s equations are equations that make the use of kinetic(T) and potential energy(V)for solving equations of motion(16). We

initially begin with the kinematics of the suspended mass ‘m’ on the pendulum across the x-axis and y-axis:

$Xm = X - L*\sin\theta$ **Eq. 4.1**

$Ym = L*\cos\theta$ **Eq. 4.2**

By taking the derivative of equation 4.1 and 4.2 from each side, we get:

$\dot{X}m = \dot{X} - L*\dot{\theta}*\cos\theta$ **Eq. 4.3**

$\dot{Y}m = -L*\dot{\theta}*\sin\theta$ **Eq. 4.4**

Then, we calculate the potential energy(V),

$V = m*g*Ym = m*g*L*\cos\theta$ **Eq. 4.5**

The kinetic energy(T) is then calculated,

$T = \frac{1}{2}*M\dot{X}^2 + \frac{1}{2}*m*(\dot{X}m^2 + \dot{Y}m^2)$

We substitute values for Xm’ and Y m’,

$T = \frac{1}{2}*M\dot{X}^2 + \frac{1}{2}\,m((\dot{X} - L*\dot{\theta}*\cos\theta)^2 + (-L*\dot{\theta}*\sin\theta)^2)$

$T = \frac{1}{2}*M\dot{X}^2 + \frac{1}{2}\,m((\dot{X}^2 - 2*\dot{X}*L*\dot{\theta}*\cos\theta + (L^2*\dot{\theta}^2*\cos^2\theta) + (L^2*\dot{\theta}^2*\sin^2\theta))$

$T = \frac{1}{2}\,(M+m)\dot{X}^2 + \frac{1}{2}\,m(-2*\dot{X}*L*\dot{\theta}*\cos\theta + (L^2*\dot{\theta}^2*\cos^2\theta) + (L^2*\dot{\theta}^2*\sin^2\theta))$

$T = \frac{1}{2}\,(M+m)\dot{X}^2 + \frac{1}{2}\,m(-2*\dot{X}*L*\dot{\theta}*\cos\theta + L^2*\dot{\theta}^2(\cos^2\theta + \sin^2\theta))$

$T = \frac{1}{2}\,(M+m)\dot{X}^2 + \frac{1}{2}\,m((-2*\dot{X}*L*\dot{\theta}*\cos\theta) + L^2*\dot{\theta}^2)$

$T = \frac{1}{2}\,(M+m)\dot{X}^2 + \frac{1}{2}\,mL^2\,\dot{\theta}^2 - (m\dot{X}L\dot{\theta}\cos\theta)$ **Eq. 4.6**

The following are the Lagrange’s equations:

$$\frac{d}{dt}\left(\frac{dL}{d\dot{q}i}\right) - \frac{dL}{dqi} = Qi$$ **Eq. 4.7**

$$L = T - V$$ **Eq. 4.8**

By substituting values for T and V in equation 4.8, we get:

$L =$(½ **(M+m)**$\dot{X}^2$ + ½ **m**L^2 $\dot{\theta}^2$**- ((m**$\dot{X}$**L**$\dot{\theta}$**)***$\cos\theta$) - (**m*g*L***$\cos\theta$) **Eq. 4.9**

Therefore, to attain the first equation of motion; we take the derivative of equation 4.9 in terms of $\dot{X}$**:**

$\frac{dL}{d\dot{X}}$ = **(M+m)**$\dot{X}$**- (m L**$\dot{\theta}$**)***$\cos\theta$= M$\dot{X}$ + **m**$\dot{X}$**- (m L**$\dot{\theta}$**)***$\cos\theta$

Then, in terms of t:

$\frac{d}{dt}\left(\frac{dL}{d\dot{X}}\right)$ = **(M+m)**$\ddot{X}$**- ((m L**$\ddot{\theta}$ **)***$\cos\theta$- **((m L**$\dot{\theta}^2$**)***$\sin\theta$)= F(t)

→ **F(t)= (M+m)**$\ddot{X}$**- (m L**$\ddot{\theta}$ **)***$\cos\theta$ + **((m L**$\dot{\theta}^2$**)***$\sin\theta$) **Eq. 4.10**

Therefore, an equation relating the force applied(F(t)) with the angle of tilt(θ) and the acceleration of the cart($\ddot{\theta}$**)**.

To attain the second equation of motion; we take the derivative of equation 4.9 in terms of $\dot{\theta}$ and then, in terms of t:

$\frac{d}{dt}\left(\frac{dL}{d\dot{\theta}}\right)$ **= (m**L^2 $\ddot{\theta}$ **)- ((m**$L\ddot{X}\cos\theta$) + (**m**$L\dot{X}\dot{\theta}\sin\theta$)

By subtracting the solution $\frac{dL}{d\theta}$, we get:

$\frac{d}{dt}\left(\frac{dL}{d\dot{\theta}}\right) - \frac{dL}{d\theta}$ = **m**L^2 $\ddot{\theta}$ **- (** **m**$L\ddot{X}\cos\theta$)+ (**m**$L\dot{X}\dot{\theta}\sin\theta$) - (**m**$L\dot{X}\dot{\theta}\sin\theta$)

- (**m***L***g**sin θ)= 0 (where $\frac{dL}{d\theta}$ = (**m***L*$\dot{X}\dot{\theta}$sin θ) + (**m***L***g**sin θ)).

= **m*L*²** $\ddot{\theta}$ - **(** **m***L*$\ddot{X}$cos θ) - (**m***L***g**sin θ)= 0

By dividing **m*L*** on **both sides, we get**

$L\ddot{\theta}$ - **(**$\ddot{X}$*****cos θ**) - (g***sin θ**)= 0** **Eq. 4.11**

Finally, the second equation of motion is achieved.

4.2.3 HARDWARE:

The hardware needed for this project includes a microcontroller, a proximity sensor, and a desktop computer with sufficient memory for installing the Coppelia Simulation software.

Figure 4.5 Microcontroller(17) + Ultrasonic(18) + Desktop Computer(19).

4.2.4 PROTOTYPE DESIGN:

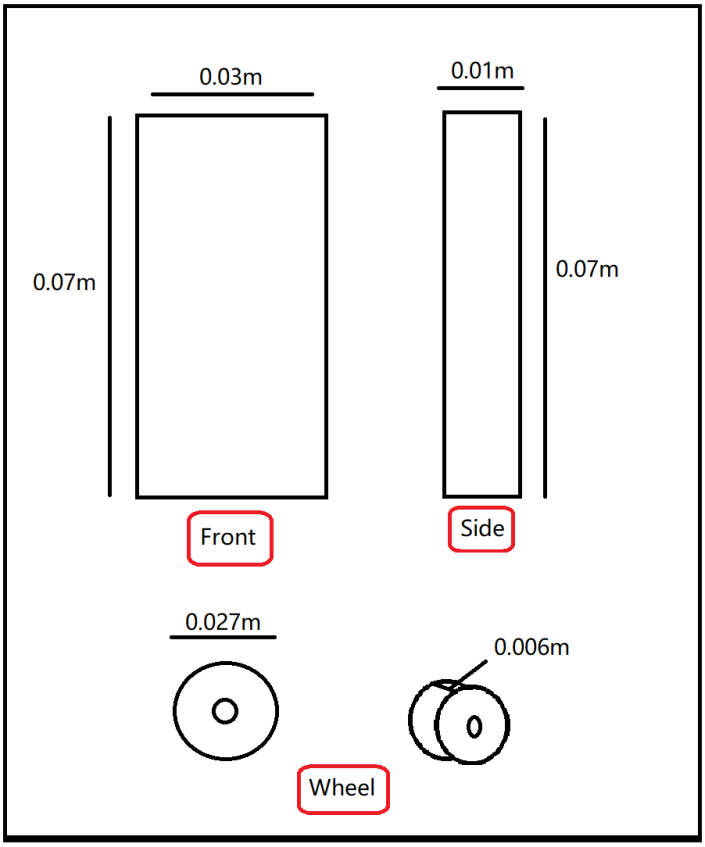


Figure 4.6 Prototype Design.

The prototype design of the self-balancing robot consists of a single frame that has two wheels connected to it. The dimensions of the frame and the wheels have been presented in the diagram above. The motors present in the Coppelia simulation software are attached.

4.2.5 SOFTWARE:

The software needed for this project includes the Coppelia Simulation software and the Arduino IDE.

4.2.6 MICROCONTROLLER:

The microcontroller that has been chosen is the Arduino Uno board due to its compatibility with Coppelia Sim and due to its cost.

4.3 DESIGN STANDARDS:

IEEE Std 100-1996: This paper provides the standard dictionary for electronic appellations. This standard has more than 24,255 terms and definitions associated with electronics. Proponents of this project used this documentation heavily when defining acronyms in the design, development, and documentation of the project at hand.

SAS 123-31 Programming Guidelines: This paper presents a group of programming guidelines and conventions that will be considered in developing code to make sure that it's clean, efficient, transferable, and maintainable.

4.4 MULTIPLE DESIGN CONSTRAINTS:

Design constraints related to cost, time, and flexibility will be presented comparing the options of having a complete physical robot and a virtual robot with hardware additions.

The Figure of Merits analysis is used. Input values, scoring functions, and the weight are the three parameters that are considered for this type of analysis. The inputs include variables such as costs, time-span, and percentage values for the possibility of success. The

weights are assigned by the project designer and can range from 0 to 1. The scoring functions are used to calculate the score and are presented in equation 4.1 and equation 4.2.

The functions require three inputs including the input value, the upper limit of the input value, and the lower limit of the input value. A higher score is achieved from equation 4.1 when there is a high input value and the higher score is preferred for this parameter while a higher score is achieved from equation 4.2 when there is a lower input value. The higher score from the lower input value in equation 4.2 is more preferable than the higher score from the higher input in equation 4.1.

Score=

$$\frac{\boldsymbol{Input\ Value - Lower\ Limit\ of\ the\ Input\ Value}}{\boldsymbol{Upper\ Limit\ of\ the\ Input\ Value - Lower\ Limit\ of\ the\ Input\ Value}}$$

Eq. 4.12 Higher input= Higher Score.

Score=

$$\frac{\boldsymbol{Upper\ Limit\ of\ Input\ Value - \ Input\ Value}}{\boldsymbol{Upper\ Limit\ of\ the\ Input\ Value - Lower\ Limit\ of\ the\ Input\ Value}}$$

Eq. 4.13 Lower input= Higher Score**.**

The following parameters have been utilized for the FoM criterion:

i. **Cost**: The constraints in costs will be presented in BHD.

ii. **Availability**: This parameter refers to the time it takes for the component to arrive from the manufacturer to the proponent's location.

ii. **Reliability**: This will be calculated by determining the failure percentage. The greater the failure rate, the lower the reliability for the device. The failure percentage is represented after combining the failure percentage of each component.

iii. **Safety:** A percentage is assigned to the level of safety each component has. This is assigned by taking into consideration if the component is RoHS compliant or not where there are no hazardous substances such as mercury, cadmium, and other such chemicals that are used in manufacturing electronic devices.

After completing the research on the components, the proponent has identified the following design plan as shown in Table 4.1 below. All the design plans fulfill the required criteria for prototyping the proponent's project.

Table 4.1 Design Plan

Component	**Plan A: Physical Robot**	**Plan B: Virtual Robot***
Hardware	Plywood Sheets- 2 weeks (6 BD)	Virtual Robotic Components (Free)-0
Ultrasonic Sensor	'HC-SRO4' (20)- 2 weeks (5 BD)	HC-SRO4 (5BD)-2
Gyroscope	'GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue' (21) (2 BD)-1 week	Gyro Sensor (Free)-0
Accelerometer	'GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue' (21) (2 BD)-1 week	Accelerometer (Free)-0
Motor	'Geeetech 1.8-Degree Nema 14, 35 BYGHW Stepper Motor for 3D Printer' (22) (11 BD)- 2 weeks	In-built DC Motor (Free)-0
Microcontroller	Arduino Uno (23) (4 BD)-4 weeks	Arduino Uno (4 BD)-4
Power Supply	'E-flite 11.1V 2200mAh 3S 30C LiPo Battery: EC3, EFLB22003S30' (24) (9 BD)- 2 weeks	Computer Power Supply for the robot & 220V external power supply for the Arduino. (Free)-0

*The virtual robot will be moved physically with the use of ultrasonic sensors.

The following parameters were chosen based on the given reasons:

Table 4.2 FoM Parameter Ranges

Parameter	Value	Reasoning
Cost Upper Limit	**700 BD**	This value is roughly the cost of a high-end self-balancing system.
Cost Lower Limit	**30 BD**	The cheapest self-balancing system is priced around this value.
Availability Upper Limit	**4 Weeks**	If no Express shipping is chosen, the shipping takes around 4 weeks.
Availability Lower Limit	**1.5 Weeks**	If Express shipping is chosen, the shipping takes around 1.5 weeks.
Accuracy Upper Limit	**99%**	To achieve 100% accuracy is impossible in the real world yet 99% accuracy is justifiable.
Accuracy Lower Limit	**35%**	Lower than or equal to 35% would certainly lead to failure once the product is in use.
Safety Upper Limit	**96%**	A safety of 100% is not always certain, therefore a percentage of 96 is chosen.
Safety Lower Limit	**52%**	When dealing with safety, the percentage needs to be above 52%. If the percentage goes below this, the prototype can be safe half the time which is not acceptable.

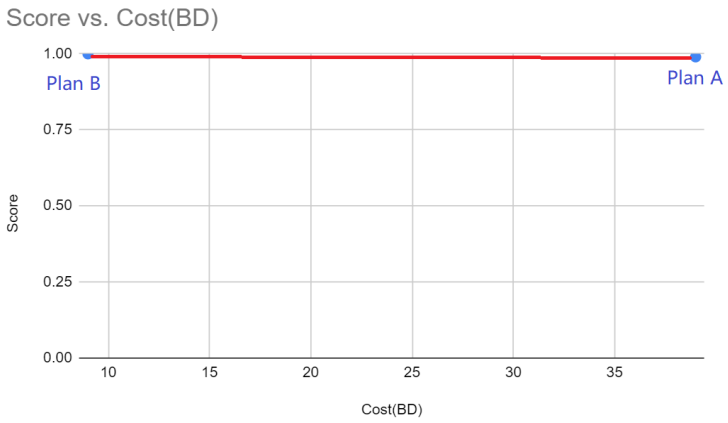
Table 4.3 FoM Plan A

FoM	Plan A			
	Input Value	Score Function	Weight	Score*Weight
Cost(BD)	39	0.99	0.25	0.248
Availability(weeks)	3	0.6	0.25	0.15
Accuracy(%)	75	0.63	0.25	0.16
Safety(%)	85	0.75	0.25	0.19
Total			1	0.8

Table 4.4 FoM Plan B

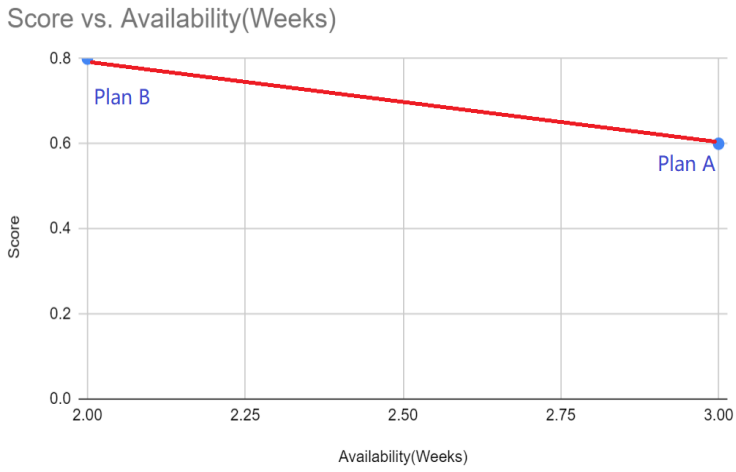
FoM	Plan B			
	Input Value	Score Function	Weight	Score*Weight
Cost(BD)	9	1	0.25	0.25
Availability(weeks)	2	0.8	0.25	0.2
Accuracy(%)	90	0.86	0.25	0.215
Safety(%)	95	0.98	0.25	0.245
Total			1	0.91

Plan B was chosen for having the best parameter values when compared to Plan A. The score for the cost was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.002 points:



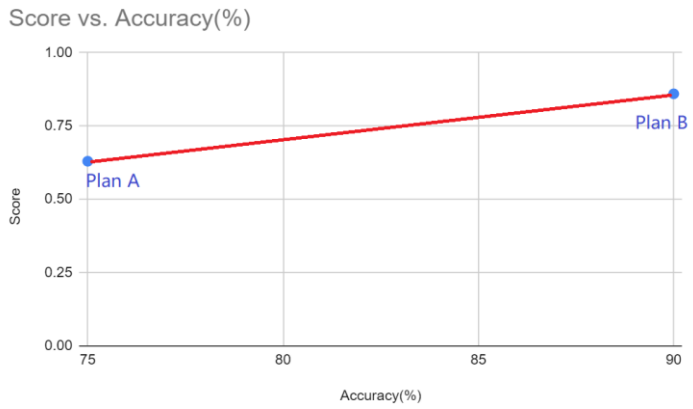
Graph 4.1 Scatter Plot for Score Vs. Cost(BD)

The score for the availability was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.002 points:



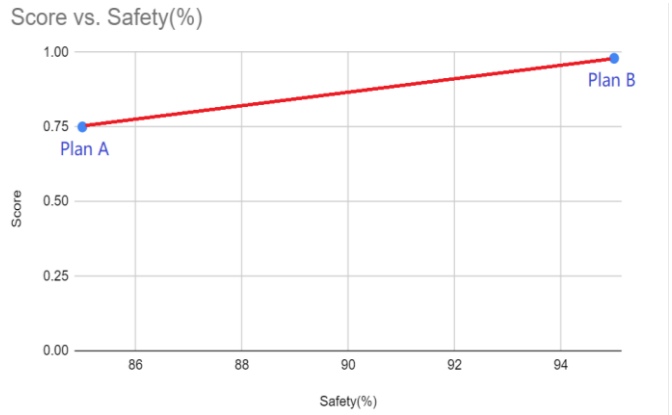
Graph 4.2 Scatter Plot for Score Vs. Availability(Weeks)

The score for the accuracy was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.05 points:



Graph 4.3 Scatter Plot for Score Vs. Accuracy(%)

The score for the safety was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.055 points:



Graph 4.4 Scatter Plot for Score Vs. Safety(%)

In conclusion, the FoM for both plans with consideration of the parameters including cost availability, accuracy, and safety shows that Plan B is a suitable option to go forward with

as it had a higher overall score when compared to Plan A(0.91 > 0.8). This option is a healthy alternative especially considering the project issues that had occurred as mentioned previously.

4.5 PROJECT BLOCK DIAGRAM:

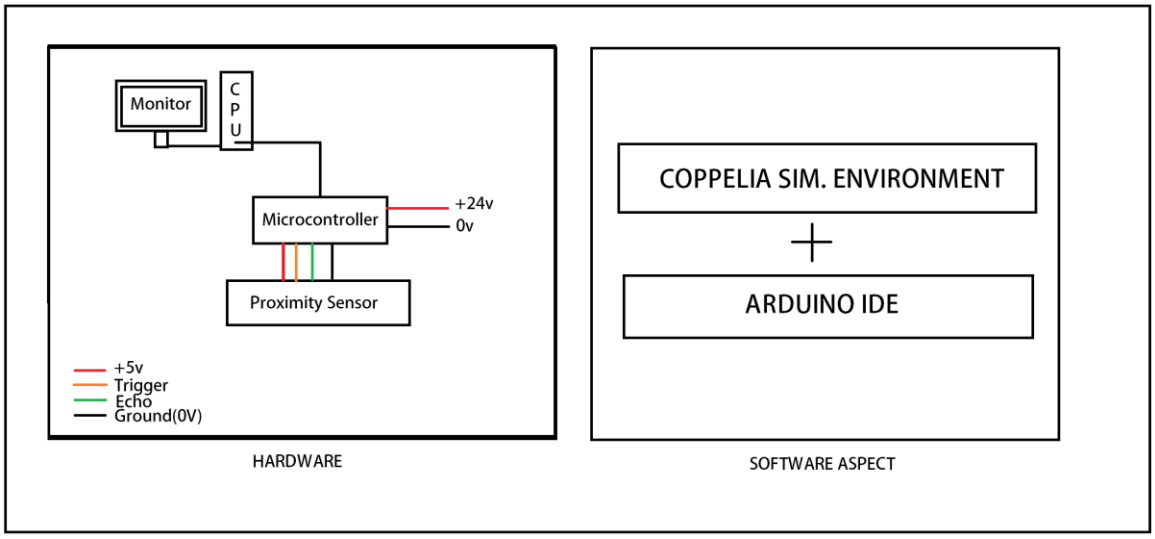


Figure 4.7 Project Block Diagram.

The project block diagram combines the software aspects and the hardware aspects of the robot as shown in the figure above.

4.6 PROJECT FLOWCHART:

The project flowcharts mentioned below include the flow of the main program and also the ultrasonic sensor subroutine for the Self-Balancing robot.

A. Self-Balancing Robot Flowchart:

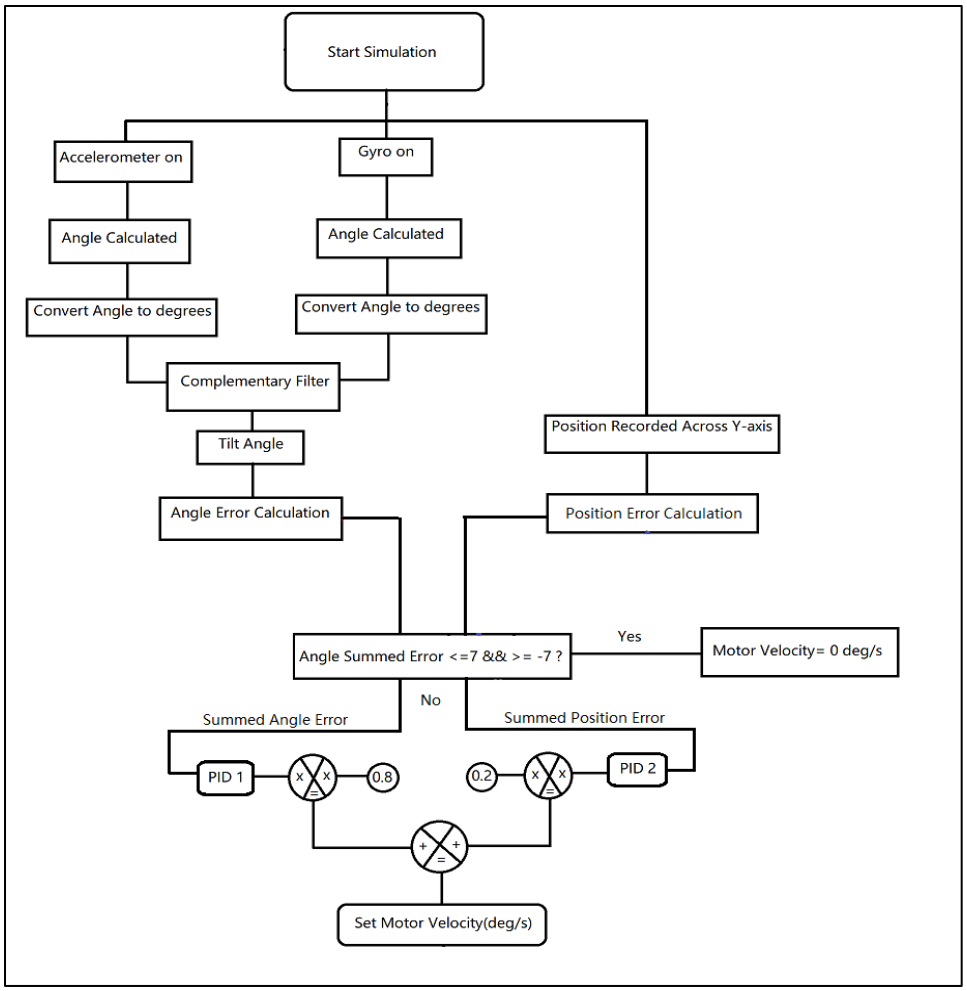


Figure 4.8 Flowchart 1.

B. Ultrasonic Sensor Sub-Routine Flowchart:

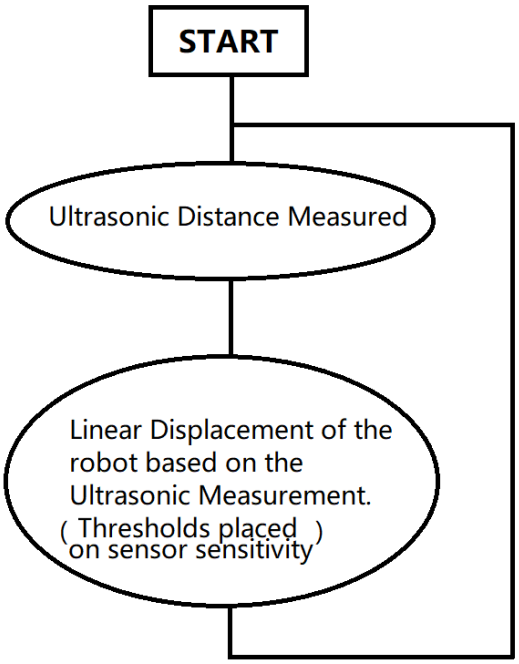


Figure 4.9 Flowchart 2.

