DEPARTMENT OF MECHANICAL ENGINEERING

FINAL YEAR PROJECT REPORT

Project Title: An Auto-Balancing Platform for Construction Robot Applications

Student Name: KIM Min Joong

Student No.: 54969649

Major: Mechanical Engineering

Supervisor Name: Dr. Liu Jun

Submission Date: 14/April/2023

Acknowledgments

First and foremost, I would like to express my special gratitude to Dr. Liu Jun for providing this opportunity to get along through this great project. Diving into the project with an expert in the field was extremely helpful in making significant progress.

Moreover, I would like to acknowledge Mr. Yonghan Kim and Mrs. Sun Mi Jin for supporting and inspiring me throughout my bachelor's degree.

Last but not least, I am grateful for all my family and friends, especially Ms. Jiyun Byun, Mr. Wooseok Kim, and Mr. Injae Song, who inspired and discussed the facing challenges regarding the project.

Abstract

This project was to design and optimize the Auto-Balancing Platform for construction applications. The Auto-Balancing Platform serves as a base for the robotic arm to place a brick wall on construction sites. As some tasks under construction sites are done on height, the robotic arm will work on top of the scissor lift like Genie 2646. Thus, an Auto-Balancing Platform, which will be placed on top of the scissor lift, is highly essential to ensure the reliability and safety of the robotic arm.

The project's main objective is to develop an Auto-Balancing Platform ensuring reliability and precision in the working environment for the robot arm. Thus, the appropriate mechanical design and fine-tuned control system of the Auto-Balancing Platform will be demonstrated. Moreover, the prototype of the platform will be developed to verify the working conditions.

For the prototype, Raspberry Pi 4B will be the central controller. Moreover, to obtain the angular deviations of the platform, MPU6050, an IMU sensor will be used. Appropriate filtering and sensor fusion techniques will be introduced to reduce sensor noise and gyro drift. To orient the platform, three continuous rotation servos will be used. Also, rack and pinion mechanisms will be utilized along the prototype to convert rotary motion into linear motion. To attain prompt and accurate response, PID control will be adopted to control the motors. Lastly, all of the control systems will be coded through Python through Raspberry Pi to control the Auto-Balancing Platform.

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1. Introduction

Robotics and automation are the talk of the town these days and are expected to play a crucial role in the upcoming industrial revolutions. Its prompt and accurate paced process accords cost and time reductions. Thus, the application of robotics on construction sites will be a very effective solution in terms of speed and economy. In particular, the application of robotics in construction industries saves labor costs and increases the construction process. However, prior to applying robotics in any field, its reliability and safety should be carefully considered.

The majority of construction robots are heavy-lifting machines which induce significant safety concerns. Even a tiny angular deviation can cause the center of gravity to shift, potentially leading to a dangerous fall. In particular, the robotic arm works on top of the scissor lift for the following project which the shift of center of gravity might cause dangerous fall more severely. To address this issue, an Auto-Balancing Platform can be utilized to suppress angular deviations and keep the robot balanced quickly on top of the scissor lift.

Thus, in the following report, development of the Auto-Balancing Platform for construction application will be examined.

1.2. Project Aims and Scope

The project's scope is to develop a prototype of an Auto-Balancing Platform, which will be placed on top of the scissor lift as a balancing basis to stabilize the construction robotic arm working on top. By providing a detailed analysis of the Auto-Balancing Platform, the aim is to contribute to the ongoing efforts to improve the safety and reliability of construction robots and facilitate their widespread use in the construction industry. To verify the control systems and design of the Auto-Balancing Platform, a prototype will be developed and demonstrated in the following project. The embedded control system, Raspberry Pi 4B, will be used to develop an algorithm to control the platform. The prototype's design will consider the robotic arm's specifications and the scissor lift (Genie 2646). A 6-DOF IMU sensor module will be utilized to capture the orientation of the platform. The sensor fusion technique will be applied using the gyroscope and accelerometer inside the sensor module to attain accurate orientation. Moreover, a filtering technique will be applied to suppress the sensor noise and gyro-drift. Lastly, three continuous rotation servos will be utilized with a PID control to derive an accurate and prompt response to the platform.

1.3. Report Outline

The report's first chapter covers the project's introduction, aims, and scope. The second chapter will cover the literature reviews done prior to the implementation. The literature review will cover papers on the topics related to the implementation of the prototype. The third chapter will cover the methodologies of the prototype, especially the mathematical equations used for implementing the design and control system of the prototype. The fourth chapter will cover the control systems' implementation and the prototype's design based on the methodologies. The fifth chapter will cover the results and discussions regarding the experimented data of the prototype. The sixth chapter will cover the applications of the Auto-Balancing Platform on construction applications. The seventh chapter will cover the conclusions and the future development of the prototype. The last chapter will provide a brief overview regarding the future work of the following project.

2. LITERATURE REVIEW

Prior to the implementation, delving into the relevant methodologies regarding the control systems and design is a significant challenge throughout the project. Thus, relevant papers regarding the implementation were closely examined for brainstorming for implementing the Auto-Balancing Platform prototype.

2.1. Auto-Balancing Platform

The Auto-Balancing System is to balance the system itself regardless of the given environment. To do so, the system is designed as a closed-loop control system, in which the feedback is attained from sensors, usually the Inertial Measurement Unit. Then, the system compensates for the obtained angular deviations using motors or actuators from the attained feedback. There are various ways to achieve the set orientation of the system. Depending on the carrying load or the required workspace of the robot, the Auto-Balancing System could be designed as a serial or parallel manipulator. Parallel manipulator was first introduced in 1965 by Stewart Gough. Bhaskar and Mruthyunjaya proposed a review of the Stewart Platform, a parallel manipulator, in 1998 [1]. The platform developed by Stewart was a 6-DOF platform utilized as a flight simulator, as shown in Figure 1. The serial manipulator holds the advantage of its large workspace and maneuverability. However, its cantilever-like structure limits the load-carrying capacity since the links tend to bend on high loads. On the other hand, parallel manipulators hold an advantage in load-carrying capacity, dynamic performance, and precise positioning. Moreover, the inverse kinematic solutions of the parallel manipulators can be expressed through the closed-form solutions, which facilitates a faster control process compared to the serial manipulators. Therefore, by accounting for the Stewart Platform's advantage, the Auto-Balancing platform's primary design will be similar to the Stewart Platform.

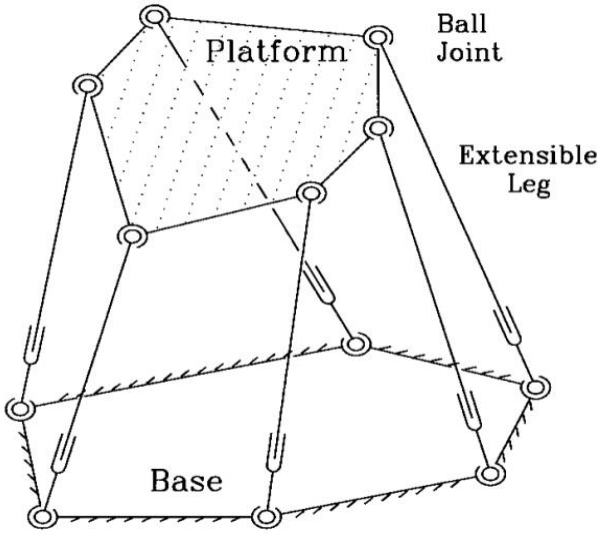


Figure 1: 6-DOF Stewart Platform [1]

2.2. The IMU Sensor

The Inertial Measurement Unit is often manufactured as a Micro-Electromechanical system that consists of an accelerometer and gyroscope, with a magnetometer depending on the sensor module. The accelerometer is responsible for the linear acceleration measurements, and the gyroscope is responsible for the angular velocity measurements on each x, y, and z-axis. Using each sensor, it is possible to measure the orientation in roll, pitch, and yaw.

The basic working principles of the accelerometer and gyroscopes are as follows [2]: The typical MEMS accelerometer works as the spring-mass system. It measures the changes in distance and the frequency of the vibrating element, the moving mass. The changes in distance between the moving mass and the fixed electrode induce a change in capacitance which is then to be measured through ADC. The acceleration is then calculated based on the rate of change in capacitance. The MEMS gyroscope works by the Coriolis Effect. The Coriolis Effect states that when a mass moves with a velocity over an angular motion, a force is generated, which causes a change in the distance of the mass [3]. Thus, when rotation is applied, the difference in capacitance is measured, which is converted into the rotation rate.

There are a few things to consider when determining which IMU module to use to increase the measurements' accuracy and reliability [2]. The bandwidth of the sensor module determines the range of frequency the sensor can measure. Thus, IMUs with higher bandwidth can capture higher precision in the motion changes. Sensitivity refers to the sensor's ability to capture the minimal changes, for IMU, the minimal changes in motion and orientations. Therefore, such

systems which require a response to such minimal changes need a sensor with high sensitivity. Lastly, the response frequency determines how quickly the sensor can respond to environmental changes. For high-speed applications, a sensor with a high response frequency is required.

2.3. Sensor Fusion Technique

The Auto-Balancing Platform will highly rely on the sensor readings. Therefore, it is significant to examine the appropriate sensor fusion technique closely. In 2021, Anh and Huang proposed a 3-DOF Stewart Auto-Balancing System for Low-Cost and High Reliable Embedded applications [4]. The proposed Auto-Balancing System by utilized a low-cost IMU, the MPU6050, to gain feedback on the system. The MPU6050 consists of two sensors inside: the accelerometer and the gyroscope. It is possible to obtain the orientation of the platform using one of the sensors inside the MPU6050 module through the followings: By the attained linear accelerations in each axis through accelerometer, the roll and pitch in radian can be obtained using trigonometric law of the acceleration vectors.

$$roll_{acc} = \sin^{-1}\left(\frac{a_y}{a_z}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$pitch_{acc} = \tan^{-1}\left(\frac{-a_x}{(a_y)^2+(a_x)^2}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where $roll_{acc}$ And $pitch_{acc}$ Are the calculated orientations through the data obtained by accelerometer and a_x, a_y, and a_z are the gravitational accelerations in each x, y, and z axis, respectively. Moreover, using the gyroscope inside MPU6050, which outputs the angular rate, the platform's orientation could be derived by integrating the angular rate over time.

$$roll_{gyro} = \int_0^t gyro_x(t)dt + bias_{roll} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$pitch_{gyro} = \int_0^t gyro_y(t)dt + bias_{pitch} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

where the $roll_{gyro}$ and $roll_{pitch}$ are the calculated orientations through the data obtained by gyroscope, $gyro_x$ and $gyro_y$ are the angular rate obtained from the gyroscope over time in each x and y axis respectively, and the $bias_{roll}$ and $bias_{pitch}$ are the constant bias from the gyroscope.

It is possible to obtain the orientation of the platform utilizing one of the sensors. However, the accelerometers are prone to noise, such as motor vibrations, which might output an inexact

measurement. Moreover, as the gyroscope outputs an angular rate with the gyro bias, integrating over time might induce the measurement to drift even though the gyroscope is placed stably. Thus, the orientation measurement from the accelerometer is favorable for long-term measurement; however, the orientation measurement from the gyroscope is favorable for short-term measurement. Therefore, fusing the data attained from the two sensors will yield a more reliable method. IMU sensor fusion is often implemented through the Complementary and Kalman Filter.