4.7 CIRCUIT DIAGRAM:

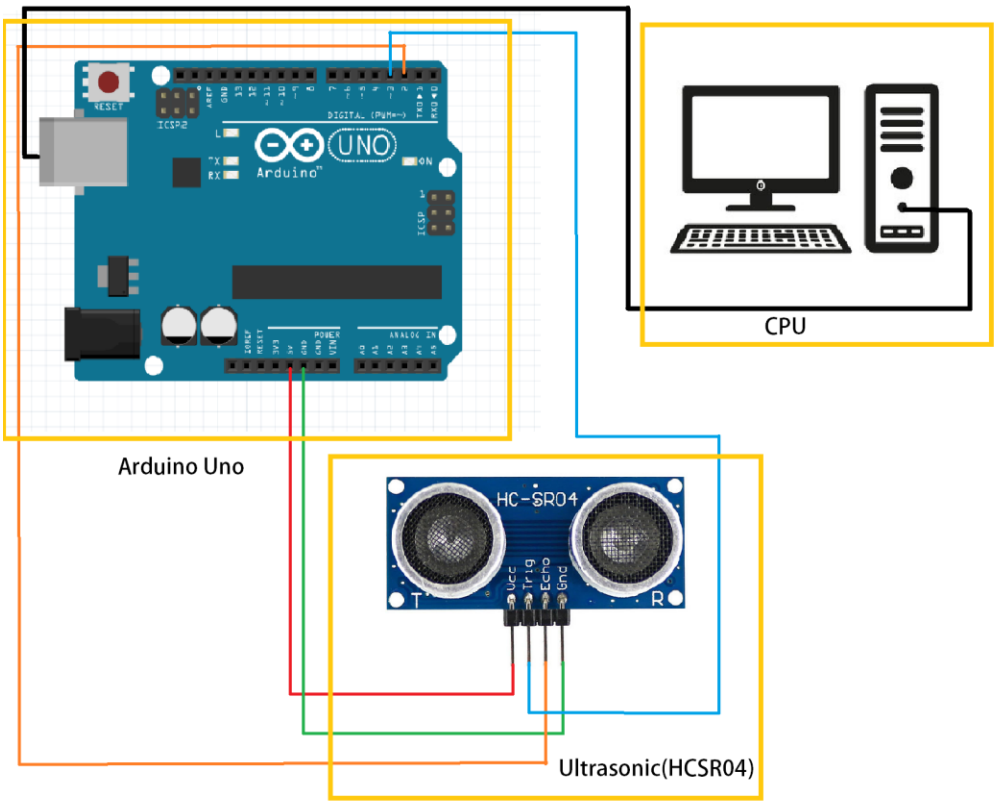


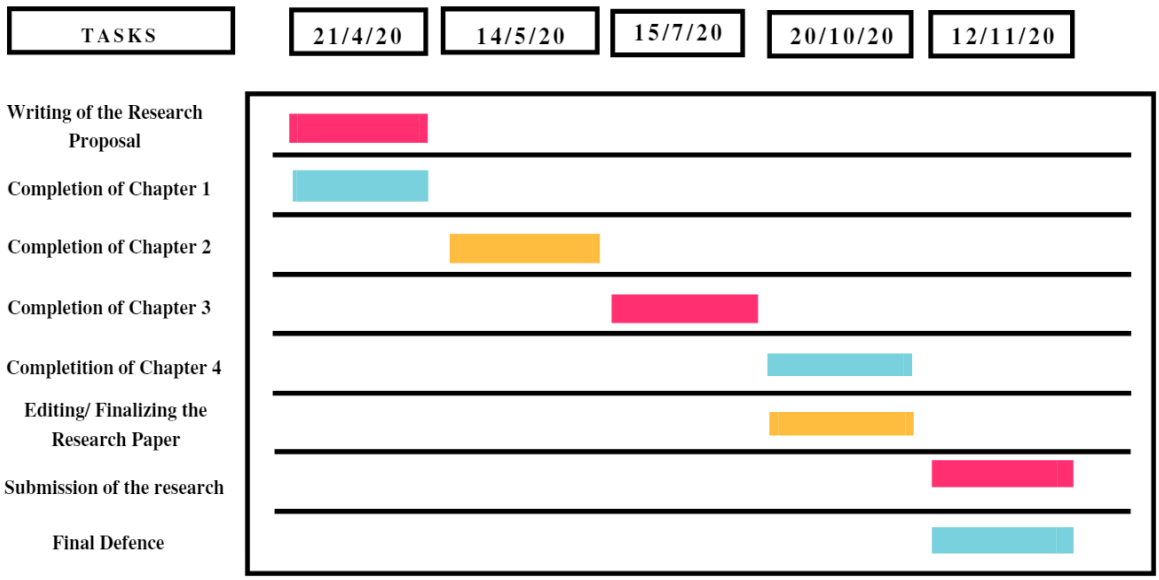
Figure 4.10 Circuit Diagram.

4.8 BILL OF MATERIALS:

Table 4.5 Bill of Materials

Item#	Description	Quantity	Total(BD)
1.	Arduino Uno	1	9
2.	USB 2.0 cable type A/B	1	1.5
3.	Ultrasonic Sensor	1	5
Total(BD)			**15.5**

4.9 GANTT CHART:



CHAPTER 5

DESIGN PROCEDURE, FUNCTIONAL ANALYSIS, AND IMPLEMENTATION

5.1 PROJECT DESCRIPTION:

To build and program the Self-balancing robot, specific hardware and software components are essential. For my Self-balancing robot, the Coppelia Simulation software is used for the creation and simulation of this robot. Within the Coppelia Simulation software, I will be using two motor-enabled rotary joints for movement, two wheels that will be attached to the joints, one Accelerometer sensor, a cuboid-shaped body for the chassis of the robot, and a dummy that is attached to the frame of the robot. The dummy and the accelerometer will be used to measure the angle of tilt of the robot. The Gyro sensor is not used due to the inaccuracy of the values found. The Arduino Uno and the Ultrasonic Sensor HC-SR04 will also not be used in creating external disturbances. Instead, a cuboid will be translated across the x-axis and the y-axis to create the external disturbances. The Lua programming language within the Coppelia Simulation project will be used to program the Self-balancing robot.

5.2 PROJECT FUNCTION ANALYSIS:

The Self-balancing robot begins functioning when the “Start/resume” simulation button is pressed within the Coppelia Simulation Project. After pressing this button, the simulator and the Accelerometer begin measuring the Euler angle around the x-axis and the linear acceleration(across the y and z-axis) respectively. These values are then used to calculate

the angle of tilt of the robot. Once the angle of tilt is determined, the error that occurs when that angle deviates from the set angle or the setpoint is found. The Setpoint for the robot is 0 degrees which are when the robot is perfectly balanced. Once the error is found, the error is summed with the previous error to find the total accumulated error(The summed error is initially zero when the simulation starts). After finding the summed error, this error is fed into a PID controller for achieving an output that is given to the motor enabled joints. A factor of 0.8 is multiplied with the calculated value.

The position of the frame of the robot is also recorded across the y-axis. The initial position of the robot is subtracted with the current position of the robot to get any error that may occur if the robot begins moving from its initial position while balancing itself. This error is then recorded and summed. The summed error is then fed into a second PID Controller and the output is multiplied by a factor of 0.2.

The outputs from both PID controllers are summed to find the appropriate target velocity for the motor joints in degrees. The motor enabled joints then move accordingly to balance the robot while trying to maintain the robot's position. Thus, a self-balancing robotic system is achieved.

While the Self-balancing robot tries to balance itself virtually, a light cuboid-shaped body is translated across the x and y-axis about the world frame to create external disturbances on the self-balancing robot.

5.3 PROJECT IMPLEMENTATION:

The project implementation procedure is as follows:

1. The Coppelia Simulation software is installed on a computer.
2. Within the Coppelia Simulation environment, the chassis of the robot is designed by shaping a cuboid in the dimensions mentioned within the Prototype Design Section in Chapter 4.
3. The wheels that have the rotary joints attached are connected to the chassis.
4. The Accelerometer sensor is attached to the centre of the chassis.
5. A 5g cuboid is placed near the self-balancing robot.

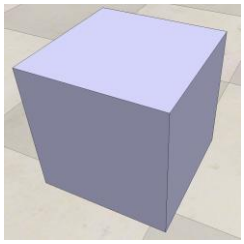


Figure 5.1 Cuboid For Shaping the Chassis

Figure 5.2 Motor Enabled Rotary Joint

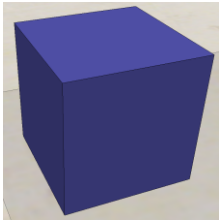


Figure 5.3 Accelerometer

5.4 EVALUATION PROCEDURES:

For the evaluation procedure, the Failure Mode And Effects Analysis(FMEA) are performed:

Table 5.1 Evaluation Procedure

Item #	Component Description	Failure Mode	Effects	Safeguard	Actions
1	Chassis	Disconnects with other parts when the movement starts	No movement occurs as the robot breaks down.	Ensure proper connections are made between the chassis and the remaining parts	Stop the robot Simulation and check all the connections made to the robot.
2	Dummy	Not giving proper Euler angle around the x-axis. Not giving accurate position values in the y-axis.	The tilt angle cannot be found using this component	Ensure that the dummy is placed at the centre of the chassis for accurate measurement.	Stop the robot Simulation and check the connection and position of the dummy in the scene hierarchy.

3	Accelerometer	Not giving acceleration values. Not giving accurate acceleration values	The tilt angle cannot be found from the accelerometer measurement.	Check the connections/programming.	Stop the robot Simulation and check the connections of the Accelerometer sensor. Also, check the Accelerometer sensor script and the Main program script within Coppelia Sim.
4	Rotary Motor Joints	Joints breaking when the movement starts. Wheels disconnect when the movement starts.	No movement occurs as the wheels of the robot disconnect.	Check the connections/programming.	Stop the robot Simulation and check the connections of the rotary joint with the wheels.
5	Computer System	No power	The complete shutdown of the project	Check the connections	Check the source of power and ensure all connections of the computer are properly made.

5.5 COMPONENT SPECIFICATIONS:

1. Coppelia Simulation:

The Coppelia Simulation software is a simulation software where robots can be built and programmed in many ways. The software offers different ways to program robots either locally using the Lua programming language or by using the remote API interface to program from software like MATLAB, Python, or Java or by using the ROS(Robot Operating System) communication method. For my program, I have used the local programming language Lua to program my Self-balancing robot. The Newton physics engine is used for the dynamics of the robot with very accurate simulations.

Within the software, a scene is created where the robot can be built and simulated. Within the Scene, a Scene Hierarchy is presented which includes all the components of the robot. The components include the sensors(Accelerometer), the main robot frame, a dummy object for obtaining the orientation, and the robot motor joints. Also, within the Scene Hierarchy graphs that present the tilt angle and the error in the change in position respectively. Otherwise, there are default objects created within the software automatically that allow the creation of the floor, default lights, and the cameras for the robot. A side view camera has also been placed to get the side view of the self-balancing robot. The Scene hierarchy has been presented below:

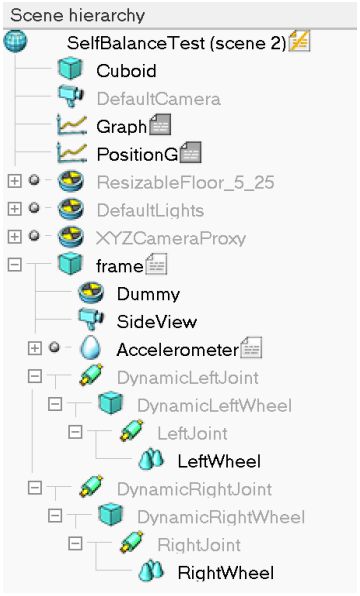


Figure 5.4 Scene Hierarchy.

The Robot that was built within the software has been presented below:

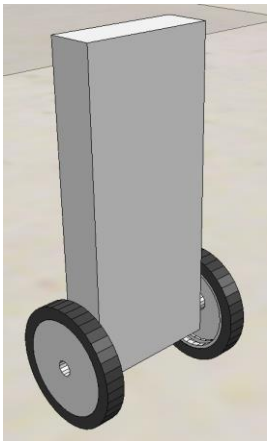


Figure 5.5 Main Self-balancing robot.

Table 5.2 Robot Specifications

Robot Specifications	
Weight of the Frame	0.25 kg
Dimensions(X*Y*Z) of the Frame	0.03m *0.01m*0.07m
Dimension(X*Y*Z) of the wheels	0.006m *0.027m*0.027m
The torque of Motor Joints	5 N*m

The weight of the frame of the robot is 0.25 kg with dimensions 0.03m on the x-axis, 0.01m on the y-axis, 0.07m on the z-axis. The wheels have a diameter of 0.027m across the y-axis and a depth of 0.006m approximately across the x-axis. The motor joints attached to the wheel are To program the robot a child script needs to be created and attached to the frame of the robot as shown below:

Figure 5.6 Child Script.

There are two kinds of child scripts that can be created for the robot including a threaded child script and a non-threaded child script. For my program, I chose a non-threaded child script. Initially, within the non-threaded child script, the function sysCall_init is written. Handles for each object present in the scene are stored in variables using the

'sim.getObjectHandle()'. A tube communication method is initialized to read the accelerometer values by writing the following code:

"accelCommunicationTube=sim.tubeOpen(0,'accelerometerData'..sim.getNameSuffix(nil),1)"

The sysCall_sensing() is then initialized where variables used in the program are declared. An if statement with the condition if '1==1' is created to form a continuous loop. Within the continuous loop, the program for measuring the tilt angle and for controlling the motors to correct the tilt angle is present. The loop time(dt) for the whole program is initialized at 10ms.

To find the angle of tilt of the robot, the Euler angle around the x-axis of the dummy placed inside the frame is measured by using the 'sim.getObjectOrientation' and the output from the accelerometer sensor is measured. This technique replaced the Gyro sensor measurement method as the Gyro sensor was giving angular values that were significantly different when compared to the accelerometer angle values.

The Euler angle around the x-axis is received in radians and therefore, this value must be converted to degrees. To convert the value to degrees, the following equation is used:

Eangle= euler[1]*(180/math.pi) **Eq. 5.1**

Where euler[1] is the value of the orientation of the dummy present in the frame of the robot around the x-axis and where Eangle is the Euler angle in degrees.

To find the angle of tilt from the accelerometer sensor the following relationship is used:

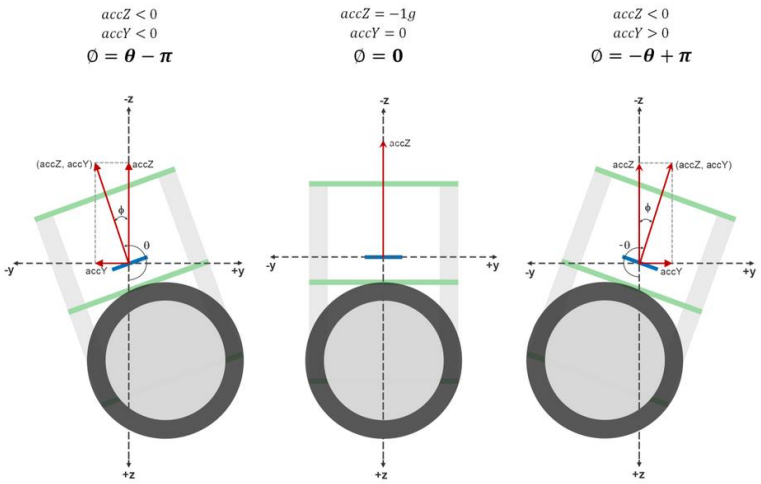


Figure 5.7 Accelerometer Sensor Tilt Angle Relations(25).

From the figure above, the Tilt Angle ϕ can be found by taking the inverse tangent of the linear acceleration across the y-axis and the z-axis. Also, the angle found needs to be converted to degrees. In the Self-Balancing robot program, the following equation was used to find the tilt angle and convert that angle to degrees:

accangle= (math.atan(accY/accZ))*(180/math.pi) **Eq. 5.2**

Where the accangle is the calculated angle from the output of the accelerometer, accY is the linear acceleration across the y-axis and accZ is the linear across the z-axis.

The angle of tilt is then found by using a complementary filter which combines the Euler angle and the accelerometer angle to get the best output by filtering the noise in the output.

The complementary filter is used to find the tilt angle in the equation shown below:

$$\text{TiltAngle} = (0.9999999 * \text{Eangle}) + (0.0000001 * \text{accangle}) \quad \textbf{Eq. 5.3}$$

Where Eangle is the measured Euler angle across the x-axis and accangle is the calculated angle from the output of the accelerometer.

After finding the tilt angle, the deviation of the title angle from the target angle is calculated to find the error by using the following equation:

$$\text{error} = \text{targetAngle} - \text{TiltAngle} \quad \textbf{Eq. 5.4}$$

Where the target angle is 0 degrees. After finding the error, this error is summed with the previous error to find the value for the summed error or errorSumm as shown in the equation below:

$$\text{errorSumm} = \text{errorSumm} + \text{error} \quad \textbf{Eq. 5.5}$$

Apart from measuring the tilt angle, the change in the position of the frame of the robot is also noted by measuring the position of the dummy. The purpose of measuring the position is to keep the robot around its initial position while the robot balances itself. This measurement is made using the "sim.getObjectPosition(dummy,-1)". The position across the y-axis is then found by entering the command " positionY= position[2]". The target position of the robot is its initial position at approximately 0.23m from the reference frame. The error in the position is then found by using the following equation:

$$\text{errorP} = \text{targetPosition} - \text{positionY} \quad \textbf{Eq. 5.6}$$

After finding the error, this error is summed with the previous error to find the value for the summed error of the position or errorSummP as shown in the equation below:

$$errorSummP = errorSummP + errorP \quad \textbf{Eq. 5.7}$$

The summation of the error in the tilt angle and the summation of the error in the position (across the y-axis) are then used to balance the robot to the target angle and keep the robot balancing around its initial position.

To achieve this, 2 PID controllers are used. The PID controller consists of the Proportionality, Integration, and Derivative controllers for correcting the tilt angle of the robot and also for correcting the position of the robot. The following diagram illustrates this relationship:

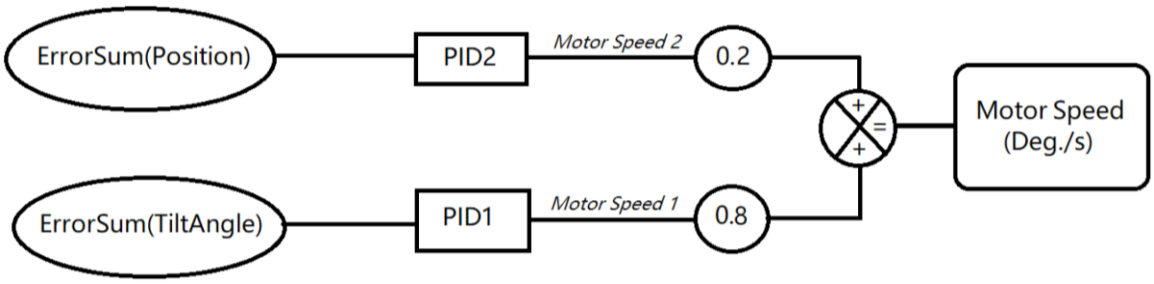


Figure 5.8 PID Controller Diagram.

For the first PID controller; the Kp, Ki, and Kd constants are 45.5, 35 ,and 0.0000045 respectively while for the second PID controller; the Kp, Ki, and Kd constants are 100, 0 ,and 0 respectively. The outputs from both PID controllers are combined to give the output

to the motor joints. The motor speed has been limited from -500 to 500 degree second. The equation for both PID controllers is presented below:

motorPower =(0.8)*((Kp*(errorSumm))+(Ki*(errorSumm)*0.01)-Kd*((TiltAngle-prevAngleF)/0.01))+(0.2)*((Kp1*(errorSummP))+(Ki1*(errorSummP)*0.01)-Kd1*((positionY-prevP)/0.01)) **Eq 5.8**

Apart from the PID correction, a condition has been placed where if the angle of tilt is between -7 and 7 degrees; the velocity would be 0 degrees/s. For creating the external disturbances, a 5-gram cuboid is made with the dimensions 0.01m*0.01m*0.08m is translated to push the self-balancing robot as shown below:

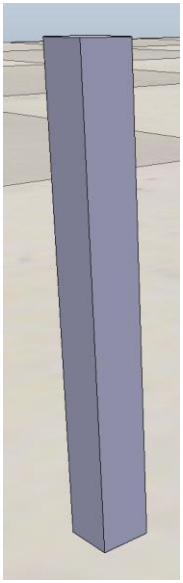


Figure 5.9 5g Cuboid.

5.6 COST-BENEFIT ANALYSIS:

Cost-Benefit analysis is a procedure utilized when making business choices. The benefits or advantages of the business activity are summed, and the expenses related to that business activity is subtracted to achieve the Cost-Benefit analysis.

Table 5.3 Costs

		Fiscal Year					
Program Element	Element Manager	2020	2021	2022	2023	2024	2025
Material Purchasing	Faizaan Mustafa	$50	$75	$120	$320	$520	720
Equipment Purchasing	Faizaan Mustafa	$150	$200	$300			500
Product Installment	Faizaan Mustafa			$150	$150	$150	150
Product Distribution	Faizaan Mustafa			$100	$100	$100	100
Facilities	Faizaan Mustafa	$100	$100	$100	$100	$100	100
Service and Maintenance	Faizaan Mustafa			$80	$80	$80	80
Hardware Development	Faizaan Mustafa			$200	$200	$200	200
Software Develpoment	Faizaan Mustafa	$50	$50	$100	$100	$100	100
Program Total Costs By Year		$350	$425	$1,150	$1,050	$1,250	$1,950
Program Grand Total Cost	$6,175						

As mentioned in Table 5.3, the project sources are broken down into elements to find the total cost. The program source elements include Material Purchasing, Equipment Purchasing, Product Installment, Product Distribution, Facilities, Service and Maintenance, Hardware Development, and Software Development. The total cost of implementing this project is computed for the current year(2020) and the upcoming 5 years.

Table 5.4 Benefits

	Fiscal Year					
Benefit Sources	2020	2021	2022	2023	2024	2025
Cost Reduction			$150	$250	$500	$700
Enhanced Revenues			$3,000		$5,500	$12,500
Labor Reduction					$200	$250
Decreased Overhead				$100	$170	$170
Total Benefits Per Year	$0	$0	$3,150	$350	$6,370	$13,620
Confidence Factor	100%	100%	50%	80%	75%	80%
Benefits Claimed for Analysis	$0	$0	$1,575	$280	$4,778	$10,896
Program Grand Total Benefit	$17,529					

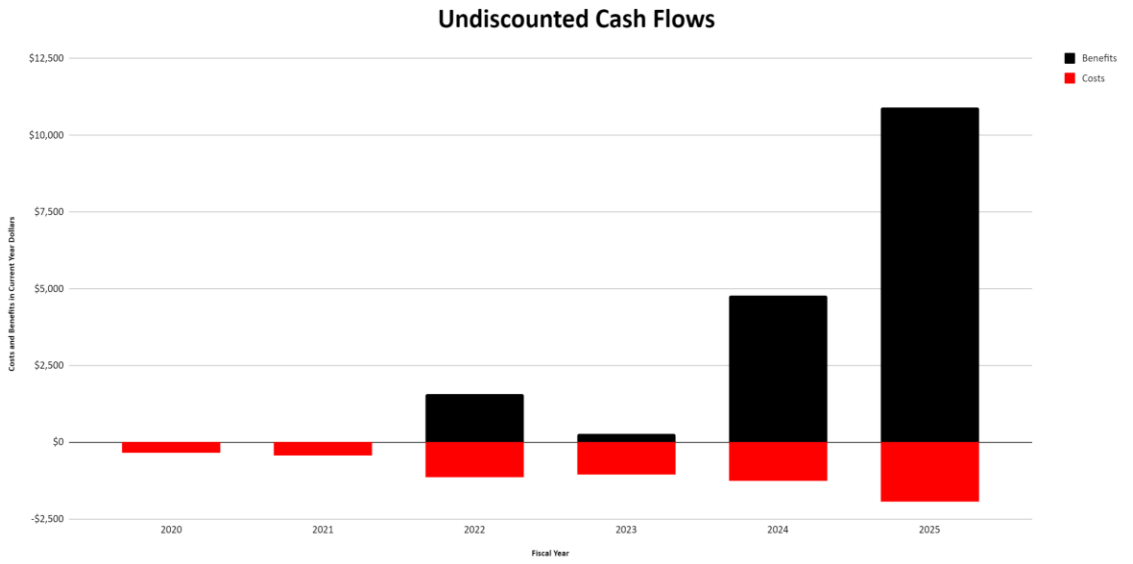
In table 5.4, the total benefit is computed for the project. There are four benefit sources including Cost Reduction, Enhanced Revenues, Labor Reduction, and Decreased Overhead. The benefit for each year of implementation is computed. Initially, the total benefit of each benefit source is summed and then, the total sum is multiplied by the confidence factor to get the benefits claimed for analysis. After finding the benefits claimed, the program's grand total benefit is found by adding the benefits of all the years.

As shown in the table above, no benefits are collected in the first two years as the collection of the benefits is expected to begin from the third year.

Table 5.5 Summary

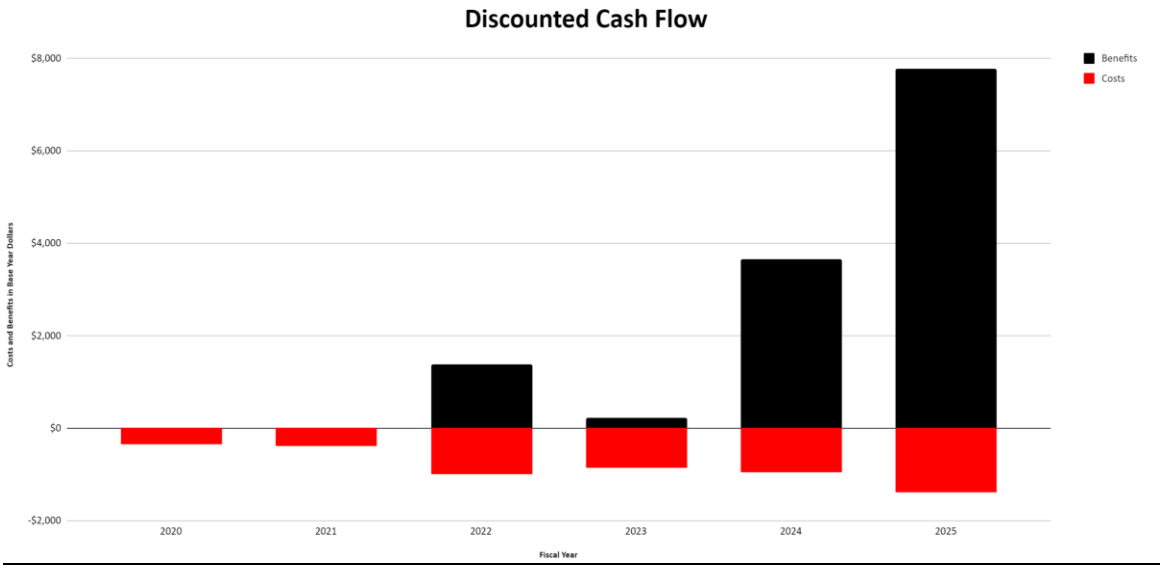
	Fiscal Year					
	2020	2021	2022	2023	2024	2025
Undiscounted Flows						
Costs	-$350	-$425	-$1,150	-$1,050	-$1,250	-$1,950
Benefits	$0	$0	$1,575	$280	$4,778	$10,896
Net Cash Flow	-$350	-$425	$425	-$770	$3,528	$8,946
Discount Factors						
Discount Rate	7.0%					
Base Year	2020					
Year Index	0	1	2	3	4	5
Discount Factor	1.0000	0.9346	0.8734	0.8163	0.7629	0.7130
Discounted Flows						
Costs	-$350	-$397	-$1,004	-$857	-$954	-$1,390
Benefits	$0	$0	$1,376	$229	$3,645	$7,769
Net	-$350	-$397	$371	-$629	$2,691	$6,378
Cumulative	-$350	-$747	-$376	-$1,005	$1,687	$8,065
Net Present Value	$8,065					
Internal Rate of Return	95%					

Table 5.5 presents a summary of the calculations for the undiscounted cash flow and the discounted cash flow. For the undiscounted cash flow; cost, benefits, net cash flow, discount rate, and base year are used to compute the net result. For the discounted cash flow; cost, benefits, net cash flow, cumulative, net present value, and the internal rate of return is used to compute the net result.



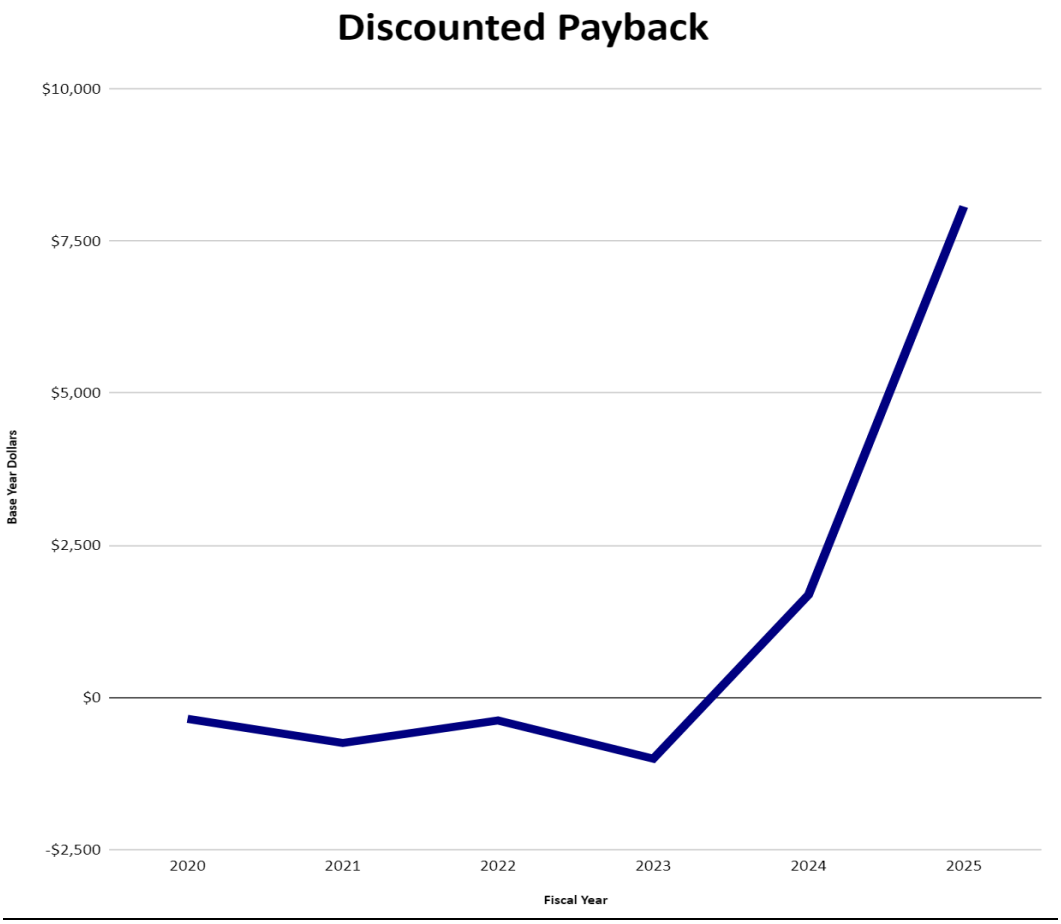
Graph 5.2 Undiscounted Cash Flows

In graph 5.2, the undiscounted cash flow graph is made using the values from table 5.4. The relationship between the cost and benefit is displayed among the several years of implementation of this project. From 2020 to 2021, only the cost is displayed while from 2022 to 2025; both cost and benefits are displayed. It can be noted that from 2024 to 2025, the benefit was constantly greater than the cost.



Graph 5.3 Discounted Cash Flows

In graph 5.3, the discounted cash flow graph is made using the values from table 5.4. The relationship between the cost and benefit is displayed among the several years of implementation of this project. From 2020 to 2021, only the cost is displayed while from 2022 to 2025; both cost and benefits are displayed. It can be noted that from 2024 to 2025, the benefit was constantly greater than the cost.



Graph 5.4 Discounted Payback

In graph 5.4, the discounted paycheck is made using the relationship between cost and benefit for the discounted cash flow. From 2020 to 2023, the cost is very high while the benefit is below zero dollars. From 2023 onwards, the trend of the curve changes with the increase of benefits in 2024 and 2025 as the cost is reduced from 2024 onward when compared to the benefits.

Interest: 0.07
Years: 10
Amount Borrowed: 5000

Table 5.6 Amortization Chart

Payment #	principal	interest	remaining
1	$28.89	$29.17	$4,971.11
2	$29.06	$29.00	$4,942.06
3	$29.23	$28.83	$4,912.83
4	$29.40	$28.66	$4,883.43
5	$29.57	$28.49	$4,853.87
6	$29.74	$28.31	$4,824.13
7	$29.91	$28.14	$4,794.21
8	$30.09	$27.97	$4,764.13
9	$30.26	$27.79	$4,733.86
10	$30.44	$27.61	$4,703.42
11	$30.62	$27.44	$4,672.80
12	$30.80	$27.26	$4,642.01
13	$30.98	$27.08	$4,611.03
14	$31.16	$26.90	$4,579.88
15	$31.34	$26.72	$4,548.54
16	$31.52	$26.53	$4,517.02
17	$31.70	$26.35	$4,485.31
18	$31.89	$26.16	$4,453.42
19	$32.08	$25.98	$4,421.35
20	$32.26	$25.79	$4,389.08
21	$32.45	$25.60	$4,356.63
22	$32.64	$25.41	$4,323.99
23	$32.83	$25.22	$4,291.16
24	$33.02	$25.03	$4,258.14
25	$33.22	$24.84	$4,224.92
26	$33.41	$24.65	$4,191.51
27	$33.60	$24.45	$4,157.91

28	$33.80	$24.25	$4,124.11
29	$34.00	$24.06	$4,090.11
30	$34.20	$23.86	$4,055.92
31	$34.39	$23.66	$4,021.52
32	$34.60	$23.46	$3,986.93
33	$34.80	$23.26	$3,952.13
34	$35.00	$23.05	$3,917.13
35	$35.20	$22.85	$3,881.93
36	$35.41	$22.64	$3,846.52
37	$35.62	$22.44	$3,810.90
38	$35.82	$22.23	$3,775.08
39	$36.03	$22.02	$3,739.04
40	$36.24	$21.81	$3,702.80
41	$36.45	$21.60	$3,666.35
42	$36.67	$21.39	$3,629.68
43	$36.88	$21.17	$3,592.80
44	$37.10	$20.96	$3,555.70
45	$37.31	$20.74	$3,518.39
46	$37.53	$20.52	$3,480.86
47	$37.75	$20.31	$3,443.11
48	$37.97	$20.08	$3,405.14
49	$38.19	$19.86	$3,366.95
50	$38.41	$19.64	$3,328.53
51	$38.64	$19.42	$3,289.90
52	$38.86	$19.19	$3,251.03
53	$39.09	$18.96	$3,211.94
54	$39.32	$18.74	$3,172.63
55	$39.55	$18.51	$3,133.08

56	$39.78	$18.28	$3,093.30
57	$40.01	$18.04	$3,053.29
58	$40.24	$17.81	$3,013.05
59	$40.48	$17.58	$2,972.57
60	$40.71	$17.34	$2,931.85
61	$40.95	$17.10	$2,890.90
62	$41.19	$16.86	$2,849.71
63	$41.43	$16.62	$2,808.28
64	$41.67	$16.38	$2,766.61
65	$41.92	$16.14	$2,724.69
66	$42.16	$15.89	$2,682.53
67	$42.41	$15.65	$2,640.13
68	$42.65	$15.40	$2,597.47
69	$42.90	$15.15	$2,554.57
70	$43.15	$14.90	$2,511.42
71	$43.40	$14.65	$2,468.01
72	$43.66	$14.40	$2,424.36
73	$43.91	$14.14	$2,380.44
74	$44.17	$13.89	$2,336.28
75	$44.43	$13.63	$2,291.85
76	$44.69	$13.37	$2,247.17
77	$44.95	$13.11	$2,202.22
78	$45.21	$12.85	$2,157.01
79	$45.47	$12.58	$2,111.54
80	$45.74	$12.32	$2,065.80
81	$46.00	$12.05	$2,019.80
82	$46.27	$11.78	$1,973.53
83	$46.54	$11.51	$1,926.99
84	$46.81	$11.24	$1,880.17

85	$47.09	$10.97	$1,833.08
86	$47.36	$10.69	$1,785.72
87	$47.64	$10.42	$1,738.09
88	$47.92	$10.14	$1,690.17
89	$48.19	$9.86	$1,641.98
90	$48.48	$9.58	$1,593.50
91	$48.76	$9.30	$1,544.74
92	$49.04	$9.01	$1,495.70
93	$49.33	$8.72	$1,446.37
94	$49.62	$8.44	$1,396.75
95	$49.91	$8.15	$1,346.84
96	$50.20	$7.86	$1,296.65
97	$50.49	$7.56	$1,246.16
98	$50.78	$7.27	$1,195.37
99	$51.08	$6.97	$1,144.29
100	$51.38	$6.68	$1,092.91
101	$51.68	$6.38	$1,041.23
102	$51.98	$6.07	$989.25
103	$52.28	$5.77	$936.97
104	$52.59	$5.47	$884.38
105	$52.90	$5.16	$831.48
106	$53.20	$4.85	$778.28
107	$53.51	$4.54	$724.77
108	$53.83	$4.23	$670.94
109	$54.14	$3.91	$616.80
110	$54.46	$3.60	$562.34
111	$54.77	$3.28	$507.57
112	$55.09	$2.96	$452.48

113	$55.41	$2.64	$397.06
114	$55.74	$2.32	$341.32
115	$56.06	$1.99	$285.26
116	$56.39	$1.66	$228.87
117	$56.72	$1.34	$172.15
118	$57.05	$1.00	$115.10
119	$57.38	$0.67	$57.72
120	$57.72	$0.34	$0.00

The Amortization table displays the amount of money that needs to be paid in intervals to pay the total loan amount. The interest rates are also included in the table. The time allocated for paying the loan is 10 years for a loan amount of $5000. The remaining amount after all the payments are made is also included in the table.

CHAPTER 6

DESIGN PROJECT SUMMARY AND CONCLUSIONS

I. SUMMARY:

The Self-Balancing robot project was very special as it revolved around solving the inverted pendulum problem. The problem involved balancing an inverted pendulum by achieving a tilt angle of 0. There were so many different solutions that were witnessed during the research in the initial stages of the project. After the completion of the research, the PID Controller was eventually chosen by me due to my familiarity with this controller when compared to the other controllers that were used by others.

In software aspects, the Coppelia Simulation offered a great alternative for building the robot especially considering the current COVID-19 situation around the world. The software allowed me to build, compose, and program my robot without any difficulties. Sensors including the gyro sensor and the accelerometer were used to calculate the tilt angle but the former was not used to its inaccuracy in getting angular values. Also, the ultrasonic sensor was not used for creating the proposed external disturbance due to inaccurate transmission of ultrasonic sensor values within the simulation scene. The P (proportionality) controller in the PID also used the summation of the errors for finding the motor output which was theoretically incorrect as I learned later. I had not relied on a mathematical model that had made my process for finding the Kp, Ki, and Kd constants lengthy by trial and error based on the reaction of the self-balancing robot.

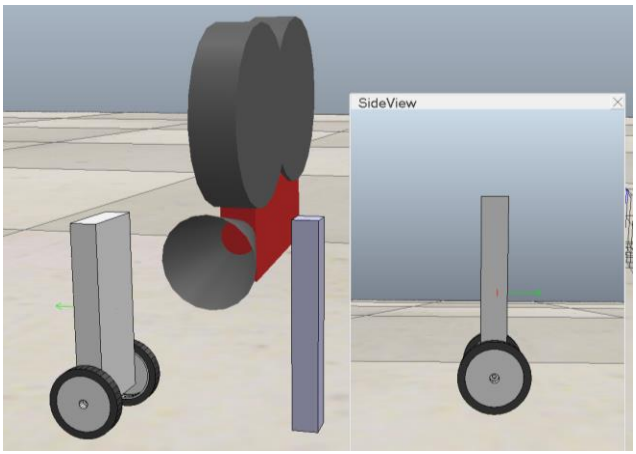
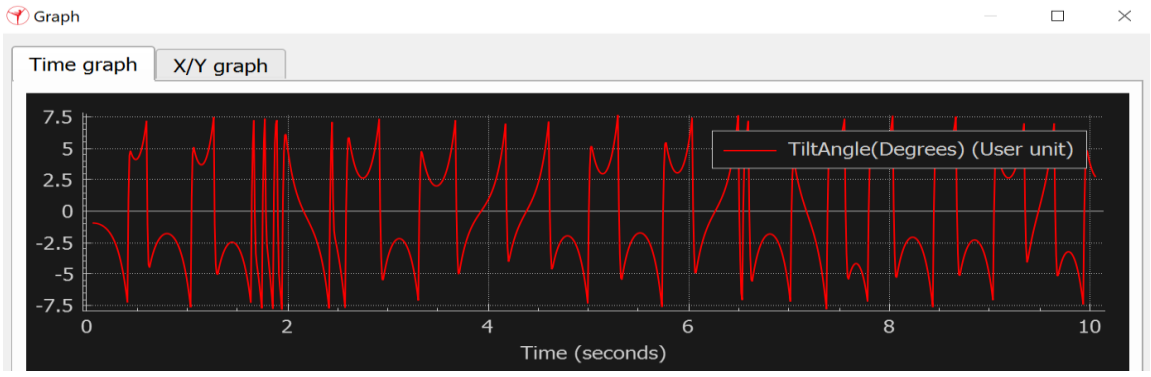


Figure 6.1 Self-Balancing Robot Scene.

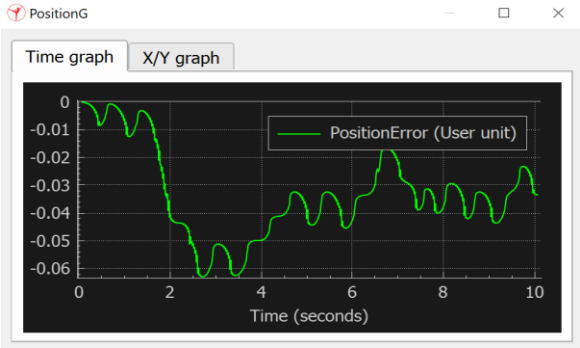
Yet, I was able to find solutions by incorporating Euler angles that were measured by the software using the command ‘sim.getObjectOrientation’. For creating the disturbances, I was able to create cuboid-shaped objects within the scene that would bump with the self-balancing robot to create those disturbances.

When the robot would balance itself, the tilt angle would oscillate between -7.5 to 7.5 degrees as shown in the graph below:



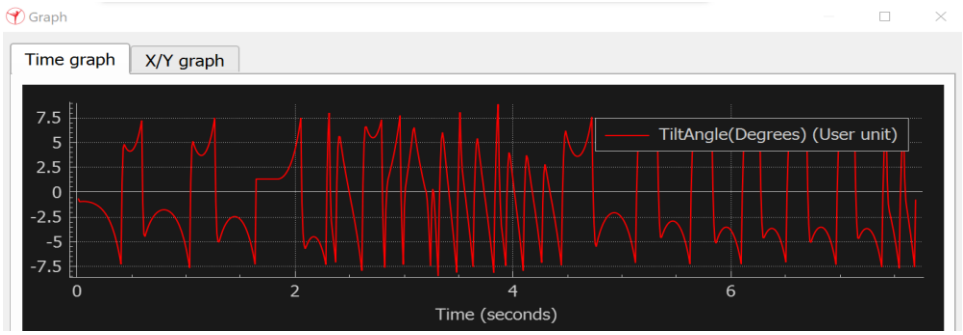
Graph 6.1 Title Angle Change(Time=10 seconds)

Such an output allowed the robot to balance itself without falling moving abruptly. The position of the robot also remained near the initial position as the degree error was almost 0 meter in the y-axis as shown in the graph below:



Graph 6.2 Position Error Change(Time=10 seconds)

The robot was balancing itself and was also able to withstand shock or disturbance that was created when it bumped with the cuboid. When subjected to a push by a 5g object, the robot would oscillate more for a few seconds but would later return to its original movement as shown in the graph below:



Graph 6.3 Tilt Angle Shift After Disturbance

It should be noted that since the weight of the frame is 0.25 kg, heavier weights can push the robot down with stronger forces.

II. CONCLUSION:

The self-balancing robot is a robot that can be used for many applications such as for cargo deliveries where the area provided by the application is limited. The controller for controlling such a robot which is the PID controller for this project is used in many real-life applications such as when adjusting the altitude of a rocket that has launched off the earth's surface. In conclusion, the project was successful as a robot was made that could balance itself and could withstand a certain degree of external disturbances.

CHAPTER 7

RECOMMENDATIONS

Recommendations on this project from the proponent would be the following:

1. Always seek alternative solutions if the current solution is not working out to save time.

2. Use mathematical modelling to build and design the robot. The purpose of this is to maintain the system's response even if there are changes in the dimensions of the robot.

3. Familiarize yourself and implement other control methods including the LQR method for getting better stability from the self-balancing robot.

4. Research well on the different control theories.

5. Pay attention to the factors that may contribute to a certain instability in your system.

6. Always stay in touch with someone who may have more knowledge than you so that you can grow your knowledge.

7. Keep testing and be prepared to face sudden challenges with patience.

REFERENCES

(1) 2 Wheel Self Balancing Robot Car Kit. (n.d.). Retrieved October 28, 2020, from https://www.auselectronicsdirect.com.au/2-wheel-self-balancing-robot-car-kit

(2) Wenxia, S., & Wei, C. (2017). Simulation and debugging of LQR control for two-wheeled self-balanced robot. *2017 Chinese Automation Congress (CAC)*. doi:10.1109/CAC.2017.8243176

(3) Ruan, X., & Cai, J. (2009). Fuzzy Backstepping Controllers for Two-Wheeled Self-Balancing Robot. *2009 International Asia Conference on Informatics in Control, Automation and Robotics*. doi:10.1109/car.2009.42

(4) Brokking, J. (2017). Your Arduino Balancing Robot (YABR). Retrieved October 22, 2020, from http://www.brokking.net/yabr_main.html

(5) Nasir, A. N., Ahmad, M. A., Ghazali, R., & Pakheri, N. S. (2010). The Control of a Highly Nonlinear Two-wheels Balancing Robot: A Comparative Assessment between LQR and PID-PID Control Schemes. *2011 First International Conference on Informatics and Computational Intelligence, 4*(10).

(6) Abreu, V. V. (November, 2009). *BALANCE-BOT* (Unpublished master's thesis). Universidade da Madeira.

(7) Al., A. A. (2015). Design Techniques Study of PID Controllers. *International Journal of Computing and Digital Systems, 4*(2), 101-109. doi:10.12785/ijcds/040204

(8) E. (2019). PROVOX™ to DeltaV™ Control Configuration Transition Service. Retrieved October 20, 2020, from https://www.emerson.com/documents/automation/configuration-data-sheet-provox-to-deltav-control-configuration-transition-service-pss-en-67874.pdf

(9) H. M. Omar, A. M. Elalawy and H. H. Ammar, "Two-wheeled Self-balancing robot Modeling and Control using Artificial Neural Networks (ANN)," *2019 Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, Giza, Egypt, 2019, pp. 196-200, doi: 10.1109/NILES.2019.8909311

(10) Ahmed, O., El-Amin, E., Salem, A., & Mohammed, A. (October, 2017). *DESIGN AND IMPLEMENTATION OF TWO WHEELED SELF BALANCING ROBOT USING PID CONTROLLER* . Sudan University of Science and Technology College of Engineering School of Electrical and Nuclear Engineering.

(11) Celik, Y., & Güneş, M. (2018). Designing an Object Tracker Self-Balancing Robot. *Academic Platform Journal of Engineering and Science, 6*(2), 124-133. doi:10.21541/apjes.414715.

(12) CoppeliaSim User Manual. (n.d.). Retrieved October 29, 2020, from https://www.coppeliarobotics.com/helpFiles/en/welcome.htm

(13) Terabee, F., LP, F., Motion, F., & Corp, F. (2020, October 26). Coppelia Robotics GmbH. Retrieved October 29, 2020, from https://www.azorobotics.com/Suppliers.aspx?SupplierID=1299

(14) Luck, K. S., Pajarinen, J., Berger, E., Kyrki, V., & Amor, H. B. (n.d.). Sparse Latent Space Policy Search. Retrieved October 28, 2020, from https://interactive-robotics.engineering.asu.edu/project/sparse-latent-space-policy-search/

(15) A. (Director). (2019, December 7). *Equations of Motion for the Inverted Pendulum (2DOF) Using Lagrange's Equations* [Video file]. Retrieved from https://www.youtube.com/watch?v=Fo7kuUAHj3s&feature=youtu.be

(16) Lagrange's Equations. (2003). Retrieved October 29, 2020, from https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-61-aerospace-dynamics-spring-2003/lecture-notes/lecture7.pdf

(17) A000073 - Buy Arduino Uno Rev3 SMD - Arduino - Distrelec. (n.d.). Retrieved October 29, 2020, from https://www.distrelec.biz/en/arduino-uno-rev3-smd-arduino-a000073/p/30101956

(18) Hc Sr04 Ultrasonic Sensor Module. (n.d.). Retrieved October 29, 2020, from https://www.indiamart.com/proddetail/hc-sr04-ultrasonic-sensor-module-19727491430.html

(19) Desktop Computers & All-In-One Computers – Inspiron and XPS: Dell Malaysia. (n.d.). Retrieved October 29, 2020, from https://www.dell.com/en-my/shop/desktop-computers/sc/desktops

(20) Smraza 5pcs Ultrasonic Module HC-SR04 Distance Sensor with 2pcs Mounting Bracket for Arduino R3 MEGA Mega2560 Duemilanove Nano Robot XBee ZigBee. (n.d.). Retrieved October 31, 2020, from https://www.amazon.sa/-/en/Smraza-Ultrasonic-Distance-Mounting-Duemilanove/dp/B01JG09DCK/ref=asc_df_B01JG09DCK/?tag=sashogostdde-21

(21) GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue. (n.d.). Retrieved October 31, 2020, from https://www.dx.com/p/gy-521-mpu6050-3-axis-acceleration-gyroscope-6dof-module-blue-2018592.html

(22) Geeetech 1.8-Degree Nema 14, 35 BYGHW Stepper Motor for 3D Printer. (n.d.). Retrieved October 31, 2020, from https://www.dx.com/p/geeetech-1-8-degree-nema-14-35-byghw-stepper-motor-for-3d-printer-black-2053251.html

(23) Arduino Uno Rev3. (n.d.). Retrieved October 31, 2020, from https://store.arduino.cc/usa/arduino-uno-rev3

(24) E-flite 11.1V 2200mAh 3S 30C LiPo Battery: EC3, EFLB22003S30.(n.d.). Retrieved October 31, 2020, from https://www.amazon.com/flite-11-1V-2200mAh-LiPo-13AWG/dp/B004PO4HF6#ace-g985962970.

(25) Midhun_s, & Instructables. (2017, September 19). Arduino Self-Balancing Robot. Retrieved November 12, 2020, from https://www.instructables.com/Arduino-Self-Balancing-Robot-1/

APPENDIX

CERTIFICATE FROM THE GRAMMARIAN

02nd December 2020

Certification - To Whom It May Concern

This is to certify that the "**Self-Balancing Robot"** research paper reviewed and edited, as requested, only for grammatical and spelling (punctuation and capitalization) accuracy.

This certification is issued at the request of the under mentioned student for whatever purpose it may serve them.