The complementary filter utilizes the calculated orientation through the gyroscope as a significant factor and employs the calculated orientation through the accelerometer to suppress gyro-drift. The mathematical model of a complementary filter is shown in equation 5 [5].

$$\theta = \alpha(\theta_{gyro}) + (1-\alpha)\theta_{acc} \quad \text{..... (5)}$$

where θ is the orientation, θ_{gyro} and θ_{acc} are the calculated orientation through gyroscope and accelerometer respectively, and the $\alpha(0 < \alpha < 1)$ is the filtering coefficient.

The Kalman Filter can be used to estimate the state of the dynamic system from noisy measurements. It recursively uses the redundant data to approximate the system states in the state space [6]. Therefore, the advantage of the Kalman Filter is that it considers both the uncertainty in the measurement and the model to yield a better estimate of the true state of the system. The filter updates the estimate of the system state recursively as the new measurements are attainable. The overall structure of the Kalman Filter algorithm is shown in Figure 2. The true state (x_k) and the measurement state (z_k) at time k of the Kalman Filter are as follows [5]:

$$x_k = F_k x_{k-1} + B_k u_k + \omega_k \quad \text{..... (6)}$$

$$z_k = H_k x_k + v_k \quad \text{..... (7)}$$

where F_k is the state transition model which shows how the system evolves overtime, B_k is the control-input model, u_k is the control input to be controlled, ω_k is the process noise which is the uncertainty in state transition model, H_k is the observation model which relates the true state to the measurement, and the v_k is the observation noise which is the uncertainty in the measurements.

Anh and Huang ran an experiment after applying different sensor fusion techniques, as shown in Figure 3 [2]. As the figure illustrates, both the complementary filter and Kalman filter show a significant improvement on suppressing the sensor noise. The complementary filter responds faster to the inputs as it requires less computation than the Kalman filter. In contrast,

the Kalman filter outputs more precise results than the complementary filter. Therefore, for the case of the Auto-Balancing Platform, the Kalman filter will be adopted to suppress sensor noise and gyro-drift for better precision of the overall system.

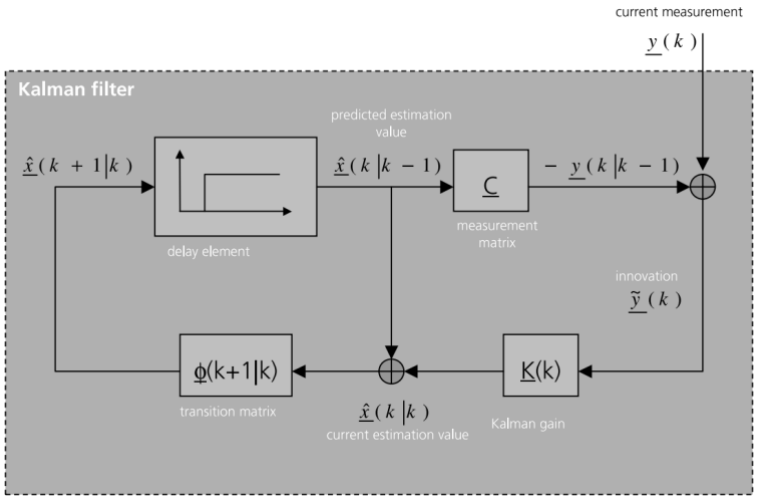


Figure 2: Kalman Filter Algorithm [5]

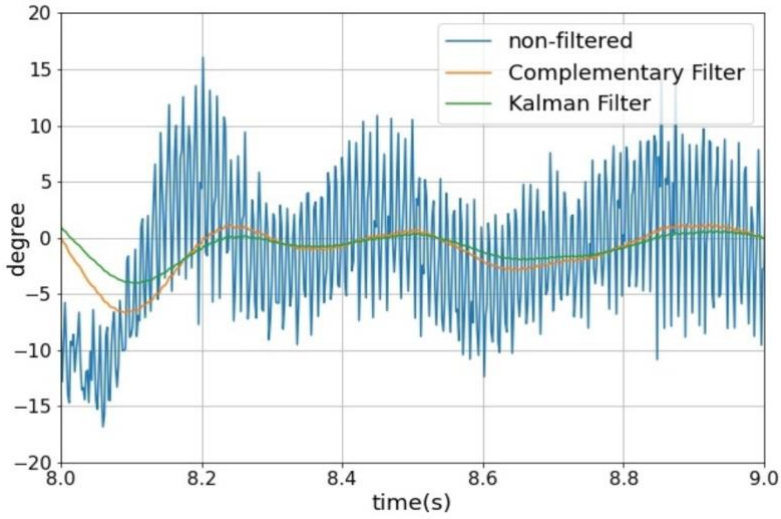


Figure 3: MPU6050 Dynamic Test with different Sensor Fusion Technique [2]

2.4. Control Strategies for Stewart Platform Manipulators

Depending on the applications, multiple ways exist to control the pose of the Stewart Platform-based Manipulators effectively. In 2022, Doan and Bui developed a Low-Cost Auto-Balancing System for Safety Mechanisms [7]. The developed system utilized Kalman Filter to suppress sensor noise and then applied Inverse Kinematics to control the pose of the system. Given the retrieved joint angles from Inverse Kinematics, PID control was utilized to control the motor to the setpoint. The implemented Inverse Kinematics solution adopted the Denavit-Hartenberg

parameters. The system was then evaluated with a load of 200 grams with an angular deviation of ± 30 degrees. The system's response showed 500 ms, as shown in Figure 4, which is a reasonable result for most applications.

In 2009, a Model-based Control for 6-DOF Parallel Manipulator driven by a hydraulic actuator was proposed by Yang and Huang [8]. Model-based Control is one method to design a control system by closely examining the mathematical model of the system being controlled to generate control actions. The derived mathematical model of the control system accounted for the model's dynamic characteristics, as shown in Figure 5. After the derivation of the mathematical model, a simulation was held to verify the control system's performance. The simulation result revealed a superior control system performance with a steady state error asymptotically converging to zero. Moreover, the control system showed a robust performance in reducing the influence of platform load variety on the system.

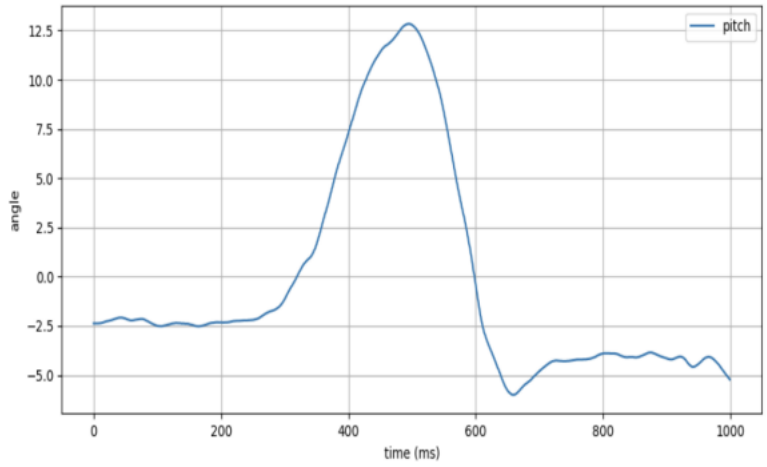


Figure 4: System Response Rate Result for Pitch by [7]

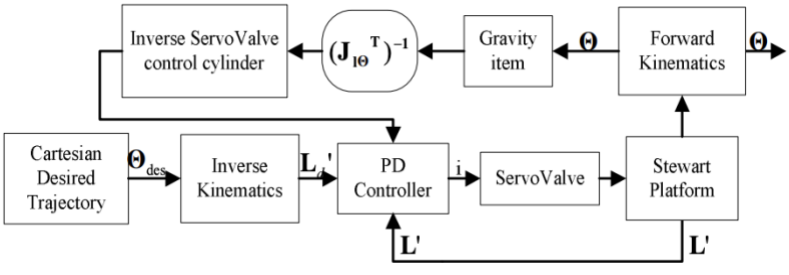


Figure 5: Model-based Control Schematic [8]

3. METHODOLOGY

The chapter includes the methodologies used to develop an Auto-Balancing Platform based on the brainstormed ideas from the literature review. It includes the derived mathematical model of the control system and prototype design.

The project was subdivided into four main sections: Hardware and Mechanical Design, Kalman Filter for IMU, Inverse Kinematics, and PID Control. The four main sections' main objectives were to opt for a high-performance Auto-Balancing Platform for Construction Applications. Thus, the following chapter will cover the methodology prior to implementation regarding each section.

3.1. Problem Statement

Before diving into the subsections of the project, an analysis of the problem statement will be given to further visualize the project. Figure 6 shows the diagram of Genie 2646 with the directions for clarification of the specifications. From the datasheet of the Genie 2646 [9], the maximum breakover angle is 14 degrees, while the maximum working angle is ± 3 and ± 1 degrees in α and β, respectively. Therefore, the objective of the Auto-Balancing Platform is to compensate for the maximum breakover angle of 14 degrees while traveling and the maximum working angle while the work is in progress. The length of the scissor lift is 2.26 m and 1.16 m in α and β directions, respectively. The design of the prototype will account for the dimensions of the Genie 2646.

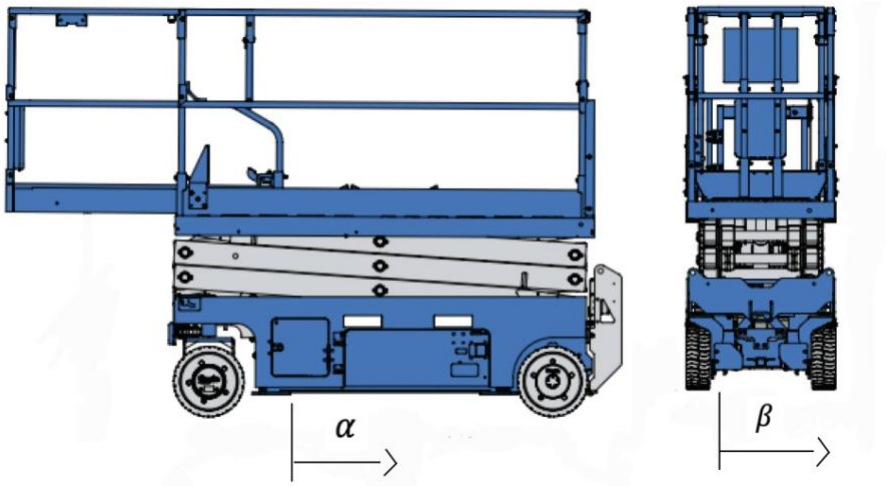


Figure 6: Genie 2646 Diagram [9]

3.2. Hardware and Mechanical Design

Hardware and Mechanical Design play a significant role in implementing the complete prototype. Appropriate selection of the hardware to be used and mechanical design will increase the overall stability and efficiency of the model. Hence, the following section will explain why the selected hardware and mechanical design is the best choice.