1. Faizaan Mohammed Mustafa BH16500026

Iconics Computers and Information Systems

PO Box 51063, Manama, Bahrain | Email: muddassar@iconics-systems.com | Tel: +973 34121374 | CR No: 114405

Self-Balancing Robot

by Muddassar

General metrics

54,865	8,202	651	32 min 48 sec	1 hr 3 min
characters	words	sentences	reading time	speaking time

Score

84

Writing Issues

48
Critical

Writing Issues

48	**Correctness**	
16	Mixed dialects of english	
23	Misspelled words	
3	Faulty subject-verb agreement	
1	Closing punctuation	
1	Determiner use (a/an/the/this, etc.)	
3	Unknown words	
1	Improper formatting	

1 **Clarity**

1 Wordy sentences

Unique Words

Measures vocabulary diversity by calculating the percentage of words used only once in your document

17%

unique words

Rare Words

Measures depth of vocabulary by identifying words that are not among the 5,000 most common English words.

42%

rare words

Word Length

Measures average word length

4.7

characters per word

Sentence Length

Measures average sentence length

12.6

words per sentence

Self-Balancing Robot

by Muddassar

General metrics

54,774	8,197	650	32 min 47 sec	1 hr 3 min
characters	words	sentences	reading time	speaking time

Score

84

Writing Issues

Critical

Unique Words

Measures vocabulary diversity by calculating the percentage of words used only once in your document

17%

unique words

Rare Words

Measures depth of vocabulary by identifying words that are not among the 5,000 most common English words.

42%

rare words

Word Length Measures average word length	**4.7** characters per word
Sentence Length Measures average sentence length	**12.6** words per sentence

CHAPTER 2

BACKGROUND OF STUDY

A Self-balancing robot has a fundamental characteristic of solving the inverted pendulum problem by understanding the angle at which it is tilting forward and backwards. The inverted pendulum problem involves an inverted pendulum that needs to be balanced with its tilt angle being at 0 degrees. The problem involves correcting the angle of tilt with a suitable closed-loop control architecture that may assist in achieving a balanced system. This project involves exploring the different solutions to the inverted pendulum problem and finding the most suitable one for building a self-balancing robot.

2.1 OBJECTIVES OF THE STUDY:

1. To design a suitable control architecture for solving the inverted pendulum problem.
2. To build a robot that uses the designed control architecture to balance itself with the use of sensors such as a gyroscope and accelerometer.
3. Constraints: For this project, a small and compact robot will be built.

2.2 SIGNIFICANCE OF THE DESIGN PROJECT:

The significance of this project relates to how much the inverted-pendulum problem relates to many worldly applications. These applications include rocket launch applications, where the rocket has to balance its trajectory and keep itself focused on the path to its destination. Another application could be that of a commercial aeroplane

which needs to keep itself steady on its path while withstanding the air pressure of the wind that comes from the front towards the aeroplane. Such applications draw the importance of why designing a stable control architecture is essential for operation in our daily activities. By developing a control architecture, and also a self-balancing robot; the student will learn how to solve complex problems such as the inverted pendulum problem.

2.3 SCOPE AND DELIMITATION:

2.3.1 SCOPE:

This project aims to build a self-balancing robot with the implementation of a control architecture for solving the inverted pendulum problem where the mass of the robot is like an inverted pendulum that needs to be balanced on a surface without falling. The self-balancing robot's mobility and portability make it a very useful object for applications where space is limited. Applications such as delivering light cargo or robots that move around and communicate with other people for giving them directions are a few of the many applications where self-balancing robots can play a significant role.

2.3.2 DELIMITATION:

While the project involves building a self-balancing robot, here are a few limitations that are associated with this project. The first limitation is that in this project, only the self-balancing robot is built to balance the mass of the robot. The second limitation is that we cannot add a lot of weight to the robot as that can make the system unstable. Lastly, the robot will be tested on a flat surface and not on surfaces that are

uneven as the latter would cause the robot to struggle in achieving a stable and balanced form.

2.4 DEFINITION OF TERMS:

1. **Gyroscope-** A gyroscope is a sensor that measures the angular accelerations of an object that rotates in the x, y, and z directions. It will be used to find the tilt angle of the robot.
2. **Accelerometer-** An accelerometer is a device that measures linear accelerations in the x, y, and z directions. It will be used to find the tilt angle of the robot.
3. **DC Motor-** A DC motor is a motor that converts electrical energy to mechanical energy. The motor is commonly used in many appliances such as electric vehicles, elevators, hoists, and more.
4. **Stepper Motor-** This motor rotates in steps as a signal is sent to it from the motor controller. It has high holding torque and is used in many devices such as telescopes, hard drives, antennas, and more.
5. **PID Controller**- The PID controller nicked the Proportionality, Integral, and Derivative controllers that can be used for error correction. In this project, the PID controller is utilized for correcting the error in the tilt angle of the robot.
6. **Motor Controller-** The motor controller is used for sending pulses to move the motor and also to control the direction in which the wheel's shaft is rotating. The use of a motor controller will be essential in controlling the movement of the motors connected to the wheels.

7. **Center of Mass-** The centre of mass is the point that is at the centre of the distributed mass. It is the point where if a force is applied, it would cause a linear acceleration in a straight path without any angular acceleration.

CHAPTER 3

REVIEW OF LITERATURE

This chapter discusses the literature that is published on developing the self-balancing robot in different ways.

3.1 CONCEPTUAL LITERATURE:

The research done was mostly related to analyzing how the inverted pendulum problem can be solved and how other researchers attempted their ideas in getting the robot to balance itself. The literature is mostly related to the different controllers that have been used for calculating the tilt angle of the robot. This is essential as the angle of tilt dictates the direction which the motors need to move towards to balance the robot. The efficiency of all the controllers that had been witnessed was compared and analyzed to find the most suitable controller for the current project.

3.2 RESEARCH LITERATURE:

- **FOREIGN STUDIES:**

(1) Sun Wenxia and Chen Wei from Qingdao University of Science & Technology had published their paper titled "Simulation and debugging of LQR control for two-wheeled self-balanced robot" in the IEEE Xplore library where they used the Newton Euler method for linearizing the inverted pendulum problem by building a model that included the mass of the robot and the mass of the wheels, the torque of the wheel, acceleration, and the angular accelerations of the motors(2). It can be noted from this research that the LQR(Linear Quadratic Regulator) controller was able to provide good results as the tilt

angle of the robot returned to 0 degrees in almost 1.6 seconds. This shows that the LQR controller is another controller that can be used in controlling the two-wheeled robot.

(2) Xiaogang Ruan, Jianxian Cai, and Jing Cheno from the Beijing University of Technology, Institute of Artificial Intelligence and Robotics published a paper titled "Learning to Control Two-Wheeled Self-Balancing Robot Using Reinforcement Learning Rules and Fuzzy Neural Networks" in the IEEE Xplore library where they developed a novel approach to balance the two-wheeled by using Reinforcement learning(RL) and Fuzzy Neural Networks(FNN)(3). The concept of Q-learning, where a system learns from its environment and develops itself to earn rewards by correcting itself is also implemented by them as they try to solve the inverted-pendulum problem. They compared their FNN and RL models with the classic PID model and discovered that the FNN and RL model was able to stabilize the robot efficiently with fewer oscillations when compared to the PID controller.

(3) A Youtuber named Joop Brokking developed a PID based self-balancing robot that had a complementary filter included for more accurate measurements of the tilt angle(4). His model consisted of a robot that would start balancing itself as the robot is lifted to a certain tilt angle. The robot used stepper motors which kept the robot fixed at a specific position when the motors weren't moving. The system was very stable and was also successful in solving the inverted pendulum problem.

(4) A. N. K. Nasir, M. A. Ahmad, and R. M. T. Raja Ismail are authors of a paper titled "The Control of a Highly Nonlinear Two-wheels Balancing Robot: A Comparative

Assessment between LQR and PID-PID Control Schemes" published in the World Academy of Science, Engineering and Technology Journals (Vol:4, No:10) in 2010 where they reported their development of LQR and PID controllers and measured the efficiency of both controllers(5). They presented the control structure of both controllers and found out that both controllers were capable enough to balance the robot but the LQR was slightly better in balancing the nonlinear system of the self-balancing robot.

(5) Victor Vicente Abreu from the University of Madrid developed a balancing robot that was modelled in the MATLAB software using the LQR controller(6). The robot used a PID controller with the inclusion of a Kalman filter for more accurate measurements of the tilt angle. From their tests, they noted as they added more mass to the top-centre position, the robot became more unstable and struggled in balancing itself as the centre of mass had moved away from the vertical axis.

- **LOCAL STUDIES:**

(6) Amin Alqudah and Asseel Qasaimeh from Yarmouk University, Irbid, Jordan reported a journal where they discussed how they enhanced and tested the PID controller(7). They combined fuzzy logic with the controller to enhance the output as the fuzzy logic allowed control over the gain and time constant during different operation times.

(7) Emerson Arabia's DeltaV controllers are controllers that are used in oil and gas plants for plant operating procedures(8). The controllers make use of self-tuning, neural

networks, and fuzzy logic with the inclusion of PID controllers for efficient control of parameters such as gas level, or the amount of oil that is passed through a pipe. The controllers establish a plant control standard and also provide a library of control templates, designed for the plant to be used or reused.

(8) H. M. Omar, A. M. Elalawy, and H. H. Ammar published their paper titled "Two-wheeled Self-balancing robot Modeling and Control using Artificial Neural Networks (ANN)," in the IEEE Xplore platform where a mathematical model was developed and the state space was developed to model the plant of the system(9). Since the system of a Self-balancing is non-linear, the used Nonlinear Autoregressive Exogenous (NARX) Neural Network model with the use of recorded data architecture. They were able to conclude the NARX model was better than the mathematical model. For controlling the self-balancing robot, two controlling architectures including the PID Controller and the Model Reference Adaptive Control(MRAC) were considered. The two controllers were tested with the result being that both controllers are suitable for the control problem but MRAC gave the best response when random inputs were given with good disturbance rejection.

(9) Ahmed, O., El-Amin, E., Salem, A., and Mohammed, A. of the Sudan University of Science and Technology College of Engineering School of Electrical and Nuclear Engineering developed a mathematical model of a two-wheeled self-balance robot system using physical and electrical laws(10). After linearizing the model, they were able to simplify the model. By manually tuning the PID controller of their robot, they were able to determine the Kp, Ki, and Kd constants for the best system response. The best

response was found at: Kp= 40, Ki= 1 ,and Kd= 1. They observed high oscillation around the setpoint with the robot not settling down. After passing a certain angle, their system would become unstable with the robot not regaining balance. Yet, the results obtained from the manual tuning was acceptable and had met their expectation.

(10) Yunus Celik and Mahit Güneş from Karamanoğlu Mehmetbey University, Turkey developed a robot that can track colourful objects while balancing itself on two wheels(11). The robot was balanced using a PID controller with the optimal values for the controller found using the trial and error method. They used the Kalman filter to make the robot stable and less noisy and made use of a colour detection algorithm where a blue object was chosen as the object to track while the robot balanced itself.

3.3 SYNTHESIS:

From the research, it was learned that there are various methods of solving the inverted pendulum problem with the PID controller, the LQR controller, and methods that integrate or customize both PID and LQR controllers. The controller that I have chosen is a PID controller as I have used parts of the PID controller for my previous projects.

It was also noted that mass distribution is essential for the stability of the robot as it tries to balance itself. The best option for the mass distribution would be to place most of the weight near the wheels at the centre of the robot frame to achieve a vertical centre of gravity. This would reduce the chances of instability in motion as the robot tries to balance itself.

CHAPTER 4

DESIGN SPECIFICATIONS

4.1 RESEARCH PARADIGM:

Phase 1: Project Development Knowledge Requirements	**Phase 2: Assembly Part Assembly**	**Phase 3: Prototype Testing**
- PID Control theory. - Inverted Pendulum Problem. - Arduino Programming. - Coppelia Simulation Scene building and programming. - Mechanical design **Hardware Requirements:** - Arduino Microcontroller - Ultrasonic Sensor **Software Requirements:** - Coppelia Simulation - Arduino IDE	- Building robot frames within the Coppelia Scene environment. - Programming the ultrasonic sensor using an Arduino - Program the robot in the Coppelia simulation software.	- Test the self-balancing robot.

Figure 4.1. Research Paradigm in Phases.

Page 11

The project development phase requires the acquisition of fundamental knowledge and resources that can aid in the next phase of the project. An understanding of each requirement is essential for development as every component and resource are interdependent for the final result.

4.2 PROJECT DEVELOPMENT:

The project began with the idea of developing a self-balancing robot with a suitable control algorithm. The research began by understanding the inverted pendulum problem and how it can be solved with the use of different control theories. From personal experience, the PID controller has been well-versed to me and thus, I have chosen the PID based control architecture. Since I needed inputs on how the robot is balancing on its own, I chose to use a gyroscope and an accelerometer to measure the tilt angle of the robot. Initially, the plan was to develop an Arduino-based prototype of the self-balancing robot but the plan had to change due to the COVID-19 pandemic.

4.2.1 PROJECT ISSUES:

Due to the Coronavirus pandemic, a lot of challenges and risks were introduced, especially the risk of receiving the components late and not having access to an in-person consultation with my project advisor. Also, the closure of the university caused me to not access the university resources on previous thesis papers of students who had worked on a control problem such as the Inverted Pendulum Problem. The solution to these challenges and risks came with distant consultation with my project advisor and online research. My plan changed with the decision to build my robot virtually on a virtual platform known as Coppelia Sim. and interfacing that robot with a physical Arduino microcontroller for external sensory inputs.

Figure 4.2 Coppelia Sim(12).

Coppelia Robotics is a small Swiss SME active in the field of robotics simulation(13). This is a simulation software where a robot of any kind can be designed, programmed, and tested. Various pre-built robots are already present within the software such as line-following robots, humanoids, and more. In addition to the virtual capabilities that are present, this software can interact with external hardware such as Arduino Uno for receiving sensory inputs. Therefore, such capabilities allowed me to bring changes to my project by turning the focus towards the interaction between reality and virtual reality. Such interaction became possible by adding an external ultrasonic sensor that would measure the displacement of my hand to create a force that is applied to the virtual self-balancing robot within the Coppelia Sim. Software. By doing this, we can verify the robot's resistance to external impact as it tries to balance itself virtually. The Lua programming language has been chosen to program the robot.

4.2.2 MATHEMATICAL MODEL:

Before designing the actual robot, it was important to develop a mathematical model by deriving the equations of motion for the inverted pendulum model to understand the non-linear dynamics of this model. The diagram below shows an ideal mass with an inverted pendulum.

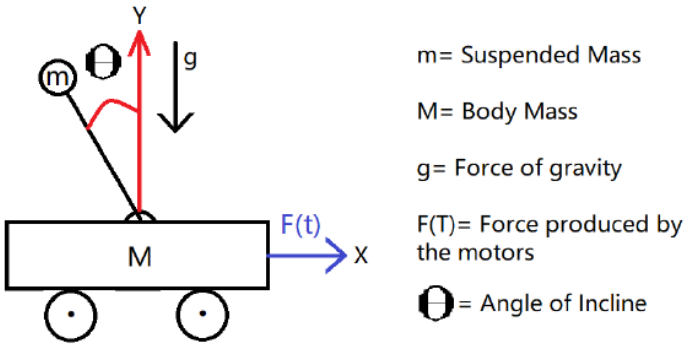


Figure 4.4 Inverted Pendulum Model

Initially, the model is assumed to be on a frictionless surface with a mass "M" that has a force "F" applied to it. The pendulum is made up of a mass "m " with a rigid massless rod attached to the same mass. The force of gravity 'g' is present and acts on the model. Using Lagrange's Equations, the equations of motion are derived for the inverted pendulum model(15). Lagrange's equations are equations that make the use of kinetic(T) and potential energy(V)for solving

equations of motion(16). We initially begin with the kinematics of the suspended mass 'm' on the pendulum across the x-axis and y-axis:

□□= X-□*sin θ **Eq. 4.1**

□□= □*cos θ **Eq. 4.2**

By taking the derivative of equation 4.1 and 4.2 from each side, we get:

$\dot{X}m$= $\dot{X}$-□*$\dot{\theta}$*cos θ **Eq. 4.3**

$\dot{Y}m$= -□*$\dot{\theta}$*sin θ **Eq. 4.4**

Then, we calculate the potential energy(V),

V= m*g*□□= m*g*□*cos θ **Eq. 4.5**

The kinetic energy(T) is then calculated,

T= ½*M$\dot{X}^2$+ ½ *m* ($\dot{X}m^2 + \dot{Y}m^2$)

We substitute values for Xm' and Y m',

T= ½*M$\dot{X}^2$ + ½ m(($\dot{X}$− □ *$\dot{\theta}$*cos θ)^2+ (−□ *$\dot{\theta}$*sin θ)^2)

T= ½*M$\dot{X}^2$+ ½ m(($\dot{X}^2$-2*$\dot{X}$*□*$\dot{\theta}$*cos θ + (□^{2}*$\dot{\theta}^2$*cos^2 θ) + (□^{2}*$\dot{\theta}^2$*sin^2 θ))

T= ½ (M+m)$\dot{X}^2$ + ½ m(-2*$\dot{X}$*L*$\dot{\theta}$*cos θ + (□^{2}*$\dot{\theta}^2$*cos^2 θ)+ (□^{2}*$\dot{\theta}^2$*sin^2 θ))

T= ½ (M+m)$\dot{X}^2$+ ½ m(-2*$\dot{X}$*□*$\dot{\theta}$*cos θ+□^{2}*$\dot{\theta}^2$(cos^2 θ+ sin^2 θ))

T= ½ (M+m)$\dot{X}^2$ + ½ m((-2*$\dot{X}$*□*$\dot{\theta}$*cos θ) +□^{2}*$\dot{\theta}^2$)

T= ½ (M+m)$\dot{X}^2$ + ½ m□□ $\dot{\theta}^2$ - (m$\dot{X}$□$\dot{\theta}$cos θ) **Eq. 4.6**

The following are the Lagrange's equations:

$$\frac{d}{dt}\left(\frac{dL}{d\dot{q}i}\right) - \frac{dL}{dqi} = Qi$$ **Eq. 4.7**

$$L = T - V$$ **Eq. 4.8**

By substituting values for T and V in equation 4.8, we get:

$L =$(½ **(M+m)**$\dot{X}^2$ + ½ **m**□□ $\dot{\theta}^2$ - **(** **(m**$\dot{X}$□$\dot{\theta}$**)***cos θ) - **(m*g*□***cos θ) **Eq. 4.9**

Therefore, to attain the first equation of motion; we take the derivative of equation 4.9 in terms of $\dot{X}$:

$\frac{dL}{d\dot{X}}$ = **(M+m)**$\dot{X}$- **(m** □$\dot{\theta}$**)***cos θ= M$\dot{X}$ + **m**$\dot{X}$- **(m** □$\dot{\theta}$**)***cos θ

Then, in terms of t:

$\frac{d}{dt}\left(\frac{dL}{d\dot{X}}\right)$ = **(M+m)**$\ddot{X}$ - **(** **(m** □ $\ddot{\theta}$ **)***cos θ- **(** **(m** □$\dot{\theta}^2$**)***sin θ)= F(t)

→ **F(t)= (M+m)**$\ddot{X}$ - **(m** □ $\ddot{\theta}$ **)***cos θ + **(** **(m** □$\dot{\theta}^2$**)***sin θ) **Eq. 4.10**

Therefore, an equation relating the force applied(F(t)) with the angle of tilt(θ) and the acceleration of the cart($\ddot{\theta}$).

To attain the second equation of motion; we take the derivative of equation 4.9 in terms of $\dot{\theta}$ and then, in terms of t:

$\frac{d}{dt}\left(\frac{dL}{d\dot{\theta}}\right)$ = **(m**□□ $\ddot{\theta}$ **)**- **(** **(m**$L\ddot{X}$ cos θ) + (**m**$L\dot{X}\dot{\theta}$ sin θ)

By subtracting the solution $\frac{dL}{d\theta}$, we get:

$\frac{d}{dt}\left(\frac{dL}{d\dot{\theta}}\right) - \frac{dL}{d\theta}$ = **m**□□ $\ddot{\theta}$ - **(** **m**$L\ddot{X}$ cos θ)+ (**m**$L\dot{X}\dot{\theta}$ sin θ) - (**m**$L\dot{X}\dot{\theta}$ sin θ) - (**m** L**g**sin θ)= 0 (where $\frac{dL}{d\theta}$ = (**m**$L\dot{X}\dot{\theta}$ sin θ) + (**m**L**g**sin θ)= 0).

$= \mathbf{m}\square^{\square} \ \ddot{\theta} - (\mathbf{m}L\ddot{X}\cos\theta) - (\mathbf{m}L\mathbf{g}\sin\theta) = 0$

By dividing **m**□ on **both sides, we get**

$$L\ddot{\theta} - (\ddot{X} * \cos\theta) - (\mathbf{g} * \sin\theta) = 0 \qquad \textbf{Eq. 4.11}$$

Finally, the second equation of motion is achieved.

4.2.3 HARDWARE:

The hardware needed for this project includes a microcontroller, a proximity sensor, and a desktop computer with sufficient memory for installing the Coppelia Simulation software.

Figure 4.5 Microcontroller(17) + Ultrasonic(18) + Desktop Computer(19).

4.2.4 PROTOTYPE DESIGN:

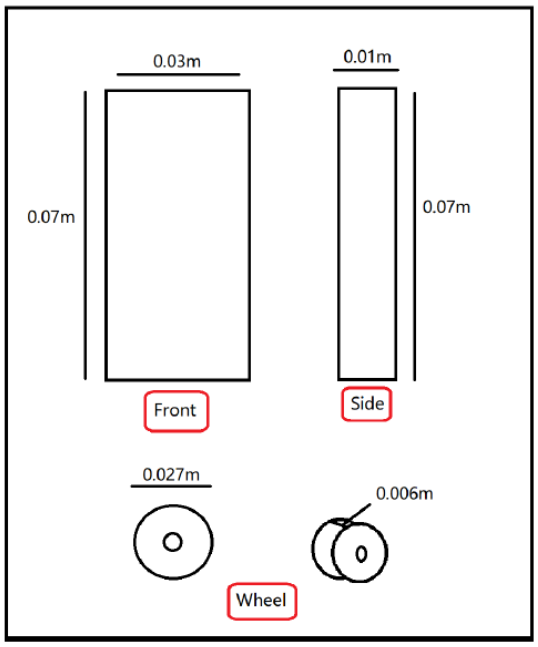


Figure 4.6 Prototype Design.

The prototype design of the self-balancing robot consists of a single frame that has two wheels connected to it. The dimensions of the frame and the wheels have been presented in the diagram above. The motors present in the Coppelia simulation software are attached.

4.2.5 SOFTWARE:

The software needed for this project includes the Coppelia Simulation software and the Arduino IDE.

4.2.6 MICROCONTROLLER:

The microcontroller that has been chosen is the Arduino Uno board due to its compatibility with Coppelia Sim and due to its cost.

4.3 DESIGN STANDARDS:

IEEE Std 100-1996: This paper provides the standard dictionary for electronic appellations. This standard has more than 24,255 terms and definitions associated with electronics. Proponents of this project used this documentation heavily when defining acronyms in the design, development, and documentation of the project at hand.
SAS 123-31 Programming Guidelines: This paper presents a group of programming guidelines and conventions that will be considered in developing code to make sure that it's clean, efficient, transferable, and maintainable.

4.4 MULTIPLE DESIGN CONSTRAINTS:

Design constraints related to cost, time, and flexibility will be presented comparing the options of having a complete physical robot and a virtual robot with hardware additions. The Figure of Merits analysis is used. Input values, scoring functions, and the weight are the three parameters that are considered for this type of analysis. The inputs include variables such as costs, time-span, and percentage values for the possibility of success.

The weights are assigned by the project designer and can range from 0 to 1. The scoring functions are used to calculate the score and are presented in equation 4.1 and equation 4.2.

The functions require three inputs including the input value, the upper limit of the input value, and the lower limit of the input value. A higher score is achieved from equation 4.1 when there is a high input value and the higher score is preferred for this parameter while a higher score is achieved from equation 4.2 when there is a lower input value. The higher score from the lower input value in equation 4.2 is more preferable than the higher score from the higher input in equation 4.1.

$$\textbf{Score} = \frac{\square\square\square\square\square\square - \square\square\square\square\square\square}{\square\square\square\square\square\square - \square\square\square\square\square\square}$$

Eq. 4.12 Higher input= Higher Score.

$$\textbf{Score} = \frac{\square\square\square\square\square\square - \square\square\square\square\square\square}{\square\square\square\square\square\square - \square\square\square\square\square\square}$$

Eq. 4.13 Lower input= Higher Score.

The following parameters have been utilized for the FoM criterion:

i. **Cost**: The constraints in costs will be presented in BHD.

ii. **Availability**: This parameter refers to the time it takes for the component to arrive from the manufacturer to the proponent's location.

ii. **Reliability**: This will be calculated by determining the failure percentage. The greater the failure rate, the lower the reliability for the device. The failure percentage is represented after combining the failure percentage of each component.

iii. **Safety:** A percentage is assigned to the level of safety each component has. This is assigned by taking into consideration if the component is RoHS compliant or not where there are no hazardous substances such as mercury, cadmium, and other such chemicals that are used in manufacturing electronic devices.

After completing the research on the components, the proponent has identified the following design plan as shown in Table 4.1 below. All the design plans fulfil the required criteria for prototyping the proponent's project.