3.2.1. End-Effector Design

The end-effector to be controlled should be closely examined for its control efficiency and mechanical stability when the load is applied. Traditional Auto-Balancing Platform utilizes a square-shaped design due to its better load distribution and ease of fabrication. However, in contrast to the square-shaped design, the equilateral triangle possesses an advantage in control efficiency over the square-shaped End-Effectors. Assuming the same perimeter and material for both the equilateral triangle and square for End-Effector, the equilateral triangle-shaped End-Effector has a smaller surface area and mass than the square-shaped End-Effector. This reduced mass reduces inertia, making it easier for the actuators to control and maneuver the End-Effector.

Moreover, 3-DOF could be achieved using three actuators for the triangular-shaped End-Effector, whereas utilizing a square-shaped End-Effector might create singularity when three actuators are used, as a square presents four corners. Therefore, if the square-shaped End-Effector should be used, the manipulator should utilize an additional actuator to compensate for the singularity. For the above reasons, in the following project, the equilateral triangle-shaped design of the End-Effector will be adopted due to its superior control efficiency.

For the actual model, the following calculations were performed to acquire higher mechanical stability under load of 300 kg:

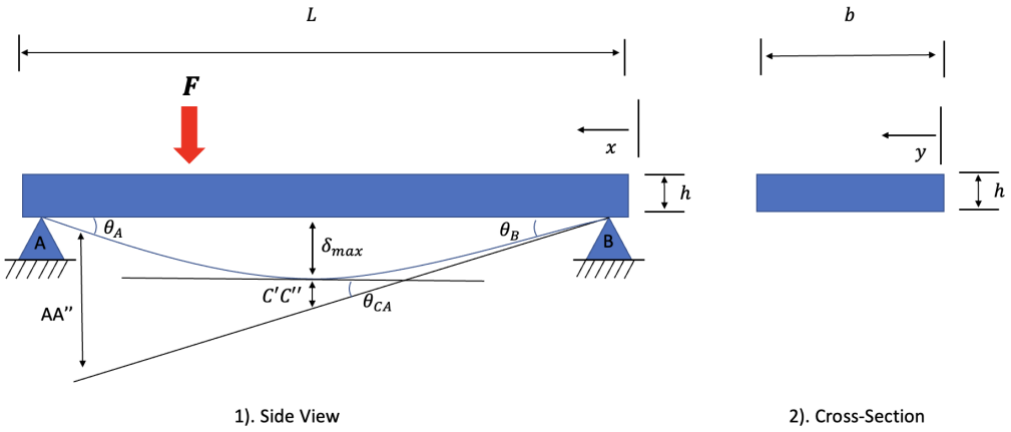


Figure 7: Illustration of Deflection Analysis of the Beam

Figure 7 was drawn to aid visualization of the assumptions taken to the calculations. The moment area method will be used to estimate the thickness, as the cross-section will vary across the height of the equilateral triangle. To obtain an estimate of the thickness prior to simulations, the End-Effector will be considered as a beam with non-uniform cross-sections hinged at both ends.

Before the calculations, objectives were set to ensure the End-Effector design was reliable. The maximum deflection, δ_{max}, was set to 3 mm. The applied force, F, to the actual model in the following project will be 2943 N, which is the estimated mass of the construction robotic arm multiplied by gravity. Moreover, accounting the length and width of the Genie 2646, the actual model will have a length of 1.1 m on each side. Therefore, the height, L, of the of the triangle becomes 0.953 m. As, the actual model should endure a high load, stainless steel is chosen for its renowned durability and corrosion resistance. The elastic modulus, E, of the stainless steel is 190 GPa [10].

The required thickness can be obtained by examining the elastic curve as shown on Figure 7.

$$\delta_{max} = AA''\left(\frac{x}{0.953}\right) - C'C'' \quad \text{..... (8)}$$

where BB'' is the distance between the elastic curve to the tangent line with respect to A, and $C'C''$ is the distance between the elastic curve to the tangent line at the point where the maximum deflection occurs

Before deriving the equations through moment area method, Bending Moment Equations should be derived. As the load is assumed to be applied at the center of the triangle which is 0.32 m away from point A from Figure 7, the reaction forces at point A and point B are:

$$R_A = 1954.79 \text{ N}, R_B = 988.2 \text{ N} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (9)$$

By the obtained reaction forces, the equations of the bending moment were derived as:

For $x < 0.633$, the equation of the bending moment is:

$$M = 988.2x \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (10)$$

For $0.633 < x < 0.953$, the equation of the bending moment is:

$$M = -1954.8x + 1862.919 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

By the obtained equation 10 and 11, AA'' can be determined through moment area method:

$$AA'' = \frac{1}{EI}\left(\int_0^{0.633} 988.2x(0.953 - x)dx + \int_{0.633}^{0.953} (-1954.8x + 1862.919)(0.953 - x)dx\right)$$

$$= \frac{1}{EI}(126.464) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (12)$$

Since the point of maximum deflection occurs at the point where slope of the elastic curve is zero, the following equation can be derived:

$$\theta_B = \frac{AA''}{L} = \theta_{CA} = \frac{988.2x^2}{2 \times EI} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (13)$$

By substituting equation 12 to 13 and by solving equation 13 for x, the point of maximum deflection can be obtained.

By the obtained point of maximum deflection, the $C'C''$ could be derived through moment area method:

$$C'C'' = \frac{1}{EI}\left(\int_0^{0.518} 988.2x(0.518 - x)dx = \frac{22.89}{EI}\right. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (14)$$

To solve equation 8, the area moment of inertia should be defined. In the following case, as the cross-section is non-uniform, the base which varies along the height of the triangle is defined as:

$$b = b_0\left(\frac{x}{L}\right) = 1.1\left(\frac{x}{0.953}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (15)$$

By substituting equation 14 to area moment of inertia, the area moment of inertia for rectangular cross-section becomes:

$$I = \frac{bh^3}{12} = 0.0962xh^3 \quad \text{……..…….…… (16)}$$

Finally, by substituting equation 16 and $E = 190$ GPa to equation 12 and 14 and solving equation 8 yields:

$$\delta_{max} = 0.003 = \frac{1}{190\times10^9\times0.0962xh^3}(126.464)(\frac{x}{0.953}) - \frac{22.89}{190\times10^9\times0.0962xh^3} \quad \text{……. (17)}$$

By solving equation 17 for h, the thickness is obtained as 15 mm. Based on the obtained calculation, an End-Effector, a 0.015 m thickness built with stainless steel ($E = 195$ GPa), will be examined through simulations on results and discussions section for further verifications.

For the prototype of the platform, a size of 1/5th of the actual model will be implemented. Therefore, the side lengths of the platform become 220 mm on each side.

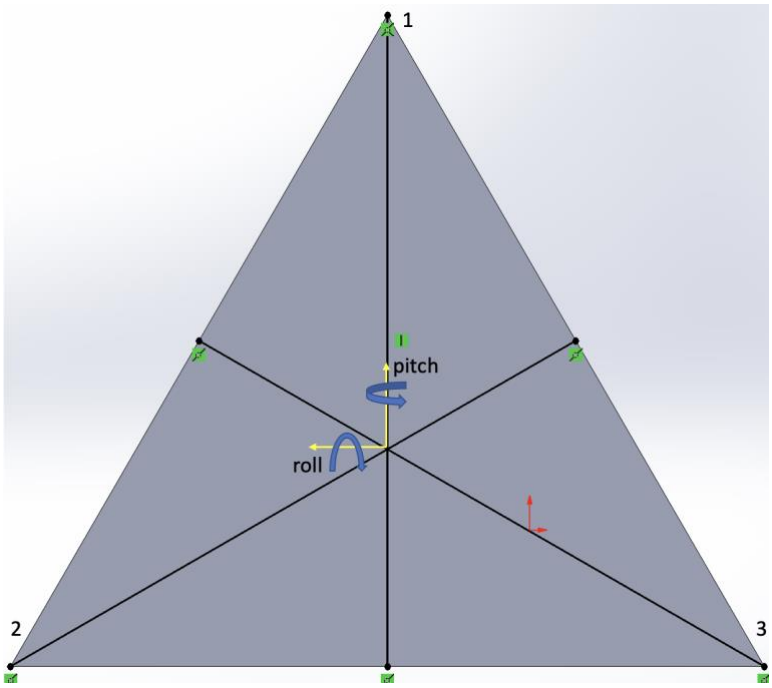


Figure 8: End-Effector Design

3.2.2. Rack and Pinion Design

As the actual model should carry high loads, the following prototype will use linear actuators attached to each vertex of the End-Effector. The electric cylinders offered in the market for high-load applications are mostly built with lead-screw or ball-screw mechanisms due to their simplicity in design and cost. However, lead-screw or ball-screw mechanisms offer a limited speed compared to other mechanisms. Therefore, the rack and pinion mechanism will be implemented for the electric cylinder design to compensate for the slow speed of such mechanisms. The rack and pinion mechanism are suitable for applications requiring high speeds and load since it gives a higher efficiency in converting the input power to the rotary motions. Moreover, in the case of the Auto-Balancing Platform, which opts for precise positioning, the rack and pinion mechanism is a suitable choice for its high accuracy.

To begin with the design, the length of the rack should account that the platform should be able to compensate 15 degrees both in roll and pitch. The maximum inclination distance of the actuator to suppress the angular deviation occurs when the platform experiences a positive angular deviation with respect to the *x*-axis of the central frame based on Figure 8. Therefore, using the trigonometric rule, the maximum stroke length of the rack becomes:

$$Stroke_{max} = 127 \tan 15 = 34.03 \text{ mm} \quad \text{..... (18)}$$

Accounting for the safety margin and the size of the pinion so that the rack does not slip out of the linear guide, the length of the rack was set to 75 mm.

The approximate pitch diameter of the pinion (spur gear) of one revolution for 75 mm rack movement was derived as follows:

$$P_{d,approx} = \frac{75}{\pi} = 23.87 \text{ mm} \quad \text{..... (19)}$$

Using the approximated pitch diameter, the number of teeth of pinion was identified by multiplying it with the gear module, GM. The gear module for the prototype was set to 0.95, considering the cost and compactness of the pinion. However, a larger gear module is required for the actual model to increase load capacity and efficiency.

$$z = P_{d,approx} \times GM = 23.87 \times 0.95 \approx 25 \text{ teeth} \quad \text{..... (20)}$$

From the obtained number of teeth of the pinion, the actual pitch diameter becomes:

$$P_d = z \times GM = 25 \times 0.95 = 23.75 \text{ mm} \quad \text{..... (21)}$$

Based on Figure 9, the mounting distance was set to 23.875 mm, considering the size of the motor. Therefore, by subtracting the pitch radius from the mounting distance, A, the height of the rack was chosen as 12 mm. The pressure angle for both rack and pinion was selected as 20 degrees, providing a good balance between strength and smoothness of operation for spur gears. The face width of the pinion will be 8 mm; however, the face width of the rack will be 12 mm to give a tolerance in case of pinion derails from the linear guide.

Table 1: The specifications for the Rack and Pinion design

	Rack	Pinion
Gear Module	0.95 mm	0.95 mm
Number of Teeth	-	25
Pitch Diameter	-	23.75 mm
Pressure Angle	20 degrees	20 degrees
Face Width	12 mm	8 mm

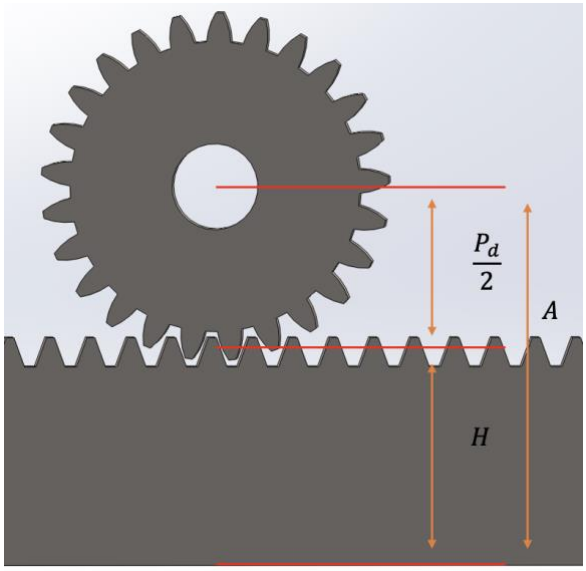


Figure 9: Rack and Pinion Illustration

3.2.3. Microprocessor Unit

In order to control the overall system, a central control unit should be used to obtain input from the sensor readings and execute an output to the motor through a relevant algorithm. Therefore, a Microprocessor Unit was adopted instead of the Microcontroller (MCU) in the following project. Primarily, the Microprocessor Unit can handle more complex tasks compared to the MCUs. The Microprocessor Unit performs better for systems with real-time calculations and

feedback control, such as the Auto-Balancing Platform. Moreover, the Microprocessor Unit enables a better development environment than the MCUs, since it runs with a more advanced operating system. Lastly, the Microprocessor Unit enables better accessibility. The Microprocessor Unit offers built-in Bluetooth, WIFI, and ethernet, allowing users to control and monitor the system remotely.

There exist multiples of MPUs offered in the market. However, the Raspberry Pi Model 4B is suitable for the following application concerning cost and computational needs. It provides high performance in terms of computations, with a large community of developers engaged in it. Moreover, it also provides enough GPIO pins for communicating with the peripheral devices.

3.2.4. Motor

The actual model of the Auto-Balancing Platform will carry a construction robotic arm of about 300 kg. Therefore, enough torque is required to balance the platform with a load of 300 kg. Moreover, to quickly adapt to the uneven grounds of the construction sites, the Auto-Balancing Platform should also utilize a motor with a reasonable speed. There are a few types of motors for high-load applications, such as AC and DC motors. AC motors are frequently used in industrial applications such as conveyor belts or compressors. The AC motors are suitable for applications requiring high starting torque to handle heavy loads. DC motors are also frequently used in high-load applications, especially for applications requiring variable speed control. In the case of the Auto-Balancing Platform, variable speed control is required to avoid overshoot of the control system; therefore, DC motors will be utilized on the actual model. The continuous rotational servos will be utilized for the prototype due to their similarity with the DC motors. The continuous rotational servos behave similarly to the DC motors since both rotate continuously without stopping at specific angles. The only difference is that the continuous rotational servos have built-in circuitry to change the direction of the motor. In contrast, DC motors require external circuitry, like motor drivers, to change the direction of the rotation. As the prototype will be carrying a 3 kg mass on top, the required torque of the motor should be:

$$T = F\left(\frac{P_d}{2}\right) = (1 \times 9.81) \times \left(\frac{0.02375}{2}\right) = 0.12 \text{ Nm} \quad \text{..... (22)}$$

where F indicates the force, each actuator should exert to compensate 3 kg, and P_d indicates the pitch diameter of the gear from equation 21.

Therefore, the FS5106r, a continuous rotational servo with a torque of 0.572 Nm from FEETECH, was selected for the prototype. Moreover, PCA9685, a motor driver, will be used to communicate with the Raspberry Pi through I^2C communications.

3.2.5. IMU Sensor

To control the continuously rotating servo upon the supplied voltage, there should be incoming feedback to decide the motor's direction and rotation speed. Therefore, for the Auto-Balancing Platform, the IMU sensor will be utilized for the system feedback. Regarding the selection of the IMU sensor, there are a few things to consider, as discussed in Chapter 2: bandwidth, sensitivity, and response frequency. By closely examining the factors mentioned above, MPU6050 was chosen. The MPU6050 has a programmable digital low-pass filter in which the unwanted-high frequency noise could be removed. Moreover, the MPU6050 provides a user-selectable-full-scale range for sensitivity, where the maximum full-scale range, according to the MPU6050 data sheet, of the accelerometer is ± 16 g and ± 2000 °/s, making it a suitable choice for the platform. The response frequency of the MPU6050 depends on the user-selected output data rate. The maximum output data rate of the accelerometer and gyroscope inside the MPU6050 is 8 kHz and 1 kHz, respectively, which is sufficient for the platform's needs. As the Auto-Balancing platform needs to achieve high accuracy and response time to maintain itself balanced, it needs prompt and accurate feedback. The acquired values regarding the MPU6050 from the MPU6050 data sheet [11] indicate the potential for achieving accurate and prompt feedback to the system.

Though the MPU6050 gives a high performance in accuracy, typical low-cost MEMS IMUs give inaccurate measurements due to manufacturing variations. Therefore, calibrations should be performed before the operation, as the Auto-Balancing Platform solely relies on the incoming feedback from the MPU6050. The MPU6050 consists of two tiny sensors: an accelerometer and a gyroscope. As the following project utilizes an accelerometer and gyroscope for sensor fusion, both tiny sensors inside MPU6050 must be calibrated. The gyroscope calibration is relatively simple compared to the accelerometer since the expected output of the measurement under steady conditions is 0 °/s. Therefore, the gyro bias will be average measured readings when placed stably from the gyroscope over time. The actual angular velocity could be obtained by subtracting the gyro bias from the measured readings.

$$gyro_{actual} = gyro_{measured} - gyro_{bias} \quad \text{..... (23)}$$

For the accelerometer, the calibration steps were performed by taking advantage of gravitational acceleration by placing the MPU6050 stably. Three positions for each axis of the accelerometer were accounted for, acquiring the measurement error, as the Figure 10. For example, when the z-axis of the accelerometer is placed upward against gravity, the accelerometer in the z-direction should output 1 g, which equals 9.81 m/s^2. Similarly, when the z-axis is placed perpendicular to gravity and downwards against gravity, the output of the accelerometer in the z-direction should output 0 g and -1 g, respectively. Thus, the offset between the measured accelerations from the acceleration and the actual acceleration was defined, as shown in Figure 11. By the obtained offsets for each data for three different positions, the line of best fit was drawn through the following equation:

$$y = mx + b \quad \text{..... (24)}$$

where the slope and the x-intercept of the line of best fit is defined as,

$$m = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \quad \text{..... (25)}$$

$$b = \frac{\sum_{i=1}^{n} y_i - m\sum_{i=1}^{n} x_i}{n} \quad \text{..... (26)}$$

The line of best fit of the offset between the actual and measured was drawn through the least square method, as shown in equations 24, 25, and 26. For equations 25 and 26, n represents the number of samples collected, x represents the actual acceleration (9.81 m/s^2, 0 m/s^2, -9.81 m/s^2) for each of the position as shown in Figure 10, and y represents the offset between the measured and actual acceleration. After obtaining m and b from equations 25 and 26, using equation 24, the offset between any randomly measured data and the actual acceleration was predicted. The output of equation 24 was then subtracted from the measured data to attain the actual acceleration. The results obtained regarding the calibration procedures of the MPU6050 will be further discussed in Chapter 5.

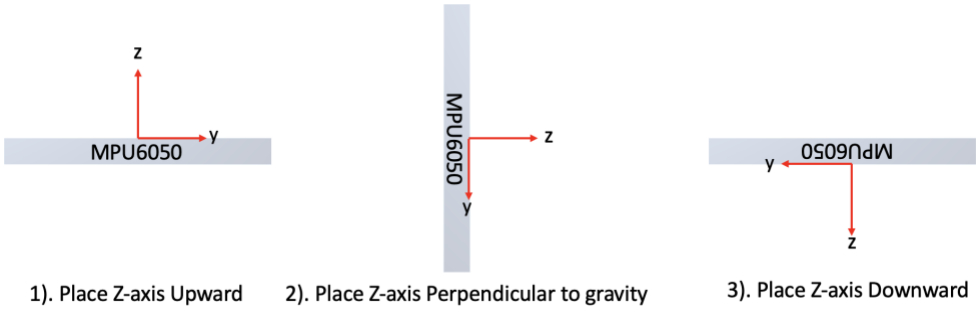


Figure 10: Accelerometer Calibration Illustration for z-axis

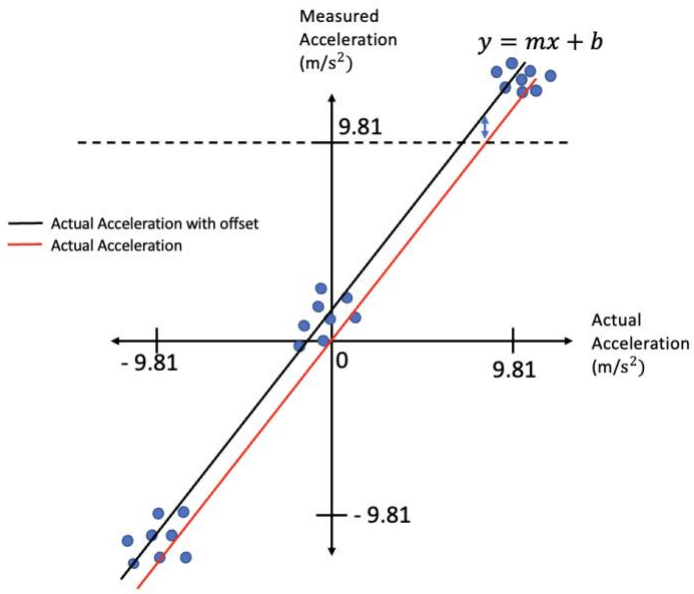


Figure 11: Illustration of Offset between Measured and Actual Accelerations

3.3. Kalman Filter

The following section will cover about the sensor fusion methodology implemented through the Kalman Filter for the Auto-Balancing Platform. The main objective of the sensor fusion technique is to acquire accurate feedback for the control system. Therefore, the data obtained from both the accelerometer and gyroscope inside the MPU6050 will be fused to opt for the enhanced accuracy of the platform's angular deviations. Kalman Filter is a very powerful tool that recursively updates the estimate of the uncertain or noisy data by the given measurement over time. The following section will be divided into two sub-sections, which are the prediction and update stage of the Kalman Filter. The derivations regarding the Kalman Filter for IMU sensors will follow the following equations for the true state, x_k, and the measured state of the true state, z_k, from literature review [5]:

$$x_k = F_k x_{k-1} + B_k u_k + \omega_k \quad \text{..... (27)}$$

$$z_k = H_k x_k + v_k \quad \text{..... (28)}$$

3.3.1. Prediction Stage

The prediction stage of the Kalman Filter will account for the measurement from the gyroscope to estimate the current system states. Therefore, the Kalman Filter will use the data obtained from gyroscope to predict the probability distribution over the current system's state. The state vector of the current state, x_k, from equation 27 should be defined to derive Kalman Filter for specific applications. The state vector is the variable indicating the current state of the system. For the case of the Auto-Balancing Platform, the orientation, θ, which is in degrees, should be defined. Moreover, the bias of the gyroscope, $\dot{\theta}_{bias}$, will be defined to ensure no drift of the obtained orientation over time. Therefore, the state vector for the Auto-Balancing Platform becomes:

$$x_k = \begin{pmatrix} \theta \\ \dot{\theta}_{bias} \end{pmatrix} \quad \text{..... (29)}$$

Moreover, the variables in Equation 27 should be defined to predict the system's current state. The state transition model, F_k, which shows how the system evolves with time, is defined as:

$$F_k = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \quad \text{..... (30)}$$

Since the control input is the angular velocity obtained from the gyroscope, and to convert the angular velocity to angle, the control input matrix, B_k, and control input, u_k, are defined as the following:

$$B_k = \begin{pmatrix} \Delta t \\ 0 \end{pmatrix}, \; u_k = \dot{\theta}_k \quad \text{..... (31)}$$

The process noise, ω_k, a Multivariate Gaussian distribution with a zero mean and covariance Q_k at time *k*, indicates the certainty of the dynamic model itself. The covariance matrix of the process is expressed as:

$$Q_k = \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_{bias}} \end{pmatrix} \Delta t \quad \text{..... (32)}$$

As Q_k is the covariance at time *k*, the covariance matrix was expressed in terms of the variance of the state vectors multiplied by the time difference.