Table 4.1 Design Plan

Component	**Plan A: Physical Robot**	**Plan B: Virtual Robot***

Hardware	Plywood Sheets- 2 weeks (6 BD)	Virtual Robotic Components (Free)-0
Ultrasonic Sensor	'HC-SRO4' (20)- 2 weeks (5 BD)	HC-SRO4 (5BD)-2
Gyroscope	'GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue' (21) (2 BD)-1 week	Gyro Sensor (Free)-0
Accelerometer	'GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue' (21) (2 BD)-1 week	Accelerometer (Free)-0
Motor	'Geeetech 1.8-Degree Nema 14, 35 BYGHW Stepper Motor for 3D Printer' (22) (11 BD)- 2 weeks	In-built DC Motor (Free)-0
Microcontroller	Arduino Uno (23) (4 BD)-4 weeks	Arduino Uno (4 BD)-4
Power Supply	'E-flite 11.1V 2200mAh 3S 30C LiPo Battery: EC3, EFLB22003S30' (24) (9 BD)- 2 weeks	Computer Power Supply for the robot & 220V external power supply for the Arduino. (Free)-0

*The virtual robot will be moved physically with the use of ultrasonic sensors.

The following parameters were chosen based on the given reasons:

Table 4.2 FoM Parameter Ranges

Parameter	Value	Reasoning
Cost Upper Limit	**700 BD**	This value is roughly the cost of a high-end self-balancing system.
Cost Lower Limit	**30 BD**	The cheapest self-balancing system is priced around this value.
Availability Upper Limit	**4 Weeks**	If no Express shipping is chosen, the shipping takes around 4 weeks.
Availability Lower Limit	**1.5 Weeks**	If Express shipping is chosen, the shipping takes around 1.5 weeks.
Accuracy Upper Limit	**99%**	To achieve 100% accuracy is impossible in the real world yet 99% accuracy is justifiable.
Accuracy Lower Limit	**35%**	Lower than or equal to 35% would certainly lead to failure once the product is in use.
Safety Upper Limit	**96%**	A safety of 100% is not always certain, therefore a percentage of 96 is chosen.
Safety Lower Limit	**52%**	When dealing with safety, the percentage needs to be above 52%. If the percentage goes below this, the prototype can be safe half the time which is not acceptable.

Table 4.3 FoM Plan A

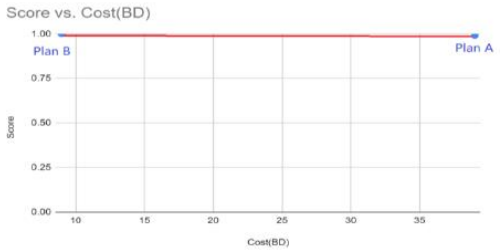
	Plan A

FoM	Input Value	Score Function	Weight	Score*Weight
Cost(BD)	39	0.99	0.25	0.248
Availability(weeks)	3	0.6	0.25	0.15
Accuracy(%)	75	0.63	0.25	0.16
Safety(%)	85	0.75	0.25	0.19
Total			1	0.8

Table 4.4 FoM Plan B

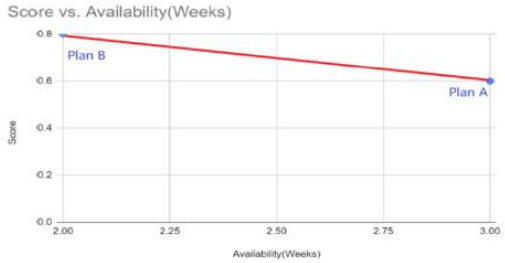
FoM	Plan B			
	Input Value	Score Function	Weight	Score*Weight
Cost(BD)	9	1	0.25	0.25
Availability(weeks)	2	0.8	0.25	0.2
Accuracy(%)	90	0.86	0.25	0.215
Safety(%)	95	0.98	0.25	0.245
Total			1	0.91

Plan B was chosen for having the best parameter values when compared to Plan A. The score for the cost was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.002 points:



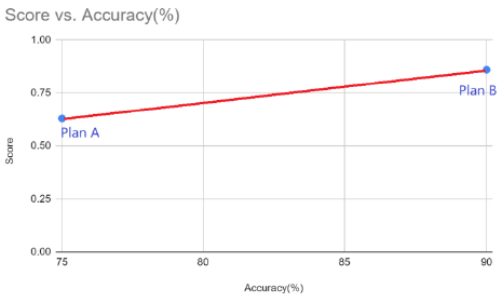
Graph 4.1 Scatter Plot for Score Vs. Cost(BD)

The score for the availability was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.002 points:



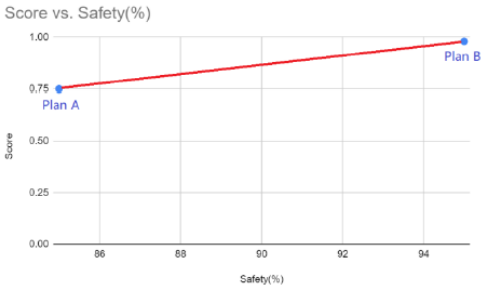
Graph 4.2 Scatter Plot for Score Vs. Availability(Weeks)

The score for the accuracy was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.05 points:



Graph 4.3 Scatter Plot for Score Vs. Accuracy(%)

The score for the safety was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.055 points:



Graph 4.4 Scatter Plot for Score Vs. Safety(%)

In conclusion, the FoM for both plans with consideration of the parameters including cost availability, accuracy, and safety shows that Plan B is a suitable option to go forward with as it had a higher overall score when compared to Plan A($0.91 > 0.8$). This option is a healthy alternative especially considering the project issues that had occurred as mentioned previously.

4.5 PROJECT BLOCK DIAGRAM:

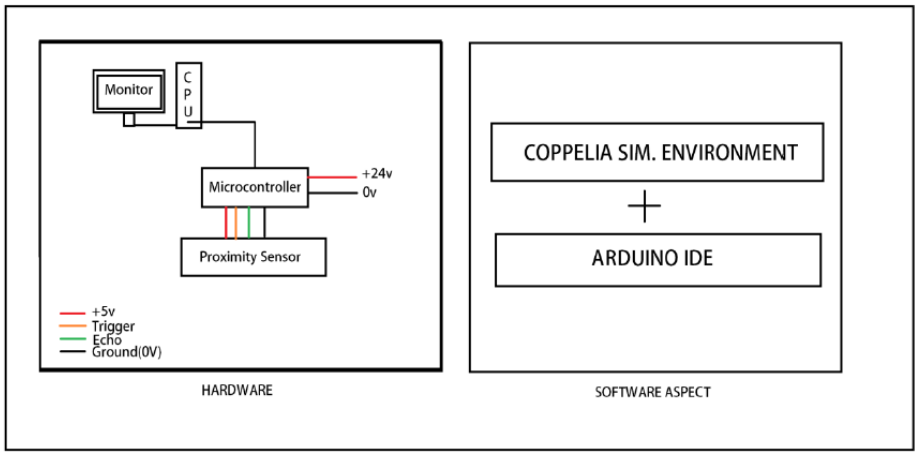


Figure 4.7 Project Block Diagram.

The project block diagram combines the software aspects and the hardware aspects of the robot as shown in the figure above.

4.6 PROJECT FLOWCHART:

A. Self-balancing Robot Flowchart:

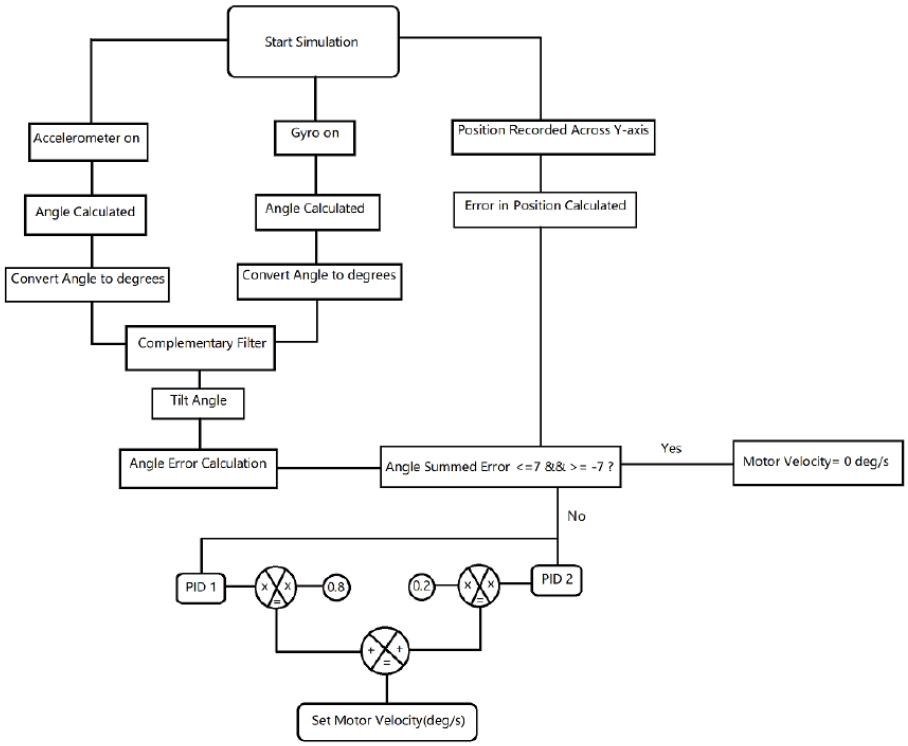


Figure 4.8 Flowchart 1.

B. Ultrasonic Sensor Sub-Routine Flowchart:

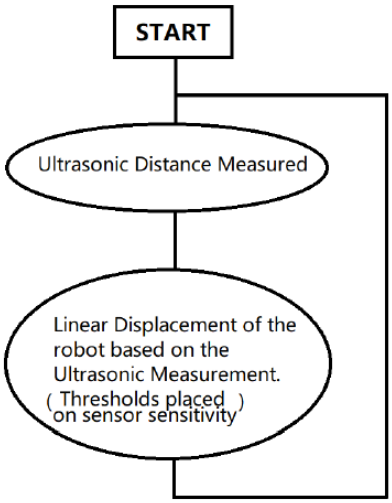


Figure 4.9 Flowchart 2.

4.7 CIRCUIT DIAGRAM:

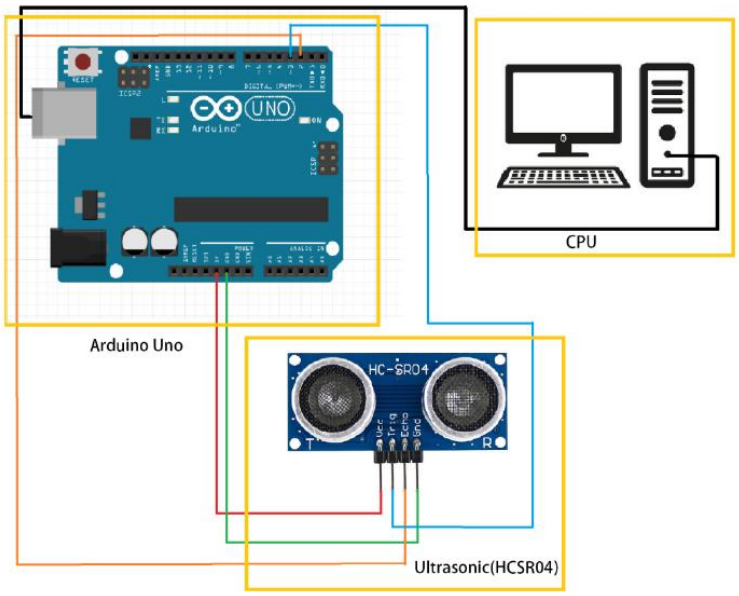


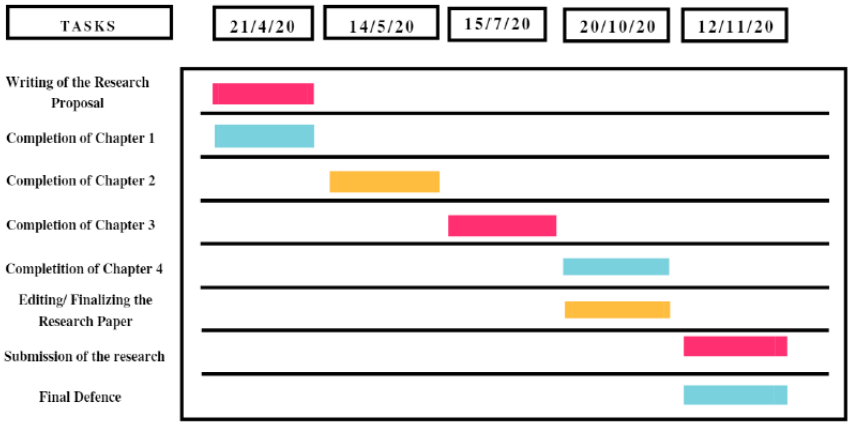
Figure 4.10 Circuit Diagram.

4.8 BILL OF MATERIALS:

Table 4.5 Bill of Materials

Item#	Description	Quantity	Total(BD)
1.	Arduino Uno	1	9
2.	USB 2.0 cable type A/B	1	1.5
3.	Ultrasonic Sensor	1	5
Total(BD)			**15.5**

4.9 GANTT CHART:



CHAPTER 5

DESIGN PROCEDURE, FUNCTIONAL ANALYSIS, AND IMPLEMENTATION

5.1 PROJECT DESCRIPTION:

To build and program the Self-balancing robot, specific hardware and software components are essential. For my Self-balancing robot, the Coppelia Simulation software is used for the creation and simulation of this robot. Within the Coppelia Simulation software, I will be using two motor-enabled rotary joints for movement, two wheels that will be attached to the joints, one Accelerometer sensor, a cuboid-shaped body for the chassis of the robot, and a dummy that is attached to the frame of the robot. The dummy and the accelerometer will be used to measure the angle of tilt of the robot. The Gyro sensor is not used due to the inaccuracy of the values found. The Arduino Uno and the Ultrasonic Sensor HC-SR04 will also not be used in creating external disturbances. Instead, a cuboid will be translated across the x-axis and the y-axis to create the external disturbances. The Lua programming language within the Coppelia Simulation project will be used to program the Self-balancing robot.

5.2 PROJECT FUNCTION ANALYSIS:

_____The Self-balancing robot begins functioning when the "Start/resume" simulation button is pressed within the Coppelia Simulation Project. After pressing this button, the simulator and the Accelerometer begin measuring the Euler angle around the x-axis and the linear acceleration(across the y and z-axis) respectively. These values are then used to

calculate the angle of tilt of the robot. Once the angle of tilt is determined, the error that occurs when that angle deviates from the set angle or the setpoint is found. The Setpoint for the robot is 0 degrees which are when the robot is perfectly balanced. Once the error is found, the error is summed with the previous error to find the total accumulated error(The summed error is initially zero when the simulation starts). After finding the summed error, this error is fed into a PID controller for achieving an output that is given to the motor enabled joints. A factor of 0.8 is multiplied with the calculated value.

The position of the frame of the robot is also recorded across the y-axis. The initial position of the robot is subtracted with the current position of the robot to get any error that may occur if the robot begins moving from its initial position while balancing itself. This error is then recorded and summed. The summed error is then fed into a second PID Controller and the output is multiplied by a factor of 0.2.

The outputs from both PID controllers are summed to find the appropriate target velocity for the motor joints in degrees. The motor enabled joints then move accordingly to balance the robot while trying to maintain the robot's position. Thus, a self-balancing robotic system is achieved.

While the Self-balancing robot tries to balance itself virtually, a light cuboid-shaped body is translated across the x and y-axis about the world frame to create external disturbances on the self-balancing robot.

5.3 PROJECT IMPLEMENTATION:

The project implementation procedure is as follows:

1. The Coppelia Simulation software is installed on a computer.
2. Within the Coppelia Simulation environment, the chassis of the robot is designed by shaping a cuboid in the dimensions mentioned within the Prototype Design Section in Chapter 4.
3. The wheels that have the rotary joints attached are connected to the chassis.
4. The Accelerometer sensor is attached to the centre of the chassis.
5. A 5g cuboid is placed near the self-balancing robot.

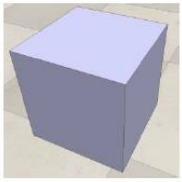


Figure 5.1 Cuboid For Shaping the Chassis

Figure 5.2 Motor Enabled Rotary Joint

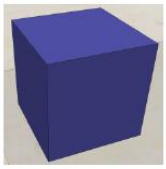


Figure 5.3 Accelerometer

5.4 EVALUATION PROCEDURES:

For the evaluation procedure, the Failure Mode And Effects Analysis(FMEA) are performed:

Table 5.1 Evaluation Procedure

Item #	Component Description	Failure Mode	Effects	Safeguard	Actions
1	Chassis	Disconnects with other parts when the movement starts	No movement occurs as the robot breaks down.	Ensure proper connections are made between the chassis and the remaining parts	Stop the robot Simulation and check all the connections made to the robot.
2	Dummy	Not giving proper Euler angle around the x-axis. Not giving accurate position values in the y-axis.	The tilt angle cannot be found using this component	Ensure that the dummy is placed at the centre of the chassis for accurate measurement.	Stop the robot Simulation and check the connection and position of the dummy in the scene hierarchy.
3	Accelerometer	Not giving acceleration values. Not giving accurate acceleration values	The tilt angle cannot be found from the accelerometer measurement	Check the connections/programming	Stop the robot Simulation and check the connections of the Accelerometer sensor. Also, check the Accelerometer sensor script and the Main program script within Coppelia Sim.
4	Rotary Motor Joints	Joints breaking	No movement occurs as the	Check the connections/pro	Stop the robot Simulation and check the connections

		when the movement starts. Wheels disconnect when the movement starts.	wheels of the robot disconnect.	gramming.	of the rotary joint with the wheels.
5	Computer System	No power	The complete shutdown of the project	Check the connections	Check the source of power and ensure all connections of the computer are properly made.

5.5 COMPONENT SPECIFICATIONS:

1. Coppelia Simulation:

The Coppelia Simulation software is a simulation software where robots can be built and programmed in many ways. The software offers different ways to program robots either locally using the Lua programming language or by using the remote API interface to program from software like MATLAB, Python, or Java or by using the ROS(Robot Operating System) communication method. For my program, I have used the local programming language Lua to program my Self-balancing robot. The Newton physics engine is used for the dynamics of the robot with very accurate simulations.

Within the software, a scene is created where the robot can be built and simulated. Within the Scene, a Scene Hierarchy is presented which includes all the components of

the robot. The components include the sensors(Accelerometer), the main robot frame, a dummy object for obtaining the orientation, and the robot motor joints. Also, within the Scene Hierarchy graphs that present the tilt angle and the error in the change in position respectively. Otherwise, there are default objects created within the software automatically that allow the creation of the floor, default lights, and the cameras for the robot. A side view camera has also been placed to get the side view of the self-balancing robot. The Scene hierarchy has been presented below

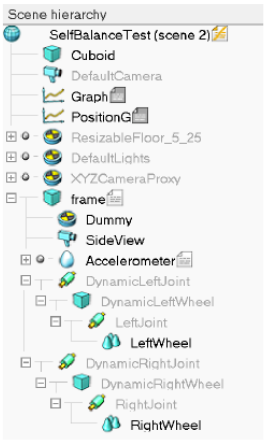


Figure 5.4 Scene Hierarchy.

The Robot that was built within the software has been presented below:

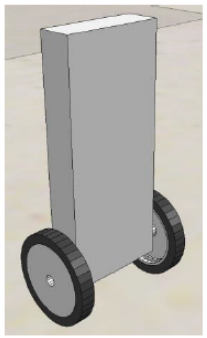


Figure 5.5 Main Self-balancing robot.

Table 5.2 Robot Specifications

Robot Specifications	
Weight of the Frame	0.25 kg
Dimensions(X*Y*Z) of the Frame	0.03m *0.01m*0.07m
Dimension(X*Y*Z) of the wheels	0.006m *0.027m*0.027m
The torque of Motor Joints	5 N*m

The weight of the frame of the robot is 0.25 kg with dimensions 0.03m on the x-axis, 0.01m on the y-axis, 0.07m on the z-axis. The wheels have a diameter of 0.027m across the y-axis and a depth of 0.006m approximately across the x-axis. The motor joints

attached to the wheel are To program the robot a child script needs to be created and attached to the frame of the robot as shown below:

Figure 5.6 Child Script.

There are two kinds of child scripts that can be created for the robot including a threaded child script and a non-threaded child script. For my program, I chose a non-threaded child script. Initially, within the non-threaded child script, the function sysCall_init is written. Handles for each object present in the scene are stored in variables using the *'sim.getObjectHandle()'*. A tube communication method is initialized to read the accelerometer values by writing the following code:

"accelCommunicationTube=sim.tubeOpen(0,'accelerometerData'..sim.getNameSuffix(nil),1)"

The sysCall_sensing() is then initialized where variables used in the program are declared. An if statement with the condition if '1==1' is created to form a continuous loop. Within the continuous loop, the program for measuring the tilt angle and for controlling the motors to correct the tilt angle is present. The loop time(dt) for the whole program is initialized at 10ms.

To find the angle of tilt of the robot, the Euler angle around the x-axis of the dummy placed inside the frame is measured by using the 'sim.getObjectOrientation' and the output from the accelerometer sensor is measured. This technique replaced the Gyro

sensor measurement method as the Gyro sensor was giving angular values that were significantly different when compared to the accelerometer angle values.

The Euler angle around the x-axis is received in radians and therefore, this value must be converted to degrees. To convert the value to degrees, the following equation is used:

$$Eangle = euler[1]*(180/math.pi) \qquad \textbf{Eq. 5.1}$$

Where euler[1] is the value of the orientation of the dummy present in the frame of the robot around the x-axis and where Eangle is the Euler angle in degrees.

To find the angle of tilt from the accelerometer sensor the following relationship is used:

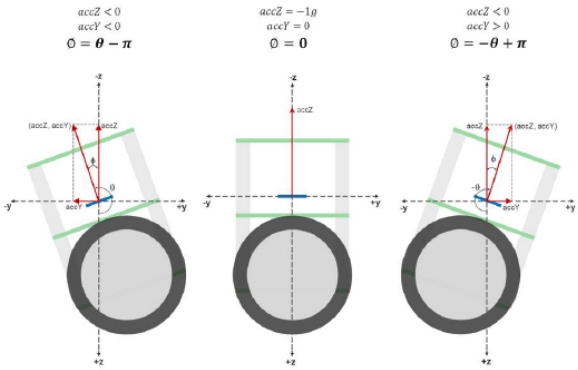


Figure 5.7 Accelerometer Sensor Tilt Angle Relations(25).

From the figure above, the Tilt Angle ϕ can be found by taking the inverse tangent of the linear acceleration across the y-axis and the z-axis. Also, the angle found needs to be converted to degrees. In the Self-Balancing robot program, the following equation was used to find the tilt angle and convert that angle to degrees:

accangle= (math.atan(accY/accZ))*(180/math.pi) **Eq. 5.2**

Where the accangle is the calculated angle from the output of the accelerometer, accY is the linear acceleration across the y-axis and accZ is the linear across the z-axis.

The angle of tilt is then found by using a complementary filter which combines the Euler angle and the accelerometer angle to get the best output by filtering the noise in the output. The complementary filter is used to find the tilt angle in the equation shown below:

TiltAngle= (0.9999999*Eangle)+(0.0000001*accangle) **Eq. 5.3**

Where Eangle is the measured Euler angle across the x-axis and accangle is the calculated angle from the output of the accelerometer.

After finding the tilt angle, the deviation of the title angle from the target angle is calculated to find the error by using the following equation:

error = targetAngle-TiltAngle **Eq. 5.54**

Where the target angle is 0 degrees. After finding the error, this error is summed with the previous error to find the value for the summed error or errorSumm as shown in the equation below:

errorSumm= errorSumm +error **Eq. 5.5**

Apart from measuring the tilt angle, the change in the position of the frame of the robot is also noted by measuring the position of the dummy. The purpose of measuring the position is to keep the robot around its initial position while the robot balances itself. This measurement is made using the "sim.getObjectPosition(dummy,-1)". The position across the y-axis is then found by entering the command " positionY= position[2]". The target position of the robot is its initial position at approximately 0.23m from the reference frame. The error in the position is then found by using the following equation:

errorP= targetPosition - positionY **Eq. 5.6**

After finding the error, this error is summed with the previous error to find the value for the summed error of the position or errorSummP as shown in the equation below:

errorSummP= errorSummP +errorP **Eq. 5.7**

The summation of the error in the tilt angle and the summation of the error in the position (across the y-axis) are then used to balance the robot to the target angle and keep the robot balancing around its initial position.

To achieve this, 2 PID controllers are used. The PID controller consists of the Proportionality, Integration, and Derivative controllers for correcting the tilt angle of the

robot and also for correcting the position of the robot. The following diagram illustrates this relationship:

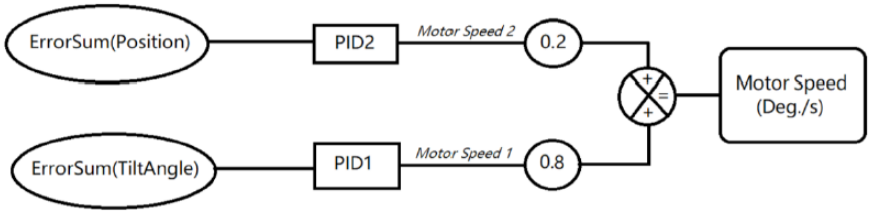


Figure 5.8 PID Controller Diagram.