After the variables inside Equation 27 are defined, the predicted state of the system at time *k* based on time *k*-1, $\hat{x}_{k|k-1}$, will be defined as the following equation.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1} + B_k \dot{\theta}_k \quad \text{..... (29)}$$

By substituting the defined variable and solving the equation, Equation 29 becomes:

$$\hat{x}_{k|k-1} = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta \\ \dot{\theta}_{bias} \end{pmatrix}_{k-1} + \begin{pmatrix} \Delta t \\ 0 \end{pmatrix} \dot{\theta}_k = \begin{pmatrix} \theta + (\dot{\theta}_k - \dot{\theta}_{bias})\Delta t \\ \dot{\theta}_{bias} \end{pmatrix} \text{ ..... (30)}$$

Then, the Error Covariance matrix at time *k*, P_k, will be estimated, which indicates the certainty of the current predicted state.

$$P_{k|k-1} = F_k P_{k-1} {F_k}^T + Q_k = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\Delta t & 1 \end{pmatrix} + \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_{bias}} \end{pmatrix} \Delta t$$

$$= \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix}_{k|k-1} = \begin{pmatrix} (P_{00} - P_{10}\Delta t) - (P_{01} - P_{11})\Delta t + Q_\theta \Delta t & P_{01} - P_{11}\Delta t \\ P_{10} - P_{11}\Delta t & P_{11} + \Delta t Q_{\dot{\theta}_{bias}} \end{pmatrix} \text{ .. (31)}$$

After the prediction stage, the predicted State Vector and Error Covariance Matrix will be updated and corrected, which will be further elaborated in the next section.

3.3.2. Update Stage

In the update stage, Kalman Filter will use the measurement of the system's state to update the probability distribution over the system's state. Prior to updating the predicted State Vector and Error Covariance Matrix, variables in Equation 28 will be defined. As the predicted state was based on the gyroscope, the measured state, z_k, will be from the calculated orientation through the accelerometer. The calculation of orientation using the data obtained from the accelerometer will follow Equations 1 and 2 from the literature review [4]. The H_k in Equation 28 is an observation model to map the measured state, z_k, to the true state, x_k. Therefore, considering the units of the elements of the true state and the measured state, the observation model, H_k, is defined as:

$$H_k = \begin{pmatrix} 1 & 0 \end{pmatrix} \text{ ..... (32)}$$

The measurement noise, v_k, follows a Gaussian distribution with zero mean and variance of R.

To update the predicted state of the system, the difference between the measured state and the predicted state, y_k, was defined.

$$y_k = z_k - H_k \hat{x}_{k|k-1} = z_k - \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \theta_k \\ \dot{\theta}_{bias} \end{pmatrix}_k = z_k - \theta_k \text{ ..... (33)}$$

The next thing is to define the Innovation Covariance at time *k*, S_k, which indicates the certainty of the measurement based on the Error Covariance matrix and measurement noise variance, *R*.

$$S_k = H_k P_{k|k-1} {H_k}^T + R = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix}_k \begin{pmatrix} 1 \\ 0 \end{pmatrix} + R$$

$$= (P_{00})_k + R \quad \text{..... (34)}$$

The Kalman Gain at time *k*, K_k, can be defined by the obtained Innovation Covariance. The Kalman Gain is the weight given to the measurement and the current predicted state. In other words, the Kalman Gain can be seen as the indicator of certainty regarding the measurement.

$$K_k = \begin{pmatrix} K_0 \\ K_1 \end{pmatrix} = P_{k|k-1} {H_k}^T {S_k}^{-1} = \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix}_{k|k-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} {S_k}^{-1} = \begin{pmatrix} \frac{(P_{00})_k}{{S_k}^{-1}} \\ \frac{(P_{10})_k}{{S_k}^{-1}} \end{pmatrix} \text{.....} \quad \text{(35)}$$

Finally, the estimated state, $\hat{x}_{k|k-1}$, can be updated by the obtained Kalman gain and the difference between the measured and estimated state.

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k y_k = \begin{pmatrix} \theta_k \\ \dot{\theta}_{bias} \end{pmatrix}_k + \begin{pmatrix} K_0 y_k \\ K_1 y_k \end{pmatrix} \text{.....} \quad \text{(36)}$$

As the Kalman Filter recursively updates the predictions, the Error Covariance matrix should be updated based on the degree of the previous estimate correction.

$$P_k = (I - K_k H_k) P_{k|k-1} = \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} K_0 \\ K_1 \end{pmatrix} (1 \quad 0) \right) \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix}_{k|k-1}$$

$$= \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix}_{k|k-1} - \begin{pmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{pmatrix} \text{.....} \quad \text{(37)}$$

The elaborated steps regarding the Kalman Filter will be implemented and recursively executed through Python to acquire an accurate orientation of the platform.

The covariance of the process noise and the measurement noise variance were selected, as shown in Table 2. The selected parameters in Table 2 were based on the trustworthiness of the estimate and the measurement through an iterative process.

Table 2: Parameters for Kalman Filter

Q_θ	0.001
$Q_{\dot{\theta}_{bias}}$	0.003
R	0.03

3.4. Inverse Kinematics

Inverse Kinematics is an effective tool to figure out the joint position given the desired orientation of the platform. Specifically, Inverse Kinematic Solution defines the actuator length to reach the desired orientation of the platform. There are various ways to obtain Inverse Kinematics Solutions. However, considering the current design of the Auto-Balancing Platform, Inverse Kinematics Solutions will be obtained through the vector-loop equation, which is frequently utilized on parallel manipulators.

To begin with, the motion range or the degrees of freedom of the prototype should be clarified. The Auto-Balancing Platform is a 3-DOF platform that can translate in the *z*-axis and rotate in the *x* and *y*-axis with respect to the base frame. By the clarified motion range of the platform, the inputs to the Inverse Kinematics Equation to figure out the joint positions, specifically the stroke length of each actuator, could be determined as follows: the rotation in the *x*-axis, α, the rotation in the *y*-axis, β, the translation in the *z*-axis, *z*, where all motions are with respect to the base frame.

Based on Figure 12, accounting the vector subtraction rule, the vector-loop equation for joint i, $\vec{q}_i$, is derived as the following.

$$\vec{q}_i = \vec{B} + R\vec{C}_i - \vec{A}_i \quad \text{..... (38)}$$

where *R* represents the rotation matrix to transform the $\vec{C}_i$, which is expressed in terms of top frame, to the base frame $\vec{B}$ represents the displacement vector from the base frame to the top frame, $\vec{C}_i$ represents the displacement vector from the top frame to the vertex of the End-Effector, $(P_E)_i$, $\vec{A}_i$ represents the displacement vector from base of the actuator, $(P_b)_i$, to the vertex of the End-Effector, $(P_E)_i$

The rotation matrix, *R*, is expressed according to the platform rotation with respect to the *x* and *y* axis. The rotation matrix is to transform the coordinate frame of $\vec{C}_i$ to the base frame.

$$R = R_x R_y = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ \sin(\alpha)\sin(\beta) & \cos(\alpha) & -\sin(\alpha)\cos(\beta) \\ -\cos(\alpha)\sin(\beta) & \sin(\alpha) & \cos(\alpha)\cos(\beta) \end{pmatrix} \text{..... (39)}$$

The displacement vector $\vec{B}$ is expressed according to the translation of the platform with respect to the *z*-axis.

$$\vec{B} = \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix} \text{..... (40)}$$

The displacement vector $\vec{A}_i$ and $\vec{C}_i$, which can physically be computed, is different for each actuator, as the points, $(P_b)_i$ and $(P_E)_i$ have different coordinates with respect to the base and top frame, respectively. The displacement vectors $\vec{A}_i$ and $\vec{C}_i$, where $i \leq 3$, are as follows:

$$A_1 = \begin{pmatrix} 0 \\ 127 \\ 0 \end{pmatrix}, A_2 = \begin{pmatrix} 110 \\ -63.5 \\ 0 \end{pmatrix}, A_3 = \begin{pmatrix} -110 \\ -63.5 \\ 0 \end{pmatrix} \text{..... (41)}$$

$$C_1 = \begin{pmatrix} 0 \\ 127 \\ 0 \end{pmatrix}, C_2 = \begin{pmatrix} 110 \\ -63.5 \\ 0 \end{pmatrix}, C_3 = \begin{pmatrix} -110 \\ -63.5 \\ 0 \end{pmatrix} \text{..... (42)}$$

By substituting the defined vectors to Equation 38, the displacement vector, $\vec{q}_i$, becomes:

$$\vec{q}_1 = \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix} + R \begin{pmatrix} 0 \\ 127 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 127 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 127\cos(\alpha) - 127 \\ z + 127\sin(\alpha) \end{pmatrix} \text{..... (43)}$$

$$\vec{q}_2 = \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix} + R \begin{pmatrix} 110 \\ -63.5 \\ 0 \end{pmatrix} - \begin{pmatrix} 110 \\ -63.5 \\ 0 \end{pmatrix} = \begin{pmatrix} 110\cos(\beta) - 110 \\ 110\sin(\alpha)\sin(\beta) - 63.5\cos(\alpha) + 63.5 \\ z - 110\cos(\alpha)\sin(\beta) - 63.5\sin(\alpha) \end{pmatrix} \text{.. (44)}$$

$$\vec{q}_3 = \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix} + R \begin{pmatrix} -110 \\ -63.5 \\ 0 \end{pmatrix} - \begin{pmatrix} -110 \\ -63.5 \\ 0 \end{pmatrix} = \begin{pmatrix} -110\cos(\beta) + 110 \\ -110\sin(\alpha)\sin(\beta) - 63.5\cos(\alpha) + 63.5 \\ z + 110\cos(\alpha)\sin(\beta) - 63.5\sin(\alpha) \end{pmatrix} \text{. (45)}$$

To obtain the actuator length, the displacement vector, $\vec{q}_i$, should be quantified into the scalar value of distance.

$$q_i = \sqrt{\vec{q}_i{}^T \vec{q}_i} \text{..... (46)}$$

Solving equation 46 defines the actuator length, q_i, to reach the desired orientation. By closely analyzing Equations 43 to 45, judgments on which actuator should extend were identified based on the angular deviation of the Platform, as shown in Table 3. Moreover, although the prototype of the Platform did not have the physical encoder in the motor, Equation 46 was used to estimate the current position of the actuator in case the actuator must be retracted, to compensate for the angular deviations of the Platform.