For the first PID controller; the Kp, Ki, and Kd constants are 45.5, 35, and 0.0000045 respectively while for the second PID controller; the Kp, Ki, and Kd constants are 100, 0 ,and 0 respectively. The outputs from both PID controllers are combined to give the output to the motor joints. The motor speed has been limited from -500 to 500 degree second. The equation for both PID controllers is presented below:

motorPower =(0.8)*((Kp*(errorSumm))+(Ki*(errorSumm)*0.01)-Kd*((TiltAngle-prevAngleF)/0.01))+(0.2)*((Kp1*(errorSummP))+(Ki1*(errorSummP)*0.01)-Kd1*((positionY-prevP)/0.01)) **Eq 5.8**

Apart from the PID correction, a condition has been placed where if the angle of tilt is between -7 and 7 degrees; the velocity would be 0 degrees/s. For creating the

external disturbances, a 5-gram cuboid is made with the dimensions 0.01m*0.01m*0.08m is translated to push the self-balancing robot as shown below:

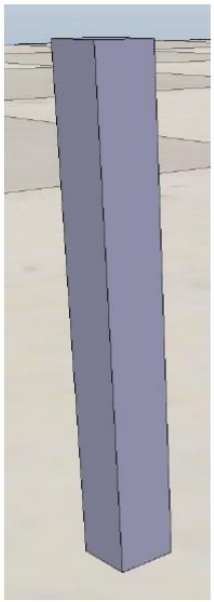


Figure 5.9 5g Cuboid

5.6 COST-BENEFIT ANALYSIS:

Cost-Benefit analysis is a procedure utilized when making business choices. The benefits or advantages of the business activity are summed, and the expenses related to that business activity is subtracted to achieve the Cost-Benefit analysis.

Table 5.3 Costs

		Fiscal Year					
Program Element	Element Manager	2020	2021	2022	2023	2024	2025
Material Purchasing	Faizaan Mustafa	$50	$75	$120	$320	$520	720
Equipment Purchasing	Faizaan Mustafa	$150	$200	$300			500
Product Installment	Faizaan Mustafa			$150	$150	$150	150
Product Distribution	Faizaan Mustafa			$100	$100	$100	100
Facilities	Faizaan Mustafa	$100	$100	$100	$100	$100	100
Service and Maintenance	Faizaan Mustafa			$80	$80	$80	80
Hardware Development	Faizaan Mustafa			$200	$200	$200	200
Software Develpoment	Faizaan Mustafa	$50	$50	$100	$100	$100	100
Program Total Costs By Year		$350	$425	$1,150	$1,050	$1,250	$1,950
Program Grand Total Cost	$6,175						

As mentioned in Table 5.3, the project sources are broken down into elements to find the total cost. The program source elements include Material Purchasing, Equipment Purchasing, Product Installment, Product Distribution, Facilities, Service and Maintenance, Hardware Development, and Software Development. The total cost of implementing this project is computed for the current year(2020) and the upcoming 5 years.

Table 5.4 Benefits

	Fiscal Year					
Benefit Sources	2020	2021	2022	2023	2024	2025
Cost Reduction			$150	$250	$500	$700
Enhanced Revenues			$3,000		$5,500	$12,500
Labor Reduction					$200	$250
Decreased Overhead				$100	$170	$170
Total Benefits Per Year	$0	$0	$3,150	$350	$6,370	$13,620
Confidence Factor	100%	100%	50%	80%	75%	80%
Benefits Claimed for Analysis	$0	$0	$1,575	$280	$4,778	$10,896
Program Grand Total Benefit	$17,529					

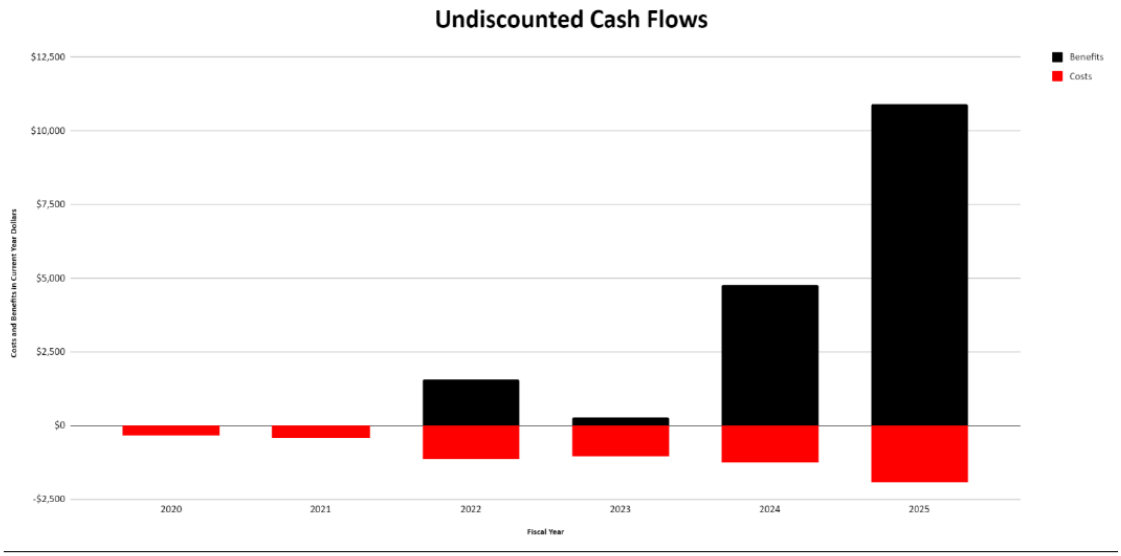
In table 5.4, the total benefit is computed for the project. There are four benefit sources including Cost Reduction, Enhanced Revenues, Labor Reduction, and Decreased Overhead. The benefit for each year of implementation is computed. Initially, the total benefit of each benefit source is summed and then, the total sum is multiplied by the confidence factor to get the benefits claimed for analysis. After finding the benefits claimed, the program's grand total benefit is found by adding the benefits of all the years.

As shown in the table above, no benefits are collected in the first two years as the collection of the benefits is expected to begin from the third year

Table 5.5 Summary

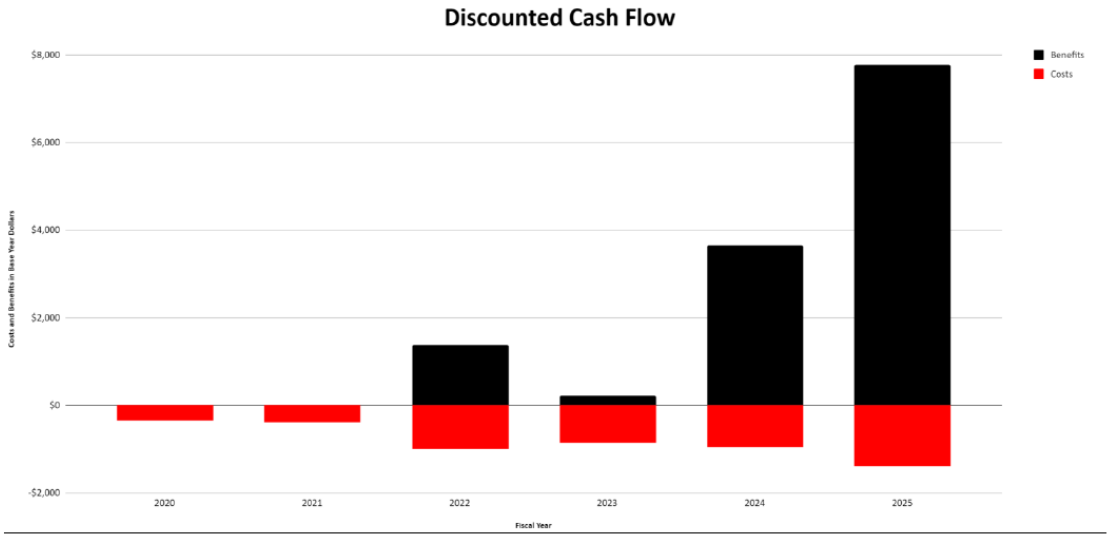
	Fiscal Year					
	2020	2021	2022	2023	2024	2025
Undiscounted Flows						
Costs	-$350	-$425	-$1,150	-$1,050	-$1,250	-$1,950
Benefits	$0	$0	$1,575	$280	$4,778	$10,896
Net Cash Flow	-$350	-$425	$425	-$770	$3,528	$8,946
Discount Factors						
Discount Rate	7.0%					
Base Year	2020					
Year Index	0	1	2	3	4	5
Discount Factor	1.0000	0.9346	0.8734	0.8163	0.7629	0.7130
Discounted Flows						
Costs	-$350	-$397	-$1,004	-$857	-$954	-$1,390
Benefits	$0	$0	$1,376	$229	$3,645	$7,769
Net	-$350	-$397	$371	-$629	$2,691	$6,378
Cumulative	-$350	-$747	-$376	-$1,005	$1,687	$8,065
Net Present Value	$8,065					
Internal Rate of Return	95%					

Table 5.5 presents a summary of the calculations for the undiscounted cash flow and the discounted cash flow. For the undiscounted cash flow; cost, benefits, net cash flow, discount rate, and base year are used to compute the net result. For the discounted cash flow; cost, benefits, net cash flow, cumulative, net present value, and the internal rate of return is used to compute the net result.



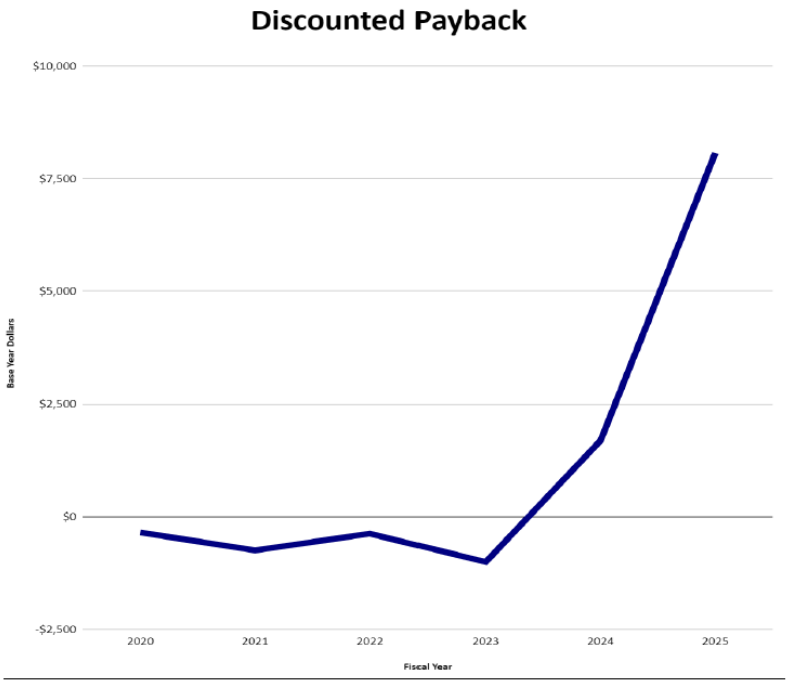
Graph 5.2 Undiscounted Cash Flows

In graph 5.2, the undiscounted cash flow graph is made using the values from table 5.4. The relationship between the cost and benefit is displayed among the several years of implementation of this project. From 2020 to 2021, only the cost is displayed while from 2022 to 2025; both cost and benefits are displayed. It can be noted that from 2024 to 2025, the benefit was constantly greater than the cost.



Graph 5.3 Discounted Cash Flows

In graph 5.3, the discounted cash flow graph is made using the values from table 5.4. The relationship between the cost and benefit is displayed among the several years of implementation of this project. From 2020 to 2021, only the cost is displayed while from 2022 to 2025; both cost and benefits are displayed. It can be noted that from 2024 to 2025, the benefit was constantly greater than the cost.



Graph 5.4 Discounted Payback

In graph 5.4, the discounted paycheck is made using the relationship between cost and benefit for the discounted cash flow. From 2020 to 2023, the cost is very high while the benefit is below zero dollars. From 2023 onwards, the trend of the curve changes with the increase of benefits in 2024 and 2025 as the cost is reduced from 2024 onward when compared to the benefits.

Interest: 0.07

Years: 10
Amount Borrowed: 5000

Table 5.6 Amortization Chart

Payment #	principal	interest	remaining
1	$28.89	$29.17	$4,971.11
2	$29.06	$29.00	$4,942.06
3	$29.23	$28.83	$4,912.83
4	$29.40	$28.66	$4,883.43
5	$29.57	$28.49	$4,853.87
6	$29.74	$28.31	$4,824.13
7	$29.91	$28.14	$4,794.21
8	$30.09	$27.97	$4,764.13
9	$30.26	$27.79	$4,733.86
10	$30.44	$27.61	$4,703.42
11	$30.62	$27.44	$4,672.80
12	$30.80	$27.26	$4,642.01
13	$30.98	$27.08	$4,611.03
14	$31.16	$26.90	$4,579.88
15	$31.34	$26.72	$4,548.54
16	$31.52	$26.53	$4,517.02
17	$31.70	$26.35	$4,485.31
18	$31.89	$26.16	$4,453.42
19	$32.08	$25.98	$4,421.35
20	$32.26	$25.79	$4,389.08
21	$32.45	$25.60	$4,356.63
22	$32.64	$25.41	$4,323.99
23	$32.83	$25.22	$4,291.16
24	$33.02	$25.03	$4,258.14
25	$33.22	$24.84	$4,224.92
26	$33.41	$24.65	$4,191.51
27	$33.60	$24.45	$4,157.91

113	$55.41	$2.64	$397.06
114	$55.74	$2.32	$341.32
115	$56.06	$1.99	$285.26
116	$56.39	$1.66	$228.87
117	$56.72	$1.34	$172.15
118	$57.05	$1.00	$115.10
119	$57.38	$0.67	$57.72
120	$57.72	$0.34	$0.00

The Amortization table displays the amount of money that needs to be paid in intervals to pay the total loan amount. The interest rates are also included in the table. The time allocated for paying the loan is 10 years for a loan amount of $5000. The remaining amount after all the payments are made is also included in the table.

CHAPTER 6

DESIGN PROJECT SUMMARY, AND CONCLUSIONS

The Self- balancing robot project was very special as it revolved around solving the inverted pendulum problem. The problem involved balancing an inverted pendulum by achieving a tilt angle of 0. There were so many different solutions that were witnessed during the research in the initial stages of the project. After the completion of the research, the PID Controller was eventually chosen by me due to my familiarity with this controller when compared to the other controllers that were used by others.

In software aspects, the Coppelia Simulation offered a great alternative for building the robot especially considering the current COVID-19 situation around the world. The software allowed me to build, compose, and program my robot without any difficulties. Sensors including the gyro sensor and the accelerometer were used to calculate the tilt angle but the former was not used to its inaccuracy in getting angular values. Also, the ultrasonic sensor was not used for creating the proposed external disturbance due to inaccurate transmission of ultrasonic sensor values within the simulation scene. The P (proportionality) controller in the PID also used the summation of the errors for finding the motor output which was theoretically incorrect as I learned later. I had not relied on a mathematical model that had made my process for finding the Kp, Ki, and Kd constants lengthy by trial and error based on the reaction of the self-balancing robot.

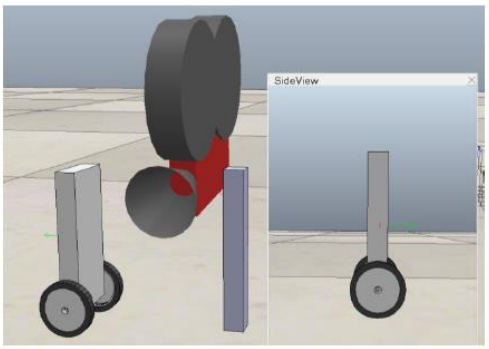
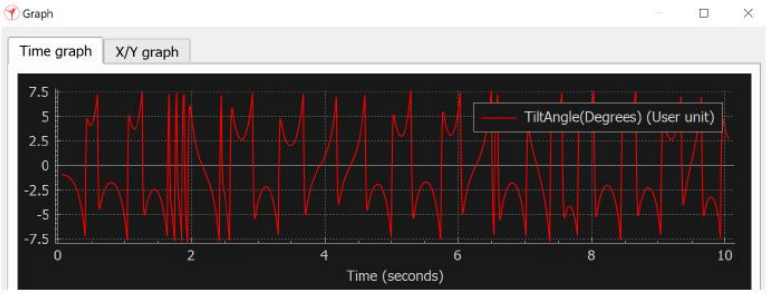


Figure 6.1 Self-balancing Robot Scene

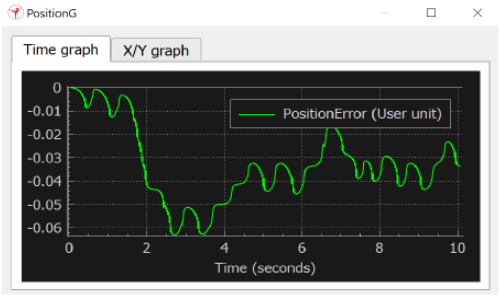
Yet, I was able to find solutions by incorporating Euler angles that were measured by the software using the command 'sim.getObjectOrientation'. For creating the disturbances, I was able to create cuboid-shaped objects within the scene that would bump with the self-balancing robot to create those disturbances.

When the robot would balance itself, the tilt angle would oscillate between -7.5 to 7.5 degrees as shown in the graph below:



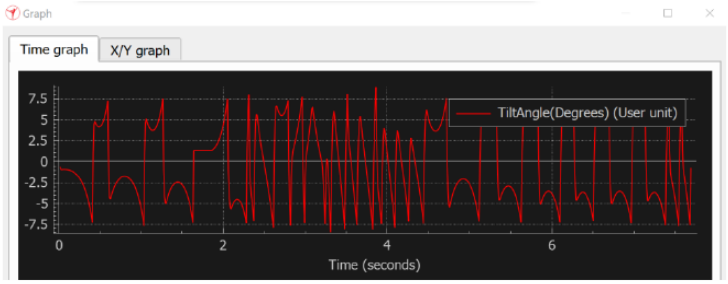
Graph 6.1 Title Angle Change(Time=10 seconds)

Such an output allowed the robot to balance itself without falling moving abruptly. The position of the robot also remained near the initial position as the degree error was almost 0 meter in the y-axis as shown in the graph below:



Graph 6.2 Position Error Change(Time=10 seconds)

The robot was balancing itself and was also able to withstand shock or disturbance that was created when it bumped with the cuboid. When subjected to a push by a 5g object, the robot would oscillate more for a few seconds but would later return to its original movement as shown in the graph below:



Graph 6.3 Tilt Angle Shift After Disturbance

It should be noted that since the weight of the frame is 0.25 kg, heavier weights can push the robot down with stronger forces. In conclusion, the project was successful as a robot was made that could balance itself and could withstand a certain degree of external disturbances.

CHAPTER 7

RECOMMENDATIONS

Recommendations on this project from the proponent would be the following:

1. Always seek alternative solutions if the current solution is not working out to save time.
2. Use mathematical modelling to build and design the robot. The purpose of this is to maintain the system's response even if there are changes in the dimensions of the robot.
3. Familiarize yourself and implement other control methods including the LQR method for getting better stability from the self-balancing robot.
4. Research well on the different control theories.
5. Pay attention to the factors that may contribute to a certain instability in your system.
6. Always stay in touch with someone who may have more knowledge than you so that you can grow your knowledge.
7. Keep testing and be prepared to face sudden challenges with patience.

REFERENCES

(1) 2 Wheel Self Balancing Robot Car Kit. (n.d.). Retrieved October 28, 2020, from https://www.auselectronicsdirect.com.au/2-wheel-self-balancing-robot-car-kit

(2) Wenxia, S., & Wei, C. (2017). Simulation and debugging of LQR control for two-wheeled self-balanced robot. *2017 Chinese Automation Congress (CAC)*. doi:10.1109/CAC.2017.8243176

(3) Ruan, X., & Cai, J. (2009). Fuzzy Backstepping Controllers for Two-Wheeled Self-Balancing Robot. *2009 International Asia Conference on Informatics in Control, Automation and Robotics*. doi:10.1109/car.2009.42

(4) Brokking, J. (2017). Your Arduino Balancing Robot (YABR). Retrieved October 22, 2020, from http://www.brokking.net/yabr_main.html

(5) Nasir, A. N., Ahmad, M. A., Ghazali, R., & Pakheri, N. S. (2010). The Control of a Highly Nonlinear Two-wheels Balancing Robot: A Comparative Assessment between LQR and PID-PID Control Schemes. *2011 First International Conference on Informatics and Computational Intelligence, 4*(10).

(6) Abreu, V. V. (November 2009). *BALANCE-BOT* (Unpublished master's thesis). Universidade da Madeira.

(7) Al., A. A. (2015). Design Techniques Study of PID Controllers. *International Journal of Computing and Digital Systems, 4*(2), 101-109. doi:10.12785/ijcds/040204

(8) E. (2019). PROVOX™ to DeltaV™ Control Configuration Transition Service. Retrieved October 20, 2020, from https://www.emerson.com/documents/automation/configuration-data-sheet-provox-to-deltav-control-configuration-transition-service-pss-en-67874.pdf

(9) H. M. Omar, A. M. Elalawy and H. H. Ammar, "Two-wheeled Self-balancing robot Modeling and Control using Artificial Neural Networks (ANN)," *2019 Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, Giza, Egypt, 2019, pp. 196-200, DOI: 10.1109/NILES.2019.8909311

(10) Ahmed, O., El-Amin, E., Salem, A., & Mohammed, A. (October 2017). *DESIGN AND IMPLEMENTATION OF TWO WHEELED SELF BALANCING ROBOT USING PID CONTROLLER*. Sudan University of Science and Technology College of Engineering School of Electrical and Nuclear Engineering.

(11) Celik, Y., & Güneş, M. (2018). Designing an Object Tracker Self-Balancing Robot. *Academic Platform Journal of Engineering and Science, 6*(2), 124-133. doi:10.21541/apjes.414715.

(12) CoppeliaSim User Manual. (n.d.). Retrieved October 29, 2020, from https://www.coppeliarobotics.com/helpFiles/en/welcome.htm

(13) Terabee, F., LP, F., Motion, F., & Corp, F. (2020, October 26). Coppelia Robotics GmbH. Retrieved October 29, 2020, from https://www.azorobotics.com/Suppliers.aspx?SupplierID=1299

(14) Luck, K. S., Pajarinen, J., Berger, E., Kyrki, V., & Amor, H. B. (n.d.). Sparse Latent Space Policy Search. Retrieved October 28, 2020, from https://interactive-robotics.engineering.asu.edu/project/sparse-latent-space-policy-search/

(15) A. (Director). (2019, December 7). *Equations of Motion for the Inverted Pendulum (2DOF) Using Lagrange's Equations* [Video file]. Retrieved from https://www.youtube.com/watch?v=Fo7kuUAHj3s&feature=youtu.be

(16) Lagrange's Equations. (2003). Retrieved October 29, 2020, from https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-61-aerospace-dynamics-spring-2003/lecture-notes/lecture7.pdf

(17) A000073 - Buy Arduino Uno Rev3 SMD - Arduino - Distrelec. (n.d.). Retrieved October 29, 2020, from https://www.distrelec.biz/en/arduino-uno-rev3-smd-arduino-a000073/p/30101956

(18) Hc Sr04 Ultrasonic Sensor Module. (n.d.). Retrieved October 29, 2020, from https://www.indiamart.com/proddetail/hc-sr04-ultrasonic-sensor-module-19727491430.html

(19) Desktop Computers & All-In-One Computers – Inspiron and XPS: Dell Malaysia. (n.d.). Retrieved October 29, 2020, from https://www.dell.com/en-my/shop/desktop-computers/sc/desktops

(20) Smraza 5pcs Ultrasonic Module HC-SR04 Distance Sensor with 2pcs Mounting Bracket for Arduino R3 MEGA Mega2560 Duemilanove Nano Robot XBee ZigBee. (n.d.). Retrieved October 31, 2020, from https://www.amazon.sa/-/en/Smraza-Ultrasonic-Distance-Mounting-Duemilanove/dp/B01JG09DCK/ref=asc_df_B01JG09DCK/?tag=sashogostdde-21

(21) GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue. (n.d.). Retrieved October 31, 2020, from https://www.dx.com/p/gy-521-mpu6050-3-axis-acceleration-gyroscope-6dof-module-blue-2018592.html

(22) Geeetech 1.8-Degree Nema 14, 35 BYGHW Stepper Motor for 3D Printer. (n.d.). Retrieved October 31, 2020, from https://www.dx.com/p/geeetech-1-8-degree-nema-14-35-byghw-stepper-motor-for-3d-printer-black-2053251.html

(23) Arduino Uno Rev3. (n.d.). Retrieved October 31, 2020, from https://store.arduino.cc/usa/arduino-uno-rev3

(24) E-flite 11.1V 2200mAh 3S 30C LiPo Battery: EC3, EFLB22003S30.(n.d.). Retrieved October 31, 2020, from https://www.amazon.com/flite-11-1V-2200mAh-LiPo-13AWG/dp/B004PO4HF6#ace-g985962970.