Table 3: Actuator Extension Judgement based on Inverse Kinematics Solution

	Actuator 1	Actuator 2	Actuator 3
Positive Roll	+	---	---
Positive Pitch	---	---	+
Negative Roll	---	+	+
Negative Pitch	---	+	---

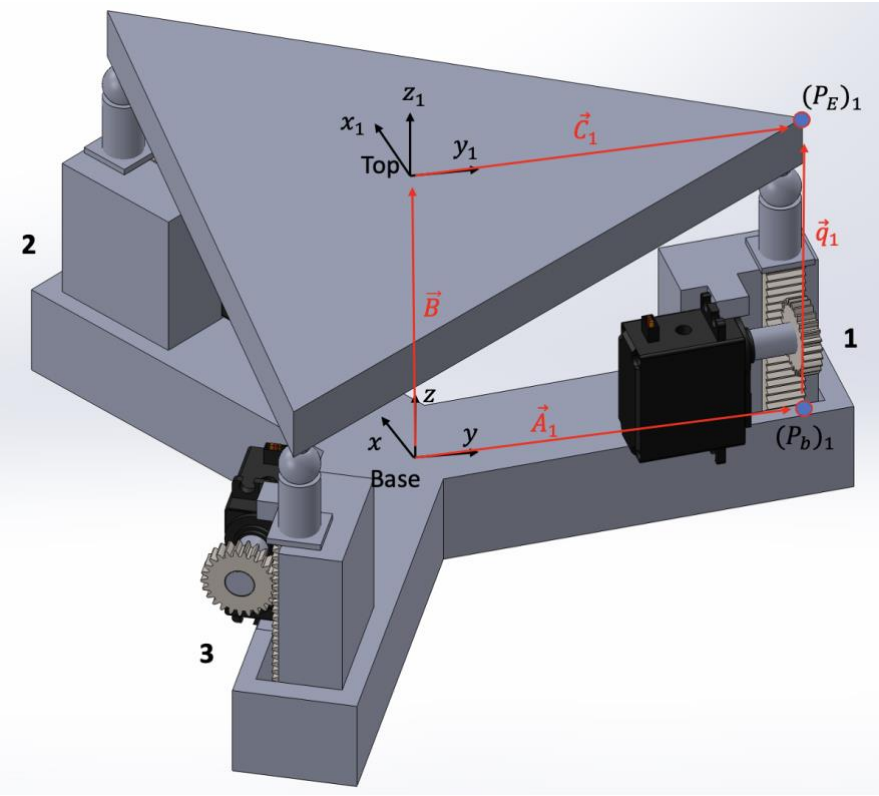


Figure 12: Illustration of the Inverse Kinematics Solution Formulation

3.5. PID Controller

PID controller is often used to control an uncertain system through feedback. The PID controller enhances the system's overall performance regarding stability, response, and accuracy. Moreover, given the feedback from sensors, the PID controller allows the system to track the setpoint by adjusting the signal getting into the plant. Therefore, the following section will provide the methodology for implementing the PID controller in the control system.

The PID controller for the following project will be implemented on a discrete-time domain. The reason for implementing the PID controller in the discrete-time domain is that it is more practical, especially for digital control systems, particularly the Raspberry Pi, operating on sampled data. Moreover, in the following project. the plant being controlled, the continuous rotation servo, are considered discrete systems since they operate on a sequence of discrete-time signals.

To convert the PID controller of the continuous-time frequency domain to a discrete-time frequency domain, the transfer function of the PID controller in the continuous-time frequency domain was derived based on Figure 13. Low-Pass Filter was added into the derivative term to avoid the derivative kick, which occurs when the system experiences a sudden change in setpoint or disturbance.

$$H(s) = \frac{u(s)}{e(s)} = K_p + \frac{K_i}{s} + \frac{K_d s}{s\tau+1} \quad \text{..... (47)}$$

The transfer function of the PID controller, $H(s)$, in the continuous-time frequency domain, was transformed into the discrete-time frequency domain through bilinear transform. The bilinear transform maps the *s*-plane, the frequency domain in continuous time, to *z*-plane, the discrete-time frequency domain. To map the *z*-plane into *s*-plane, the following equation from discrete-time approximation should be used:

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \quad \text{..... (48)}$$

where *z* is the time-shift operator which indicates the signal in terms of past and current values, and *T* represents the sampling time of discrete controller in seconds

By substituting Equation 48 to 47, the transfer function of the PID controller was obtained.

$$H(z) = \frac{u(z)}{e(z)} = K_p + \frac{K_i T(z+1)}{2(z-1)} + \frac{2K_d z - 2K_d}{2\tau(z-1)+T(z+1)} \quad \text{..... (49)}$$

Based on the derived transfer function of the PID controller in the discrete-time signal domain, for the ease of software implementation, each term in the PID controller was transformed into a difference equation, which is in the discrete-time domain, using inverse z-transform.

The difference equation for the proportional term was as follows:

$$P(k) = K_p e(k) \quad \text{(50)}$$

where *k* is the sample index which is used to denote the time samples

To derive the difference equation for integral term, each of the terms in the integral term was divided by the leading term.

$$\frac{I(z)}{e(z)} = \frac{K_i T + K_i T z^{-1}}{2 - 2z^{-1}} \quad \text{(51)}$$

By cross-multiplying Equation 51, the Equation 51 becomes:

$$I(z)(2 - 2z^{-1}) = e(z)(K_i T + K_i T z^{-1}) \quad \text{(52)}$$

As z^{-1} represents the delay of one sample period in the discrete-time frequency domain, the delay in terms of sample index is expressed as $k - 1$ in time domain. Therefore, Equation 52 in the discrete-time domain becomes:

$$I(k) = \frac{K_i T(e(k) + e(k-1))}{2} + I(k-1) \quad \text{(53)}$$

Similarly, the difference equation of derivative term, $D(k)$, can be derived using the same approach as the integral term.

$$D(k) = \frac{2K_d}{T + 2\tau}\big(e(k) - e(k-1)\big) - \frac{T - 2\tau}{T + 2\tau}(D(k-1)) \quad \text{(54)}$$

Combining the three terms in the difference equation yields the output of the PID controller in the discrete-time domain.

$$u(k) = P(k) + I(k) + D(k) \quad \text{(55)}$$

To avoid integrator wind-up, which saturates the controller output, anti-windup, particularly clamping, was accounted to prevent the integral term from reaching an excessively large value when the control output is at its maximum value. The PID gains were adjusted by recursively analyzing the system response to meet the desired performance, as shown in Table 4. All of the parameters were adjusted to output at a maximum of 1, the maximum value for the Pulse Width Modulation of the utilized motor for the prototype.

Table 4: PID Gains

K_p	0.2
K_i	0.02
K_d	0.001

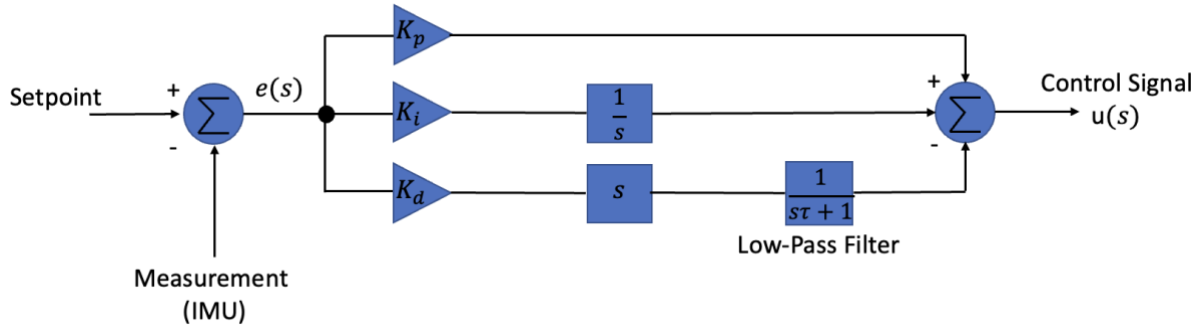


Figure 13: PID Block Diagram

4. IMPLEMENTATION

The following chapter includes the implementation of the elaborated methodology regarding the prototype of the Auto-Balancing Platform. The chapter will be divided into three main sections: Hardware Implementation, Mechanical Implementation, Software Implementation. Thus, the following chapter will provide the mechanical drawings and electrical circuit design regarding the prototype and flowchart regarding the methodologies.

4.1. Hardware Implementation

This section will provide an overview of the physical implementation of the hardware on the prototype. The list of hardware used to build the prototype of the Auto-Balancing Platform is as follows:

1. Raspberry Pi 4B: Microprocessor Unit which is utilized as the main controller for the prototype.
2. MPU6050: an IMU sensor which consists of accelerometer and gyroscope.
3. FS5106R: a Continuous Rotation Servo
4. PCA9685: a Motor Driver used to control the FS5106R and communicate with the Raspberry Pi

For the FS5106R and MPU6050 to communicate with the Raspberry Pi, I^2C communication was utilized. The main advantage of using I^2C communications is its simplicity and versatility. I^2C is a two-wire serial communication protocol that uses a single clock signal (SCL) to synchronize data transfer (SDA) between devices. It allows multiple peripherals to be connected to the same bus, and each peripheral can be identified by a unique address. The I^2C interface of the prototype with the address of the peripherals is shown in Figure 14. To connect master and the peripherals with I^2C communication, GPIO 2 (pin no. 3) and 3 (pin no. 5) of Raspberry Pi for Serial Data (SDL) and Serial Clock were used as the I^2C ports. To connect multiple peripherals, a breadboard was used for the prototype. The wiring through the breadboard was done as the schematics shown in Figure 15, where the green line indicates Serial Data Line and the yellow line indicates the Serial Clock Line.

To run the MPU6050 and PCA9685, 5 V and 3.3 V power supplies have been connected using Raspberry Pi (Pin No. 1, Pin No. 2), based on the datasheet of the peripherals. However, an external power supply to the motor driver was needed to run the motor. Therefore, an additional 5 V was supplied to the motor driver from the Raspberry Pi (Pin No. 4). Ground ports in

Raspberry Pi (Pin No.9, Pin No. 14) were utilized for the current to flow. Figure 15 shows the schematic of the Electrical Circuit design for the prototype.