(25) Midhun_s, & Instructables. (2017, September 19). Arduino Self-Balancing Robot. Retrieved November 12, 2020, from https://www.instructables.com/Arduino-Self-Balancing-Robot-1/

PROGRAMMING

MAIN PROGRAM(CHILD SCRIPT OF “FRAME” OBJECT):

```
function sysCall_init()

dummy= sim.getObjectHandle("Dummy")
right= sim.getObjectHandle("DynamicRightJoint")
Positiongraph=sim.getObjectHandle('PositionG')
graph=sim.getObjectHandle('Graph')
left= sim.getObjectHandle("DynamicLeftJoint")

accelCommunicationTube=sim.tubeOpen(0,'accelerometerData'..sim.getNameSuffix(nil),
1)
end

function sysCall_sensing()

prevAngle= 0
currentFAngle= 0
previoustime= 0
error= 0
prevAngleF= 0
accangle= 0
errorSumm= 0
```

```
errorSummP=0
prevP=0
number=0
data1=0

if(1==1)then

local euler=sim.getObjectOrientation(dummy,-1)

local position=sim.getObjectPosition(dummy,-1)

Eangle= euler[1]*(180/math.pi) --Conversion from radians to deg.
positionY= position[2]

Kp=45.5
Ki=35
Kd=0.0000045

Kp1=100
Ki1=0
Kd1=0
```

```
-----------------------------------------------------------------------------------------------
data=sim.tubeRead(accelCommunicationTube)

if (data) then
acceleration=sim.unpackFloatTable(data)
accY= acceleration[2]
accZ= acceleration[3]

accangle= (math.atan(accY/accZ))*(180/math.pi) --- conversion from radians to degrees.

end

-----------------------------------------------------------------------------------------------
TiltAngle= 0

TiltAngle = (0.9999999*Eangle)+(0.0000001*accangle) --Complementary Filter

targetAngle=0

targetPosition=0.23228950798512
```

```
errorP= targetPosition - positionY
errorSummP= errorSummP +errorP

error = targetAngle-TiltAngle;
errorSumm= errorSumm +error

-----------------------------------------------------------------------------------------------

sim.setGraphUserData(graph, 'TiltAngle(Degrees)', TiltAngle)
sim.setGraphUserData(Positiongraph, 'PositionError', errorSummP)

if (errorSumm<=7) and (errorSumm>=-7)then--and (errorSummP<=0.2) and
(errorSummP>=-0.2) then
motorPower=0

sim.setJointTargetVelocity(right,0)
sim.setJointTargetVelocity(left,0)

prevAngleF= currentFAngle

else
motorPower =(0.8)*((Kp*(errorSumm))+(Ki*(errorSumm)*0.01)- Kd*((TiltAngle-
prevAngleF)/0.01))+ (0.2)*((Kp1*(errorSummP))+(Ki1*(errorSummP)*0.01)-
Kd1*((positionY-prevP)/0.01))
```

```
if (motorPower>= -500) and (motorPower<=500)then

sim.setJointTargetVelocity(right,-motorPower)
sim.setJointTargetVelocity(left,-motorPower)

prevAngleF= currentFAngle
prevP=positionY

end
end
end
end
```

SUPPORTING PROGRAM(ACCELEROMETER CHILD SCRIPT):

```
function sysCall_init()
    modelBase=sim.getObjectAssociatedWithScript(sim.handle_self)
    massObject=sim.getObjectHandle('Accelerometer_mass')
    sensor=sim.getObjectHandle('Accelerometer_forceSensor')
    result,mass=sim.getObjectFloatParameter(massObject,sim.shapefloatparam_mass)
    ui=simGetUIHandle('Accelerometer_UI')
    simSetUIButtonLabel(ui,0,sim.getObjectName(modelBase))
```

```
accelCommunicationTube=sim.tubeOpen(0,'accelerometerData'..sim.getNameSuffix(nil),
1)
end
-- Check the end of the script for some explanations!

function sysCall_cleanup()
end

function sysCall_sensing()
    result,force=sim.readForceSensor(sensor)
    if (result>0) then
        accel={force[1]/mass,force[2]/mass,force[3]/mass}
        sim.tubeWrite(accelCommunicationTube,sim.packFloatTable(accel))
        simSetUIButtonLabel(ui,3,string.format("X-Accel: %.4f",accel[1]))
        simSetUIButtonLabel(ui,4,string.format("Y-Accel: %.4f",accel[2]))
        simSetUIButtonLabel(ui,5,string.format("Z-Accel: %.4f",accel[3]))
    else
        simSetUIButtonLabel(ui,3,"X-Accel: -")
        simSetUIButtonLabel(ui,4,"Y-Accel: -")
        simSetUIButtonLabel(ui,5,"Z-Accel: -")
    end
    end
```

PLAGIARISM REPORTS

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	861	Date	December 01,2020
Characters	5483	Exclude URL	

0%	100%	0	39
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

CHAPTER 1
INTRODUCTION
The world today is surrounded by ideas and methods for solving problems within the fields of engineering, architecture, medicine, and more. One such problem is the classical inverted pendulum problem that involves balancing an inverted pendulum to keep the pendulum upright without falling. Many control theories have been developed to control the unstable system of an inverted pendulum. For the final thesis of the student, it is intended to create a self-balancing robot that can solve the inverted pendulum problem and that can balance itself on a surface without falling on the floor. With the use of motors and sensors, a robot can be designed and programmed to efficiently solve the inverted pendulum problem.

Figure 1.1 Self Balancing Robot Model(1)
CHAPTER 2
BACKGROUND OF STUDY
A Self-balancing robot has a fundamental characteristic of solving the inverted pendulum problem by understanding the angle at which it is tilting forward and backward. The inverted pendulum problem involves an inverted pendulum that needs to be balanced with its tilt angle being at 0 degrees. The problem involves correcting the angle of tilt with a suitable closed-loop control architecture that may assist in achieving a balanced system. This project involves exploring the different solutions to the inverted pendulum problem and finding the most suitable one for building a self-balancing robot.
2.1 OBJECTIVES OF THE STUDY:
To design a suitable control architecture for solving the inverted pendulum problem.
To build a robot that uses the designed control architecture to balance itself with the use of sensors such as a gyroscope and accelerometer.
Constraints: For this project, a small and compact robot will be built.
2.2 SIGNIFICANCE OF THE DESIGN PROJECT:
The significance of this project relates to how much the inverted-pendulum problem relates to many worldly applications. These applications include rocket launch applications where the rocket has to balance its trajectory and keep itself focused on the path to its destination. Another application could be that of a commercial airplane which needs to keep itself steady on its path while withstanding the air pressure of the wind that comes from the front towards the airplane. Such applications draw the importance of why designing a stable control architecture is essential for operation in our daily activities. By developing a control architecture, and also a self-balancing robot; the student will learn how to solve complex problems such as the inverted pendulum problem.
2.3 SCOPE AND DELIMITATION:
2.3.1 SCOPE:
This project aims to build a self-balancing robot with the implementation of a control architecture for solving the inverted pendulum problem where the mass of the robot is like an inverted pendulum that needs to be balanced on a surface without falling. The self-balancing robot's mobility and portability make it a very useful object for applications where space is limited. Applications such as delivering light cargo or robots that move around and communicate with other

people for giving them directions are a few of the many applications where self-balancing robots can play a significant role.

2.3.2 DELIMITATION:

While the project involves building a self-balancing robot and here are a few limitations that are associated with this project. The first limitation is that in this project, only the self-balancing robot is built to balance the mass of the robot. The second limitation is that we cannot add a lot of weight to the robot as that can make the system unstable. Lastly, the robot will be tested on a flat surface and not on surfaces that are uneven as the latter would cause the robot to struggle in achieving a stable and balanced form.

2.4 DEFINITION OF TERMS:

Gyroscope- A gyroscope is a sensor that measures the angular accelerations of an object that rotates in the x, y, and z directions. It will be used to find the tilt angle of the robot.

Accelerometer- An accelerometer is a device that measures linear accelerations in the x, y, and z directions. It will be used to find the tilt angle of the robot.

DC Motor- A DC motor is a motor that converts electrical energy to mechanical energy. The motor is commonly used in many appliances such as electric vehicles, elevators, hoists, and more.

Stepper Motor- This motor rotates in steps as a signal is sent to it from the motor controller. It has high holding torque and is used in many devices such as telescopes, hard drives, antennas, and more.

PID Controller- The PID controller nicked the Proportionality, Integral, and Derivative controllers that can be used for error correction. In this project, the PID controller is utilized for correcting the error in the tilt angle of the robot.

Motor Controller- the motor controller is used for sending pulses to move the motor and also to control the direction in which the wheel's shaft is rotating. The use of a motor controller will be essential in controlling the movement of the motors connected to the wheels.

Center of Mass- The center of mass is the point that is at the center of the distributed mass. It is the point where if a force is applied, it would cause a linear acceleration in a straight path without any angular acceleration.

Sources	Similarity

CHAPTER 3
REVIEW OF LITERATURE
This chapter discusses the literature that has been published on developing the self-balancing robot in different ways.
3.1 CONCEPTUAL LITERATURE:
The research that was done was mostly related to analyzing how the inverted pendulum problem can be solved and how other researchers have attempted their ideas in getting the robot to balance itself. The literature is mostly related to the different controllers that had been used for calculating the tilt angle of the robot. This is essential as the angle of tilt dictates the direction which the motors need to move towards to balance the robot. The efficiency of all the controllers that had been witnessed was compared and analyzed to find the most suitable controller for the current project.
3.2 RESEARCH LITERATURE:
FOREIGN STUDIES:
(1) Sun Wenxia and Chen Wei from Qingdao University of Science & Technology had published their paper titled "Simulation and debugging of LQR control for two-wheeled self-balanced robot" in the IEEE Xplore library where they used the Newton Euler method for linearizing the inverted pendulum problem by building a model that included the mass of the robot and the mass of the wheels, the torque of the wheel, acceleration, and the angular accelerations of the motors(2). It can be noted from this research that the LQR(Linear Quadratic Regulator) controller was able to provide good results as the tilt angle of the robot returned to 0 degrees in almost 1.6 seconds. This shows that the LQR controller is another controller that can be used in controlling the two-wheeled robot.
(2) Xiaogang Ruan, Jianxian Cai, and Jing Cheno from the Beijing University of Technology, Institute of Artificial Intelligence and Robotics published a paper titled "Learning to Control Two-Wheeled Self-Balancing Robot Using Reinforcement Learning Rules and Fuzzy Neural Networks" in the IEEE Xplore library where they developed a novel approach to balance the two-wheeled by using Reinforcement learning(RL) and Fuzzy Neural Networks(FNN)(3). The concept of Q-learning, where a system learns from its environment and develops itself to earn rewards by correcting itself is also implemented by them as they try to solve the inverted-pendulum problem. They compared their FNN and RL models with the classic PID model and discovered that the FNN and RL model was able to stabilize the robot efficiently with fewer oscillations when compared to the PID controller.
(3) A Youtuber named Joop Brokking developed a PID based self-balancing robot that had a complementary filter included for more accurate measurements of the tilt angle(4). His model consisted of a robot that would start balancing itself as the robot is lifted to a certain tilt angle. The robot used stepper motors which kept the robot fixed at a specific position when the motors weren't moving. The system was very stable and was also successful in solving the inverted pendulum problem.
(4) A. N. K. Nasir, M. A. Ahmad, and R. M. T. Raja Ismail are authors of a paper titled "The Control of a Highly Nonlinear Two-wheels Balancing Robot: A Comparative Assessment between LQR and PID-PID Control Schemes" published in the World Academy of Science, Engineering and Technology Journals (Vol:4, No:10) in 2010 where they reported their development of LQR and PID controllers and measured the efficiency of both controllers(5). They presented the control structure of both controllers and found out that both controllers were capable enough to balance the robot but the LQR was slightly better in balancing the nonlinear system of the self-balancing robot.
(5) Victor Vicente Abreu from the University of Madrid developed a balancing robot that was modeled in the MATLAB software using the LQR controller(6). The robot used a PID controller with the inclusion of a Kalman filter for more

accurate measurements of the tilt angle. From their tests, they noted as they added more mass to the top-center position, the robot became more unstable and struggled in balancing itself as the center of mass had moved away from the vertical axis.

LOCAL STUDIES:

(6) Amin Alqudah and Asseel Qasaimeh from Yarmouk University, Irbid, Jordan reported a journal where they discussed how they enhanced and tested the PID controller(7). They combined fuzzy logic with the controller to enhance the output as the fuzzy logic allowed control over the gain and time constant during different operation times.

(7) Emerson Arabia's DeltaV controllers are controllers that are used in oil and gas plants for plant operating procedures(8). The controllers make use of self-tuning, neural networks, and fuzzy logic with the inclusion of PID controllers for efficient control of parameters such as gas level, or the amount of oil that is passed through a pipe. The controllers establish a plant control standard and also provide a library of control templates, designed for the plant to be used or reused.

(8) H. M. Omar, A. M. Elalawy, and H. H. Ammar published their paper titled "Two-wheeled Self-balancing robot Modeling and Control using Artificial Neural Networks (ANN)," in the IEEE Xplore platform where a mathematical model was developed and the state space was developed to model the plant of the system(9). Since the system of a Self-balancing is non-linear, the used Nonlinear Autoregressive Exogenous (NARX) Neural Network model with the use of recorded data architecture. They were able to conclude the NARX model was better than the mathematical model. For controlling the self-balancing robot, two controlling architectures including the PID Controller and the Model Reference Adaptive Control(MRAC) were considered. The two controllers were tested with the result being that both controllers are suitable for the control problem but MRAC gave the best response when random inputs were given with good disturbance rejection.

Sources	Similarity
Learning to Control Two-Wheeled Self-Balancing Robot Using... download citation | learning to control two-wheeled self-balancing robot using reinforcement learning rules and fuzzy neuralthis paper present a novel method to control the balance of a two-wheeled robot by using reinforcement learning and fuzzy neural networks(fnn) which can... https://www.researchgate.net/publication/224346658_Learning_to_Control_Two-Wheeled_Self-Balancing_Robot_Using_Reinforcement_Learning_Rules_and_Fuzzy_Neural_Networks	9%

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	349	Date	December 01,2020
Characters	2120	Exclude URL	

0%	100%	0	16
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

(9) Ahmed, O., El-Amin, E., Salem, A., and Mohammed, A. of the Sudan University of Science and Technology College of Engineering School of Electrical and Nuclear Engineering developed a mathematical model of a two-wheeled self-balance robot system using physical and electrical laws(10). After linearizing the model, they were able to simplify the model. By manually tuning the PID controller of their robot, they were able to determine the Kp, Ki, and Kd constants for the best system response. The best response was found at: Kp= 40, Ki= 1 ,and Kd= 1. They observed high oscillation around the setpoint with the robot not settling down. After passing a certain angle, their system would become unstable with the robot not regaining balance. Yet, the results obtained from the manual tuning was acceptable and had met their expectation.
(10) Yunus Celik and Mahit Güneş from Karamanoğlu Mehmetbey University, Turkey developed a robot that can track colorful objects while balancing itself on two wheels(11). The robot was balanced using a PID controller with the optimal values for the controller found using the trial and error method. They used the Kalman filter to make the robot stable and less noisy and made use of a color detection algorithm where a blue object was chosen as the object to track while the robot balanced itself.
3.3 SYNTHESIS:
From the research, it was learned that there are various methods of solving the inverted pendulum problem with the PID controller, the LQR controller, and methods that integrate or customize both PID and LQR controllers. The controller that I have chosen would be the PID controller as I have used parts of the PID controller for my previous projects.
It was also noted that mass distribution is essential for the stability of the robot as it tries to balance itself. The best option for the mass distribution would be to place most of the weight near the wheels at the center of the robot frame to achieve a vertical center of gravity. This would reduce the chances of instability in motion as the robot tries to balance itself.

Sources	Similarity

CHAPTER 4
DESIGN SPECIFICATIONS
4.1 RESEARCH PARADIGM:
Phase 1: Project Development
Knowledge Requirements
PID Control theory.
Inverted Pendulum Problem.
Arduino Programming.
Coppelia Simulation Scene building and programming.
Mechanical design
Hardware Requirements:
Arduino Microcontroller
Ultrasonic Sensor
Software Requirements:
Coppelia Simulation
Arduino IDE
Phase 2: Assembly
Part Assembly
Building robot frames within the Coppelia Scene environment.
Programming the ultrasonic sensor using an Arduino
Program the robot in the Coppelia simulation software.
Phase 3: Prototype
Testing
Test the self-balancing robot.
Figure 4.1. Research Paradigm in Phases.
The project development phase requires the acquisition of fundamental knowledge and resources that can aid in the next phase of the project. An understanding of each requirement is essential for development as every component and resource are interdependent for the final result.
4.2 PROJECT DEVELOPMENT:
The project began with the idea of developing a self-balancing robot with a suitable control algorithm. The research began by understanding the inverted pendulum problem and how it can be solved with the use of different control theories. From personal experience, the PID controller has been well-versed to me and thus, I have chosen the PID based control architecture. Since I needed inputs on how the robot is balancing on its own, I chose to use a gyroscope and an accelerometer to measure the tilt angle of the robot. Initially, the plan was to develop an Arduino-based prototype of the self-balancing robot but the plan had to change due to the COVID-19 pandemic.
4.2.1 PROJECT ISSUES:
Due to the Coronavirus pandemic, a lot of challenges and risks were introduced, especially the risk of receiving the components late and also not having access to an in-person consultation with my project advisor. Also, the closure of the university caused me to not access the university resources on previous thesis papers of students who had worked

on a control problem such as the Inverted Pendulum Problem. The solution to these challenges and risks came with distant consultation with my project advisor and online research. My plan changed with the decision to build my robot virtually on a virtual platform known as Coppelia Sim. and interfacing that robot with a physical Arduino microcontroller for external sensory inputs.
Figure 4.2 Coppelia Sim(12).
Coppelia Robotics is a small Swiss SME active in the field of robotics simulation(13). This is a simulation software where a robot of any kind can be designed, programmed, and tested. Various pre-built robots are already present within the software such as line-following robots, humanoids, and more. In addition to the virtual capabilities that are present, this software can interact with external hardware such as Arduino Uno for receiving sensory inputs. Therefore, such capabilities allowed me to bring changes to my project by turning the focus towards the interaction between reality and virtual reality. Such interaction became possible by adding an external ultrasonic sensor that would measure the displacement of my hand to create a force that is applied to the virtual self-balancing robot within the Coppelia Sim. software. By doing this, we can verify the robot's resistance to external impact as it tries to balance itself virtually. The Lua programming language has been chosen to program the robot.
Figure 4.3 A Virtual Humanoid Robot within Coppelia Sim(14).
4.2.2 MATHEMATICAL MODEL:
Before designing the actual robot, it was important to develop a mathematical model by deriving the equations of motion for the inverted pendulum model to understand the non-linear dynamics of this model. The diagram below shows an ideal mass with an inverted pendulum.
Figure 4.4 Inverted Pendulum Model
Initially, the model is assumed to be on a frictionless surface with a mass "M" that has a force "F" applied to it. The pendulum is made up of a mass "m " with a rigid massless rod attached to the same mass. The force of gravity 'g' is present and acts on the model. Using Lagrange's Equations, the equations of motion are derived for the inverted pendulum model(15). Lagrange's equations are equations that make the use of kinetic(T) and potential energy(V)for solving equations of motion(16). We initially begin with the kinematics of the suspended mass 'm' on the pendulum across the x-axis and y-axis:
Xm= -L* Eq. 4.1
Ym= L* Eq. 4.2
By taking the derivative of equation 4.1 and 4.2 from each side, we get:
= -L** Eq. 4.3
= -L** Eq. 4.4
Then, we calculate the potential energy(V),
V= m*g*Ym= m*g*L* Eq. 4.5

The kinetic energy(T) is then calculated,
T= ½*M+ ½ *m* (+)
We substitute values for Xm' and Y m',
T= ½*M + ½ m((- L**)2+ (-L**)2)
T= ½*M+ ½ m((-2**L** + (L2**) + (L2**))
T= ½ (M+m) + ½ m(-2**L** + (L2**)+ (L2**))
T= ½ (M+m)+ ½ m(-2**L**+L2*(+))
T= ½ (M+m) + ½ m((-2**L**) +L2*)
T= ½ (M+m) + ½ mL2- (mL) Eq. 4.6
The following are the Lagrange's equations:
Eq. 4.7
Eq. 4.8
By substituting values for T and V in equation 4.8, we get:
(½ (M+m) + ½ mL2- ((mL)*) - (m*g*L*) Eq. 4.9
Therefore, to attain the first equation of motion; we take the derivative of equation 4.9 in terms of :
= (M+m)- (m L)*= M + m- (m L)*
Then, in terms of t:
= (M+m)- ((m L)*- ((m L)*)= F(t)
→ F(t)= (M+m)- (m L)* + ((m L)*) Eq. 4.10
Therefore, an equation relating the force applied(F(t)) with the angle of tilt() and the acceleration of the cart().
To attain the second equation of motion; we take the derivative of equation 4.9 in terms of and then, in terms of t:
= (mL2)- ((m) + (m)
By subtracting the solution , we get:
= mL2- (m)+ (m) - (m) - (mg)= 0 (where = (m) + (mg)= 0).
= mL2- (m) - (mg)= 0
By dividing mL on both sides, we get
- (*) - (g*)= 0 Eq. 4.11
Finally, the second equation of motion is achieved.

Sources	Similarity

the theory of the natural urban transformation process

THE THEORY OF THE NATURAL URBAN TRANSFORMATION PROCESS: The Relationship between Street Network Configuration, Density and Degree of ...

https://repository.tudelft.nl/islandora/object/uuid:03eb7bf8-8252-4386-9b09-8f7d5dec3a9a/datastream/OBJ/download

3% **Page 3**

4.2.3 HARDWARE:
The hardware needed for this project includes a microcontroller, a proximity sensor, and a desktop computer with sufficient memory for installing the Coppelia Simulation software.

Figure 4.5 Microcontroller(17) + Ultrasonic(18) + Desktop Computer(19).
4.2.4 PROTOTYPE DESIGN:
Figure 4.6 Prototype Design.
The prototype design of the self-balancing robot consists of a single frame that has two wheels connected to it. The dimensions of the frame and the wheels have been presented in the diagram above. The motors present in the Coppelia simulation software are attached.
4.2.5 SOFTWARE:
The software needed for this project include the Coppelia Simulation software and the Arduino IDE.
4.2.6 MICROCONTROLLER:
The microcontroller that has been chosen is the Arduino Uno board due to its compatibility with Coppelia Sim and due to its cost.
4.3 DESIGN STANDARDS:
IEEE Std 100-1996: This paper provides the standard dictionary for electronic appellations. This standard has more than 24,255 terms and definitions associated with electronics. Proponents of this project used this documentation heavily when defining acronyms in the design, development, and documentation of the project at hand.
SAS 123-31 Programming Guidelines: This paper presents a group of programming guidelines and conventions that will be considered in developing code to make sure that it's clean, efficient, transferable, and maintainable.
4.4 MULTIPLE DESIGN CONSTRAINTS:
Design constraints related to cost, time, and flexibility will be presented comparing the options of having a complete physical robot and a virtual robot with hardware additions.
The Figure of Merits analysis is used. Input values, scoring functions, and the weight are the three parameters that are considered for this type of analysis. The inputs include variables such as costs, time-span, and percentage values for the possibility of success. The weights are assigned by the project designer and can range from 0 to 1. The scoring functions are used to calculate the score and are presented in equation 4.1 and equation 4.2.
The functions require three inputs including the input value, the upper limit of the input value, and the lower limit of the input value. A higher score is achieved from equation 4.1 when there is a high input value and the higher score is preferred for this parameter while a higher score is achieved from equation 4.2 when there is a lower input value. The higher score from the lower input value in equation 4.2 is more preferable than the higher score from the higher input in equation 4.1.
Score= Input Value- Lower Limit of the Input ValueUpper Limit of the Input Value- Lower Limit of the Input Value
Eq. 4.12 Higher input= Higher Score.
Score= Upper Limit of Input Value- Input ValueUpper Limit of the Input Value- Lower Limit of the Input Value
Eq. 4.13 Lower input= Higher Score.
The following parameters have been utilized for the FoM criterion:
i. Cost: The constraints in costs will be presented in BHD.

ii. Availability: This parameter refers to the time it takes for the component to arrive from the manufacturer to the proponent's location.
ii. Reliability: This will be calculated by determining the failure percentage. The greater the failure rate, the lower the reliability for the device. The failure percentage is represented after combining the failure percentage of each component.
iii. Safety: A percentage is assigned to the level of safety each component has. This is assigned by taking into consideration if the component is RoHS compliant or not where there are no hazardous substances such as mercury, cadmium, and other such chemicals that are used in manufacturing electronic devices.
After completing the research on the components, the proponent has identified the following design plan as shown in Table 4.1 below. All the design plans fulfill the required criteria for prototyping the proponent's project.
Table 4.1 Design Plan
Component
Plan A: Physical Robot
Plan B: Virtual Robot*
Hardware
Plywood Sheets- 2 weeks
(6 BD)
Virtual Robotic Components
(Free)-0
Ultrasonic Sensor
'HC-SRO4' (20)- 2 weeks
(5 BD)
HC-SRO4
(5BD)-2
Gyroscope
'GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue' (21)
(2 BD)-1 week
Gyro Sensor
(Free)-0
Accelerometer
'GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module - Blue' (21)
(2 BD)-1 week
Accelerometer
(Free)-0
Motor
'Geeetech 1.8-Degree Nema 14, 35 BYGHW Stepper Motor for 3D Printer' (22)
(11 BD)- 2 weeks
In-built DC Motor
(Free)-0
Microcontroller
Arduino Uno (23)
(4 BD)-4 weeks
Arduino Uno
(4 BD)-4
Power Supply
'E-flite 11.1V 2200mAh 3S 30C LiPo Battery: EC3, EFLB22003S30' (24)
(9 BD)- 2 weeks
Computer Power Supply for the robot & 220V external power supply for the Arduino.
(Free)-0
*The virtual robot will be moved physically with the use of ultrasonic sensors.
The following parameters were chosen based on the given reasons:
Table 4.2 FoM Parameter Ranges
Parameter
Value
Reasoning
Cost Upper Limit
700 BD
This value is roughly the cost of a high-end self-balancing system.
Cost Lower Limit
30 BD
The cheapest self-balancing system is priced around this value.
Availability Upper Limit
4 Weeks
If no Express shipping is chosen, the shipping takes around 4 weeks.
Availability Lower Limit

1.5 Weeks
If Express shipping is chosen, the shipping takes around 1.5 weeks.
Accuracy Upper Limit
99%
To achieve 100% accuracy is impossible in the real world yet 99% accuracy is justifiable.
Accuracy Lower Limit
35%
Lower than or equal to 35% would certainly lead to failure once the product is in use.
Safety Upper Limit
96%
A safety of 100% is not always certain, therefore a percentage of 96 is chosen.
Safety Lower Limit
52%
When dealing with safety, the percentage needs to be above 52%. If the percentage goes below this, the prototype can be safe half the time which is not acceptable.

Sources	Similarity

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	370	Date	December 01,2020
Characters	2786	Exclude URL	

0%	100%	0	10
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

Table 4.3 FoM Plan A
FoM
Plan A
Input Value
Score Function
Weight
Score*Weight
Cost(BD)
39
0.99
0.25
0.248
Availability(weeks)
3
0.6
0.25
0.15
Accuracy(%)
75
0.63
0.25
0.16
Safety(%)
85
0.75
0.25
0.19
Total
1
0.8
Table 4.4 FoM Plan B
FoM
Plan B
Input Value
Score Function
Weight
Score*Weight
Cost(BD)
9

1
0.25
0.25
Availability(weeks)
2
0.8
0.25
0.2
Accuracy(%)
90
0.86
0.25
0.215
Safety(%)
95
0.98
0.25
0.245
Total
1
0.91
Plan B was chosen for having the best parameter values when compared to Plan A. The score for the cost was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.002 points:
Graph 4.1 Scatter Plot for Score Vs. Cost(BD)
The score for the availability was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.002 points:
Graph 4.2 Scatter Plot for Score Vs. Availability(Weeks)
The score for the accuracy was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.05 points:
Graph 4.3 Scatter Plot for Score Vs. Accuracy(%)
The score for the safety was slightly higher for Plan B when compared to Plan A as shown below by a difference of 0.055 points:
Graph 4.4 Scatter Plot for Score Vs. Safety(%)
In conclusion, the FoM for both plans with consideration of the parameters including cost availability, accuracy, and safety shows that Plan B is a suitable option to go forward with as it had a higher overall score when compared to Plan A(0.91 > 0.8). This option is a healthy alternative especially considering the project issues that had occurred as mentioned previously.
4.5 PROJECT BLOCK DIAGRAM:
Figure 4.7 Project Block Diagram.

The project block diagram combines the software aspects and the hardware aspects of the robot as shown in the figure above.
4.6 PROJECT FLOWCHART:
Self-balancing Robot Flowchart:
Figure 4.8 Flowchart 1.
Ultrasonic Sensor Sub-Routine Flowchart:
Figure 4.9 Flowchart 2.
4.7 CIRCUIT DIAGRAM:
Figure 4.10 Circuit Diagram.
4.8 BILL OF MATERIALS:
Table 4.5 Bill of Materials
Item#
Description
Quantity
Total(BD)
Arduino Uno
1
9
USB 2.0 cable type A/B
1
1.5
Ultrasonic Sensor
1
5
Total(BD)

15.5
4.9 GANTT CHART:

Sources	Similarity

CHAPTER 5
DESIGN PROCEDURE, FUNCTIONAL ANALYSIS, AND IMPLEMENTATION
5.1 PROJECT DESCRIPTION:
To build and program the Self-balancing robot, specific hardware and software components are essential. For my Self-balancing robot, the Coppelia Simulation software is used for the creation and simulation of this robot. Within the Coppelia Simulation software, I will be using two motor-enabled rotary joints for movement, two wheels that will be attached to the joints, one Accelerometer sensor, a cuboid-shaped body for the chassis of the robot, and a dummy that is attached to the frame of the robot. The dummy and the accelerometer will be used to measure the angle of tilt of the robot. The Gyro sensor is not used due to the inaccuracy of the values found. The Arduino Uno and the Ultrasonic Sensor HC-SR04 will also not be used in creating external disturbances. Instead, a cuboid will be translated across the x-axis and the y-axis to create the external disturbances. The Lua programming language within the Coppelia Simulation project will be used to program the Self-balancing robot.
5.2 PROJECT FUNCTION ANALYSIS:
The Self-balancing robot begins functioning when the “Start/resume” simulation button is pressed within the Coppelia Simulation Project. After pressing this button, the simulator and the Accelerometer begin measuring the Euler angle around the x-axis and the linear acceleration(across the y and z-axis) respectively. These values are then used to calculate the angle of tilt of the robot. Once the angle of tilt is determined, the error that occurs when that angle deviates from the set angle or the setpoint is found. The Setpoint for the robot is 0 degrees which is when the robot is perfectly balanced. Once the error is found, the error is summed with the previous error to find the total accumulated error(The summed error is initially zero when the simulation starts). After finding the summed error, this error is fed into a PID controller for achieving an output that is given to the motor enabled joints. A factor of 0.8 is multiplied with the calculated value.
The position of the frame of the robot is also recorded across the y-axis. The initial position of the robot is subtracted with the current position of the robot to get any error that may occur if the robot begins moving from its initial position while balancing itself. This error is then recorded and summed. The summed error is then fed into a second PID Controller and the output is multiplied by a factor of 0.2.
The outputs from both PID controllers are summed to find the appropriate target velocity for the motor joints in degrees. The motor enabled joints then move accordingly to balance the robot while trying to maintain the robot’s position. Thus, a self-balancing robotic system is achieved.
While the Self-balancing robot tries to balance itself virtually, a light cuboid-shaped body is translated across the x and y-axis about the world frame to create external disturbances on the self-balancing robot.
5.3 PROJECT IMPLEMENTATION:
The project implementation procedure is as follows:
The Coppelia Simulation software is installed on a computer.
Within the Coppelia Simulation environment, the chassis of the robot is designed by shaping a cuboid in the dimensions mentioned within the Prototype Design Section in Chapter 4.
The wheels that have the rotary joints attached are connected to the chassis.
The Accelerometer sensor is attached to the center of the chassis.
A 5g cuboid is placed near the self-balancing robot.

Figure 5.1 Cuboid For Shaping the Chassis Figure 5.2 Motor Enabled Rotary Joint
Figure 5.3 Accelerometer
5.4 EVALUATION PROCEDURES:
For the evaluation procedure, the Failure Mode And Effects Analysis(FMEA) is performed:
Table 5.1 Evaluation Procedure

Item#	Component Description	Failure Mode	Effects	Safeguard	Actions
1	Chassis	Disconnects with other parts when the movement starts	No movement occurs as the robot breaks down.	Ensure proper connections are made between the chassis and the remaining parts	Stop the robot Simulation and check all the connections made to the robot.
2	Dummy	Not giving proper Euler angle around the x-axis. Not giving accurate position values in the y-axis.	The tilt angle cannot be found using this component	Ensure that the dummy is placed at the center of the chassis for accurate measurement..	Stop the robot Simulation and check the connection and position of the dummy in the scene hierarchy.
3	Accelerometer	Not giving acceleration values. Not giving accurate acceleration values	The tilt angle cannot be found from the accelerometer measurement	Check the connections/programming	Stop the robot Simulation and check the connections of the Accelerometer sensor. Also, check the Accelerometer sensor script and the Main program script within Coppelia Sim.
4	Rotary Motor Joints	Joints breaking when the movement starts. Wheels disconnect when the movement starts.	No movement occurs as the wheels of the robot disconnect.	Check the connections/programming.	Stop the robot Simulation and check the connections of the rotary joint with the wheels.
5	Computer System	No power	The complete shutdown of the project	Check the connections	Check the source of power and ensure all connections of the computer are properly made.

Sources	Similarity

5.5 COMPONENT SPECIFICATIONS:
Coppelia Simulation:
The Coppelia Simulation software is a simulation software where robots can be built and programmed in many ways. The software offers different ways to program robots either locally using the Lua programming language or by using the remote API interface to program from software like MATLAB, Python, or Java or by using the ROS(Robot Operating System) communication method. For my program, I have used the local programming language Lua to program my Self-balancing robot. The Newton physics engine is used for the dynamics of the robot with very accurate simulations. Within the software, a scene is created where the robot can be built and simulated. Within the Scene, a Scene Hierarchy is presented which includes all the components of the robot. The components include the sensors(Accelerometer), the main robot frame, a dummy object for obtaining the orientation, and the robot motor joints. Also, within the Scene Hierarchy graphs that present the tilt angle and the error in the change in position respectively. Otherwise, there are default objects created within the software automatically that allow the creation of the floor, default lights, and the cameras for the robot. A side view camera has also been placed to get the side view of the self-balancing robot. The Scene hierarchy has been presented below
Figure 5.4 Scene Hierarchy.
The Robot that was built within the software has been presented below:
Figure 5.5 Main Self-balancing robot.
Table 5.2 Robot Specifications
Robot Specifications
Weight of the Frame
0.25 kg
Dimensions(X*Y*Z) of the Frame
0.03m *0.01m*0.07m
Dimension(X*Y*Z) of the wheels
0.006m *0.027m*0.027m
Torque of Motor Joints
5 N*m

The weight of the frame of the robot is 0.25 kg with dimensions 0.03m on the x-axis, 0.01m on the y-axis, 0.07m on the z-axis. The wheels have a diameter of 0.027m across the y-axis and a depth of 0.006m approximately across the x-axis. The motor joints attached to the wheel are To program the robot a child script needs to be created and attached to the frame of the robot as shown below:

Figure 5.6 Child Script.
There are two kinds of child scripts that can be created for the robot including a threaded child script and a non-threaded child script. For my program, I chose a non-threaded child script. Initially, within the non-threaded child script, the function sysCall_init is written. Handles for each object present in the scene are stored in variables using the 'sim.getObjectHandle()'. A tube communication method is initialized to read the accelerometer values by writing the following code:

“accelCommunicationTube=sim.tubeOpen(0,'accelerometerData'..sim.getNameSuffix(nil),1)”
The sysCall_sensing() is then initialized where variables used in the program are declared. An if statement with the condition if ‘1==1’ is created to form a continuous loop. Within the continuous loop, the program for measuring the tilt angle and for controlling the motors to correct the tilt angle is present. The loop time(dt) for the whole program is initialized at 10ms.
To find the angle of tilt of the robot, the Euler angle around the x-axis of the dummy placed inside the frame is measured by using the ‘sim.getObjectOrientation’ and the output from the accelerometer sensor is measured. This technique replaced the Gyro sensor measurement method as the Gyro sensor was giving angular values that were significantly different when compared to the accelerometer angle values.
The Euler angle around the x-axis is received in radians and therefore, this value must be converted to degrees. To convert the value to degrees, the following equation is used:
Eangle= euler[1]*(180/math.pi) Eq. 5.1
Where euler[1] is the value of the orientation of the dummy present in the frame of the robot around the x-axis and where Eangle is the Euler angle in degrees.
To find the angle of tilt from the accelerometer sensor the following relationship is used:
Figure 5.7 Accelerometer Sensor Tilt Angle Relations(25).
From the figure above, the Tilt Angle ϕ can be found by taking the inverse tangent of the linear acceleration across the y-axis and the z-axis. Also, the angle found needs to be converted to degrees. In the Self-Balancing robot program, the following equation was used to find the tilt angle and convert that angle to degrees:
accangle= (math.atan(accY/accZ))*(180/math.pi) Eq. 5.2
Where the accangle is the calculated angle from the output of the accelerometer, accY is the linear acceleration across the y-axis and accZ is the linear across the z-axis.
The angle of tilt is then found by using a complementary filter which combines the Euler angle and the accelerometer angle to get the best output by filtering the noise in the output. The complementary filter is used to find the tilt angle in the equation shown below:
TiltAngle= (0.9999999*Eangle)+(0.0000001*accangle) Eq. 5.3
Where Eangle is the measured euler angle across the x-axis and accangle is the calculated angle from the output of the accelerometer.
After finding the tilt angle, the deviation of the title angle from the target angle is calculated to find the error by using the following equation:
error = targetAngle-TiltAngle Eq. 5.54
Where the target angle is 0 degrees. After finding the error, this error is summed with the previous error to find the value for the summed error or errorSumm as shown in the equation below:
errorSumm= errorSumm +error Eq. 5.5
Apart from measuring the tilt angle, the change in the position of the frame of the robot is also noted by measuring the position of the dummy. The purpose of measuring the position is to keep the robot around its initial position while the robot balances itself. This measurement is made using the “sim.getObjectPosition(dummy,-1)”. The position across the y-axis is then found by entering the command “ positionY= position[2]”. The target position of the robot is its initial position at approximately 0.23m from the reference frame. The error in the position is then found by using the following equation:
errorP= targetPosition - positionY Eq. 5.6

Sources	Similarity

After finding the error, this error is summed with the previous error to find the value for the summed error of the position or errorSummP as shown in the equation below:
errorSummP= errorSummP +errorP Eq. 5.7
The summation of the error in the tilt angle and the summation of the error in the position (across the y-axis) are then used to balance the robot to the target angle and keep the robot balancing around its initial position.

To achieve this, 2 PID controllers are used. The PID controller consists of the Proportionality, Integration, and Derivative controllers for correcting the tilt angle of the robot and also for correcting the position of the robot. The following diagram illustrates this relationship:
Figure 5.8 PID Controller Diagram.
For the first PID controller; the Kp, Ki, and Kd constants are 45.5, 35 ,and 0.0000045 respectively while for the second PID controller; the Kp, Ki, and Kd constants are 100, 0 ,and 0 respectively. The outputs from both PID controllers are combined to give the output to the motor joints. The motor speed has been limited from -500 to 500 degree second. The equation for both PID controllers is presented below:
motorPower =(0.8)*((Kp*(errorSumm))+(Ki*(errorSumm)*0.01)- Kd*((TiltAngle-prevAngleF)/0.01))+(0.2)*((Kp1*(errorSummP))+(Ki1*(errorSummP)*0.01)- Kd1*((positionY-prevP)/0.01)) Eq 5.8

Apart from the PID correction, a condition has been placed where if the angle of tilt is between -7 and 7 degrees; the velocity would be 0 degrees/s. For creating the external disturbances, a 5-gram cuboid is made with the dimensions 0.01m*0.01m*0.08m is translated to push the self-balancing robot as shown below:
Figure 5.9 5g Cuboid

5.6 COST-BENEFIT ANALYSIS:

Cost-Benefit analysis is a procedure utilized when making business choices. The benefits or advantages of the business activity are summed, and the expenses related to that business activity is subtracted to achieve the Cost-Benefit analysis.
Table 5.3 Costs
As mentioned in Table 5.3, the project sources are broken down into elements to find the total cost. The program source elements include Material Purchasing, Equipment Purchasing, Product Installment, Product Distribution, Facilities, Service and Maintenance, Hardware Development, and Software Development. The total cost of implementing this project is computed for the current year(2020) and the upcoming 5 years.
Table 5.4 Benefits
In table 5.4, the total benefit is computed for the project. There are four benefit sources including Cost Reduction, Enhanced Revenues, Labor Reduction, and Decreased Overhead. The benefit for each year of implementation is

computed. Initially, the total benefit of each benefit source is summed and then, the total sum is multiplied by the confidence factor to get the benefits claimed for analysis. After finding the benefits claimed, the program's grand total benefit is found by adding the benefits of all the years.

As shown in the table above, no benefits are collected in the first two years as the collection of the benefits is expected to begin from the third year

Table 5.5 Summary

Table 5.5 presents a summary of the calculations for the undiscounted cash flow and the discounted cash flow. For the undiscounted cash flow; cost, benefits, net cash flow, discount rate, and base year are used to compute the net result. For the discounted cash flow; cost, benefits, net cash flow, cumulative, net present value, and the internal rate of return are used to compute the net result.

Graph 5.2 Undiscounted Cash Flows

In graph 5.2, the undiscounted cash flow graph is made using the values from table 5.4. The relationship between the cost and benefit is displayed among the several years of implementation of this project. From 2020 to 2021, only the cost is displayed while from 2022 to 2025; both cost and benefits are displayed. It can be noted that from 2024 to 2025, the benefit was constantly greater than the cost.

Graph 5.3 Discounted Cash Flows

In graph 5.3, the discounted cash flow graph is made using the values from table 5.4. The relationship between the cost and benefit is displayed among the several years of implementation of this project. From 2020 to 2021, only the cost is displayed while from 2022 to 2025; both cost and benefits are displayed. It can be noted that from 2024 to 2025, the benefit was constantly greater than the cost.

Graph 5.4 Discounted Payback

In graph 5.4, the discounted paycheck is made using the relationship between cost and benefit for the discounted cash flow. From 2020 to 2023, the cost is very high while the benefit is below zero dollars. From 2023 onwards, the trend of the curve changes with the increase of benefits in 2024 and 2025 as the cost is reduced from 2024 onward when compared to the benefits.

Interest: 0.07

Years: 10

Amount Borrowed: 5000

Table 5.6 Amortization Chart

The Amortization table displays the amount of money that needs to be paid in intervals to pay the total loan amount. The interest rates are also included in the table. The time allocated for paying the loan is 10 years for a loan amount of $5000. The remaining amount after all the payments are made is also included in the table.

Sources	Similarity

CHAPTER 6
DESIGN PROJECT SUMMARY AND CONCLUSIONS
SUMMARY:
The Self- balancing robot project was very special as it revolved around solving the inverted pendulum problem. The problem involved balancing an inverted pendulum by achieving a tilt angle of 0. There were so many different solutions that were witnessed during the research in the initial stages of the project. After the completion of the research, the PID Controller was eventually chosen by me due to my familiarity with this controller when compared to the other controllers that were used by others.
In software aspects, the Coppelia Simulation offered a great alternative for building the robot especially considering the current COVID-19 situation around the world. The software allowed me to build, compose, and program my robot without any difficulties. Sensors including the gyro sensor and the accelerometer were used to calculate the tilt angle but the former was not used to its inaccuracy in getting angular values. Also, the ultrasonic sensor was not used for creating the proposed external disturbance due to inaccurate transmission of ultrasonic sensor values within the simulation scene. The P (proportionality) controller in the PID also used the summation of the errors for finding the motor output which was theoretically incorrect as I learned later. I had not relied on a mathematical model that had made my process for finding the Kp, Ki, and Kd constants lengthy by trial and error based on the reaction of the self-balancing robot.
Figure 6.1 Self-balancing Robot Scene
Yet, I was able to find solutions by incorporating Euler angles that were measured by the software using the command 'sim.getObjectOrientation'. For creating the disturbances, I was able to create cuboid-shaped objects within the scene that would bump with the self-balancing robot to create those disturbances.
When the robot would balance itself, the tilt angle would oscillate between -7.5 to 7.5 degrees as shown in the graph below:
Graph 6.1 Title Angle Change(Time=10 seconds)
Such an output allowed the robot to balance itself without falling moving abruptly. The position of the robot also remained near the initial position as the degree error was almost 0 meter in the y-axis as shown in the graph below:

Graph 6.2 Position Error Change(Time=10 seconds)
The robot was balancing itself and was also able to withstand shock or disturbance that was created when it bumped with the cuboid. When subjected to a push by a 5g object, the robot would oscillate more for a few seconds but would later return to its original movement as shown in the graph below:
Graph 6.3 Tilt Angle Shift After Disturbance
It should be noted that since the weight of the frame is 0.25 kg, heavier weights can push the robot down with stronger forces.
CONCLUSION:
The self-balancing robot is a robot that can be used for many applications such as for cargo deliveries where the area provided by the application is limited. The controller for controlling such a robot which is the PID controller for this project is used in many real-life applications such as when adjusting the altitude of a rocket that has launched off the earth's surface. In conclusion, the project was successful as a robot was made that could balance itself and could

withstand a certain degree of external disturbances.

CHAPTER 7

RECOMMENDATIONS

Recommendations on this project from the proponent would be the following:

Always seek alternative solutions if the current solution is not working out to save time.

Use mathematical modeling to build and design the robot. The purpose of this is to maintain the system's response even if there are changes in the dimensions of the robot.

Familiarize yourself and implement other control methods including the LQR method for getting better stability from the self-balancing robot.

Research well on the different control theories.

Pay attention to the factors that may contribute to a certain instability in your system.

Always stay in touch with someone who may have more knowledge than you so that you can grow your knowledge.

Keep testing and be prepared to face sudden challenges with patience.

Sources	Similarity

INDIVIDUAL CV

Faizaan Mohammed Mustafa

King Faisal Bin Abdulaziz Rd., Al Khobar 31952, Saudi Arabia
Email: mustafafaizaan2@gmail.com
Tel: +973 35572137

Objective: To acquire the highest level of education in an Engineering field so as to obtain a position that provides valuable experiences in life skills and organizational skills. Doing so also provides familiarity with common research skills, which may assist me in my future educational endeavor and my professional career.

Responsibilities:

● Current AMA International University Robotics Mentor for the IET GCC Robotics Competition 2020 and the WRO Competition 2020.

Education / Academics:

School: INTERNATIONAL SCHOOLS GROUP (ISG), DAMMAM Major: AP Physics, AP Calculus AB, Economics
Graduation anticipated date: JUNE 2016
GPA: 3.63

SAT score: 1730, Writing 590, Reading 520, Math 620

IELTS Score: Reading 8, Listening 8.5, Speaking 7.5, Writing 7, Overall 8

University: AMA INTERNATIONAL UNIVERSITY BAHRAIN
Major: Mechatronics Engineering, Graduation anticipated date: DEC. 2020 CGPA: 3.74.

Personal Skills:

● Consistently shows leadership qualities by initiating and promoting new activities in school and university including the Flash Mob for Breast Cancer Awareness, Walk For Charity, Team Building activities, Secret Admirer Event, AMA Gaming Tournament for Charity, and AMA University Breast Cancer Awareness Event 2016.

● Displayed good public speaking skills by earning the Distinction“Most Likely to Become a UN Delegate " in AISRMUN 2014.

● A hard-working young boy who is willing to learn more. Contains the qualities of being responsible, motivated, educated, and punctual. A friendly person who enjoys helping others.

● Accepts feedback and takes constructive criticism well.

● Able to develop and maintain good relationships with classmates, colleagues, superiors, and teachers.

● Being a positive role model for younger students and siblings.

● Ability to produce consistently accurate work even whilst under pressure.
● Willingness to learn and teach new things.

● Effective time management skills and ability to prioritize tasks in order of importance.

● Going the extra mile to make a difference, having the drive to lead and succeed.

Achievements:

● Received distinction in AISRMUN 2014 for the delegate who is “Most Likely to Become a UN Member ".
● Gained acceptance to the National Honor Society in grade 10.
● Placed on AMA University’s President's List for the acquired GPA in 2017.
● Winner of the AMA Public Speaking Competition in 2017.
● Finished with a World Rank of No.11 in the World Robot Olympiad 2019 Hungary as a member of the AMA Robotics Team traveling to Hungary.

Honors, Awards & Memberships:
● Awarded High Honor Roll Certificate in every Semester (Grade 9,10,11,12).
● National Honor Society Treasurer 2014-15.
● National Honor Society Media Manager 2015-16.
● Student Council Member 2013-14.
● Member, Chess Club 2013-14.
● Member, SAT Club.
● Member, Modern United Nations 2013-15.
● NSHSS member since 2015.
●Community Engagement Club President 2016-17
●AMA Public Speaking Competition Winner 2017.
●AMA Robotics Competition 2017 Winner.
●IEEE GCC Robotics Competition 2018 Local Round Winner
●IET GCC Robotics Competition 2019 Local Round Winner
●IET GCC Robotics Competition 2019 Special Award for Design Excellence
●World Robot Olympiad 2018 Local Round Winner
●World Robot Olympiad 2019 Local Round Winner
●IET GCC Robotics Competition 2020 Mentor
●World Robot Olympiad 2020 Mentor

Hobbies & Interests:

- Website designing.
- Public Speaking.
- Filmmaking.
- Photography.
- Reading.
- Technology.
- Robotics.
- Programming.

Computer & Special skills:

- Well versed with MS Office & Google Apps.
- Excellent writing skills in English.
- Well versed with software such as Adobe Photoshop, Indesign, Acrobat Pro, and others.
- Well versed with Final Cut Pro X.
- Well-versed with National Instruments Labview programming.
- Well-versed with Eclipse Java IDE.
- Well-versed with MATLAB especially with the Math, Control Systems, Fuzzy, and Neural network toolboxes.
- Well-versed with NI Multism
- Well-versed with PLC programming using Festo CodeSys software.
- Well-versed with C++ programming using RobotC software.
- Well-versed with Lua programming using Coppelia Simulation software.

Languages: English, French, Arabic, Hindi, and Urdu.

Work Experience:

1. NATIONAL HONOR SOCIETY Dammam, Saudi Arabia Media designer

● Helped in making photos, photos, and other forms of media for the NHS Dammam page.
● Maintained proactive commitment towards weekly posts.
● Focused on the suggestions provided by page viewers.

2. ALMEER SAUDI TECHNICAL SERVICES CO., AL Khobar, Saudi Arabia Media Advisor
● Proposed ideas with the web designer to promote the company through its website.
● Gave suggestions to improve the design of webpages, templates, animations, webpage contents, and documents, etc.

Volunteer Work:
● MUN 2014-15 movie night for Kindergarten.
● ISG Dammam Carnival 2015 booth setup.
● Bahrain Science & Technology University booth presenter.
● AMA International University Robotics Fair 2019 presenter.

References:

1. Mr. John Rutten Principal, ISG Dammam High School.
Email: rutte.j.01@isg.edu.sa

2. Mrs.Veron Naidoo High School Counselor, ISG Dammam High School.
Email: naido.v.01@isg.edu.sa

3. Mr. Gulam Mustafa Father and General Manager of Almeer Saudi Technical Services Co. Email: mustafagulam26@gmail.com Phone no.: +966508257980

4. Ms. Buena Gracia Canzana Acting
Vice President for Administration and Finance, AMA International University Bahrain
Email: bgacanzana@amaiu.edu.bh
Tel: 17787964

5. Dr. Amin El-Meligi
Dean of Student Affairs, AMA International University Bahrain
Email: aemeligi@amaiu.edu.bh
Tel: 17787900 6.

6. Mr. Ali Al Jasim
Director for External Engagement Department, AMA International University Bahrain
Email: ali.aljasim@amaiu.edu.bh
Tel: 17787989

7. Dr. Beda Aleta
Dean of the College of Engineering, AMA International University Bahrain
Email: btaleta@amaiu.edu.bh

8. Dr. Noaman Mohammad Noaman
Associate Dean of the College of Mechatronics Engineering, AMA International University Bahrain
Email: nmnoaman@amaiu.edu.bh