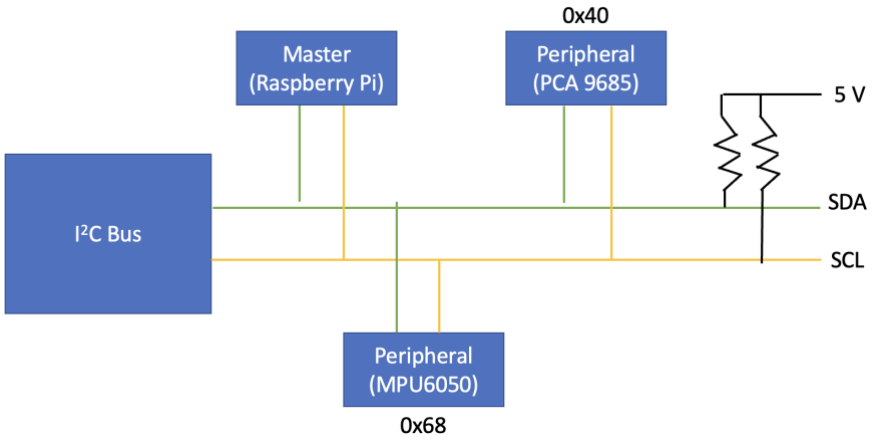


Figure 14: I2C Interface for the prototype

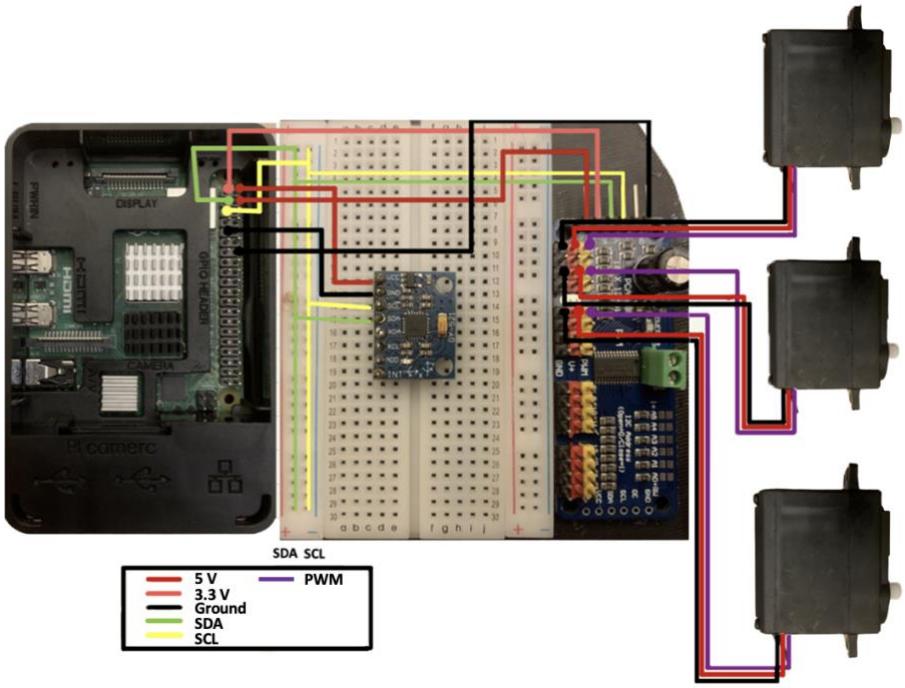


Figure 15: Schematic Electrical Circuit Design for the Prototype

4.2. Mechanical Implementation

This section will provide the Mechanical Implementation of the prototype based on the methodology. Therefore, the following section will provide relevant mechanical drawings with its mechanical parts.

As mentioned in Section 3.2.1, the prototype was scaled down to $1/5^{th}$ of the actual model. Therefore, each side length of the End-Effector for the prototype was designed as 220 mm on each side. Three M4 screw holes of 12 mm were drilled near each vertex of the platform for the universal joint to be attached. However, due to the cost, stainless steel was not used to implement the prototype, whereas the prototype utilized PLA, which is sufficient to endure a mass of 3 kg on top. The thickness of the platform was 17 mm as the M4 screw holes were drilled at 12 mm. The Base to tightly hold each actuator was designed. As each actuator was positioned perpendicular to each vertex of the End-Effector, the Base of the platform also had a triangular-like shape in which the actuators could be placed perpendicular to the End-Effector. At each end of the Base, the rectangular extruded cut was performed to place the actuators inside firmly. The dimensions of the rectangular extruded cut were 45 mm in length and 26.5 mm in width, based on the dimensions of the designed actuator. Figure 16 shows the 3D CAD Model of the End-Effector and the Base designed through SolidWorks.

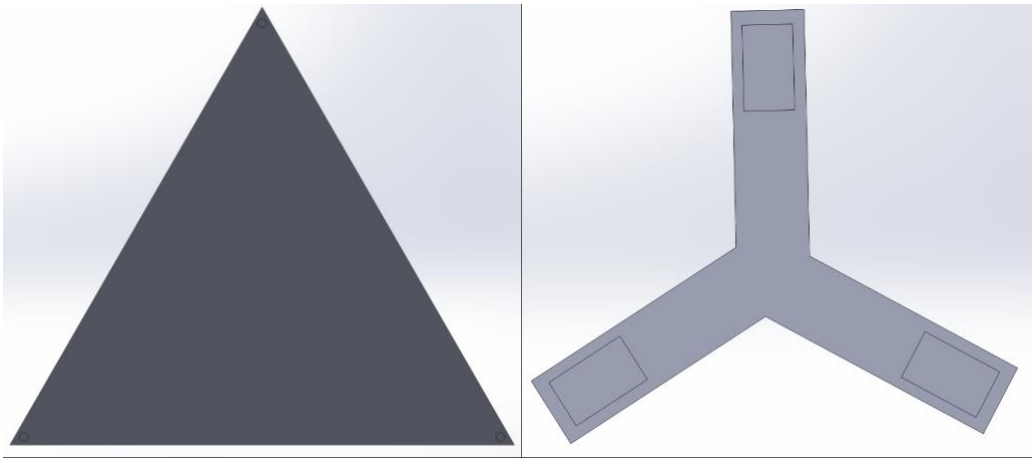


Figure 16: 3D CAD Model of the End-Effector (Left) and the Base (Right)

The rack and pinion mechanism was designed as shown in Table 1 in Chapter 3. In order to fully implement the rack and pinion mechanism, a linear guide was designed. The path where the rack would be placed to move in the linear guide was provided with a clearance of 1 mm to provide a smoother transition. The U-shaped extruded surface was designed to hold the motor firmly. The dimensions of the U-shaped extruded surface accounted for the rack's position and the pinion's pitch diameter so that the pinion and the rack could fit accurately to provide a smooth linear translation. The designed rack and pinion were sent to a third party for fabrication; however, the linear guide was fabricated using 3D printing. Detailed dimensions of the linear guide, rack, and pinion will be provided in Appendix.

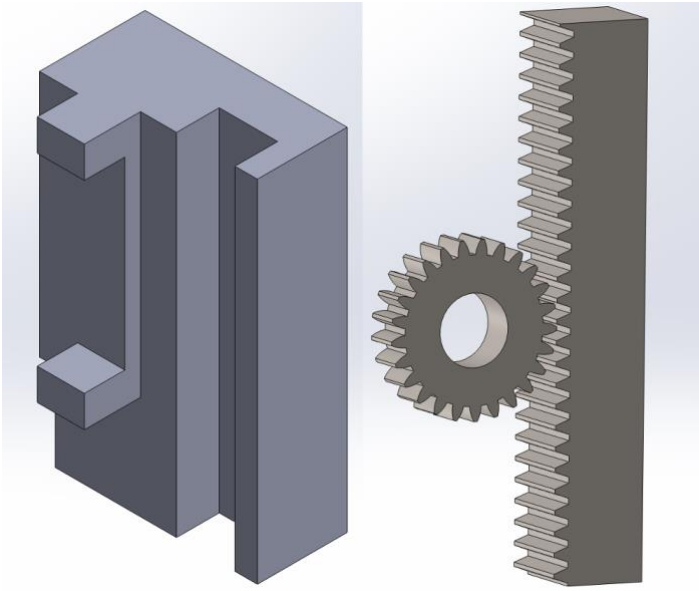


Figure 17: 3D CAD Model of the Linear Guide (Left) and Rack and Pinion (Right)

Various small parts were designed to assemble the final prototype. It is possible to attach the actuators themselves with the End-Effector. However, the problem arises when one of the actuators arises to compensate for the angular deviations of the platform since the other two actuators are placed stably, trying to form a perpendicular angle with the End-Effector. Therefore, a universal joint was utilized in the platform design to solve this issue. The universal joint used in the platform utilized a magnetic force of 50 N to couple the ball and socket of the joint. Thus, the magnetic force exerted is sufficient to hold the platform stably placed without tilting towards gravity, though the actuator translates.

Moreover, a coupling part was designed to couple the motor with the pinion. The circular extruded cut accounted for the diameter of the motor's rotating tip, which is 5.5 mm. The outer circle's diameter was designed as 10 mm based on the bore diameter of the pinion. The part was then 3D printed with high infill densities to avoid failure under operation.

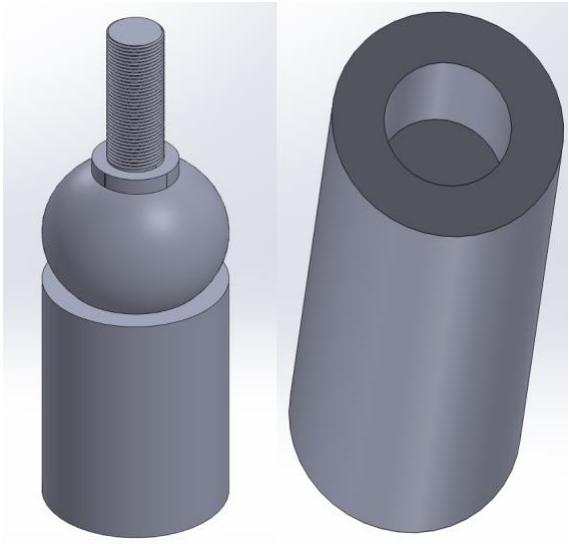


Figure 18: 3D CAD Model of Universal Joint (Left) and Pinion Coupling Part (Right)

After all of the parts were designed and fabricated, the final assembly of the prototype was built, as shown in Figure 19.

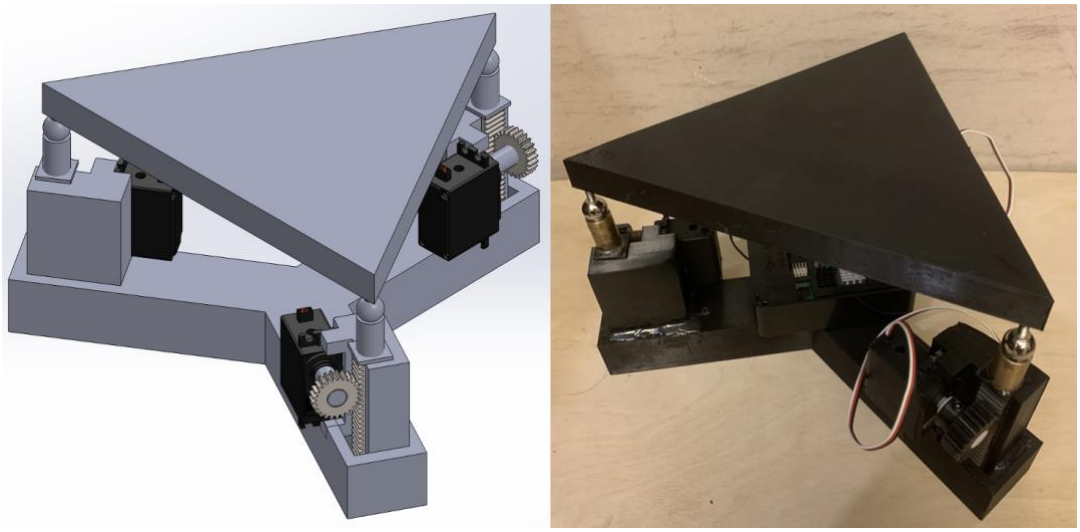


Figure 19: 3D CAD Model (Left) and the Actual Final Assembly (Right)

4.3. Software Implementation

For the prototype, as the main controller, the Raspberry Pi, runs through a Linux-based operating system with pre-installed Python, the elaborated methodology in Chapter 3 will be implemented through Python. Moreover, as Python is an object-oriented language, the program to implement the prototype will actively take advantage of using Object-Oriented Programming. The use of Object-Oriented Programming will ensure a breakdown of complex problems into smaller and more manageable pieces. The whole program was divided into four main sections: Main Program, Kalman Filter Class, and IMU Calibration Class. The following section of the report will elaborate on the structure of the code implemented, as Chapter 3 covers all the relevant methodologies. The codes written to implement the methodology of the prototype will be provided in the Appendix through the GitHub link.

The Kalman Filter Class was resolved using the provided equations in Chapter 3.3 to obtain the system's estimated state (angles). Kalman Filter was implemented as a Class to ensure efficient and independent updating of the variables, particularly the state vectors, within the object. The IMU Calibration was implemented through Class for ease of code management. Inside the IMU Calibration Class, three variables were declared for the gyroscope offsets, and six variables to represent the line of best fit were declared to estimate the offset of the accelerometer. The main program contained the following:

- the PID Controller Function to control the motor
- the function for obtaining the platform's orientation
- the main function of executing the program

Inside the main program, there were two PID Controller Functions to track the desired setpoint of the system in terms of roll and pitch. Each of the variables inside the functions was declared globally. Though using the global variables in software implementation is not recommended, the PID controller function requires access to variables that are updated in real-time during the program's execution. Therefore, rather than passing these variables as parameters to the function, using global variables makes it easier for the function to access and modify them. Therefore, in this case, the use of global variables in the PID controller function was a practical decision made to optimize program performance and simplify code management. The Main function follows the flowchart in Figure 20. The Exit condition main program was by KeyboardInterrupt; however, in the actual model, a push button will be used to interrupt and end the running program.

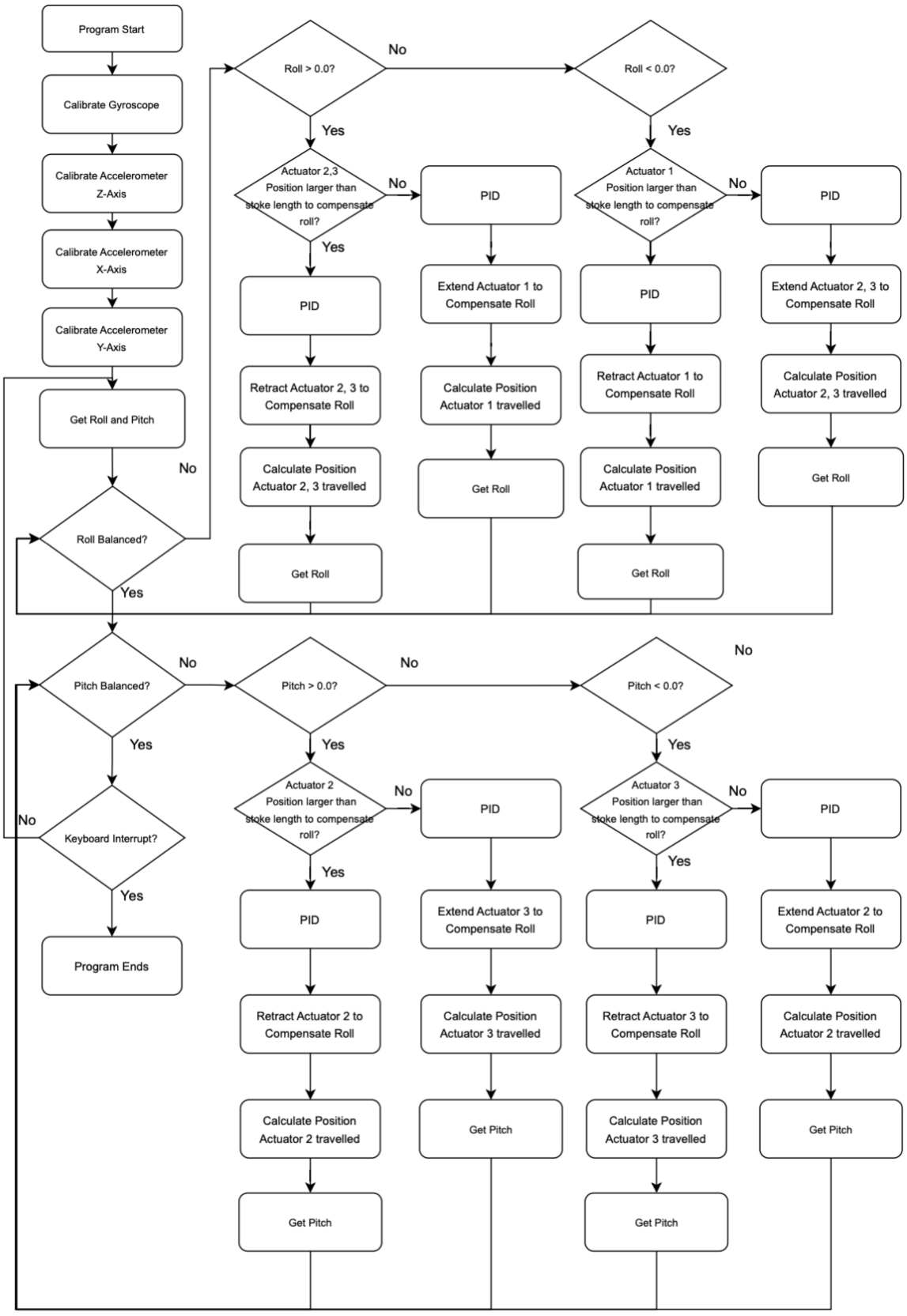


Figure 20: Flowchart of the Main Function

5. RESULTS AND DISCUSSIONS

The following chapter of the report will elaborate and discuss the obtained experimental results and Simulations regarding the Auto-Balancing Platform.

5.1. IMU Sensor Testing

The IMU sensor in the prototype plays a significant role. As the utilized motor was not a position servo, the only way to direct the motor's direction and speed was based on the feedback from the IMU Sensor. Therefore, the IMU sensor was closely examined to feed the system with an accurate orientation. All of the tests regarding the IMU Sensor were performed for 30 seconds.

Figure 21 shows the obtained angular velocity from the uncalibrated gyroscope under steady conditions. The blue, orange, and green lines indicate the angular velocity with respect to the *x*, *y*, and *z* axes, respectively. As the testing was performed under steady conditions, the gyroscope should have provided 0 deg/s for all axes. However, the obtained graph apparently indicates a gyro bias of specific amounts for each axis. Though the gyro bias is insignificant except for the angular velocity with respect to the *y*-axis, integrating angular velocity to obtain orientation in degrees over time will induce an immense drift value as time passes. Therefore, calibration to suppress gyro bias is a must to ensure the system's reliability. Based on the proposed method to calibrate the gyroscope, the testing result, as shown in Figure 22, indicates a significant decrease in gyro bias. The angular velocity obtained from the calibrated gyroscope was approximately centered around 0 deg/s in all axes under steady conditions.

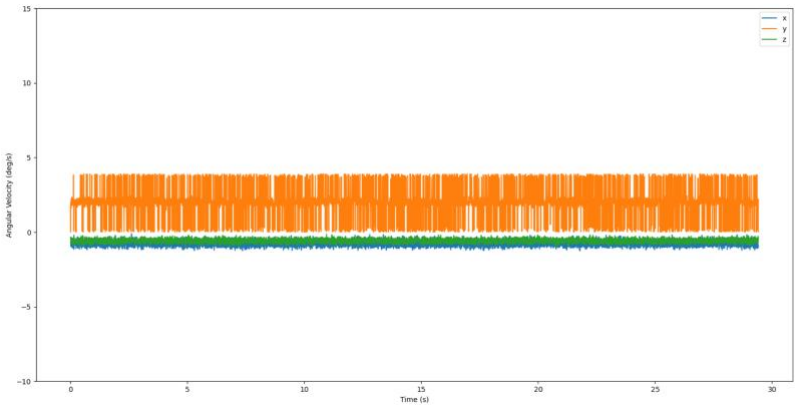


Figure 21: Angular Velocity from Gyroscope before Calibration

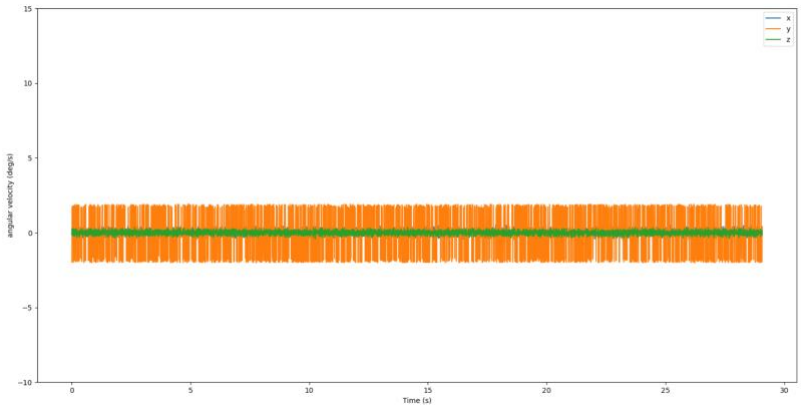


Figure 22: Angular Velocity from Gyroscope after Calibration

Accelerometer was calibrated as particularized in Chapter 3. Figure 23 shows linear accelerations obtained from the uncalibrated accelerometer under steady conditions. Testing was performed by placing the *z*-axis of the MPU6050 upwards against gravity. Thus, the expected data from the accelerometer should have given an acceleration of 0 m/s^2 for the *x* and *y*-axis, whereas 9.81 m/s^2 (1 g) for the *z*-axis. As the sensor fusion for the following project utilizes the gyroscope and accelerometer, the accelerometer must output reliable data. Hence, the accelerometer was calibrated for all axes using linear regression to estimate the offset, as discussed in Chapter 3. Figure 24 shows the graph of the obtained data after the accelerometer was calibrated. The result indicated a significant enhancement in which the linear acceleration in the *x* and *y*-axis was approximately centered around 0 m/s^2 and the *z*-axis of approximately 9.81 m/s^2.

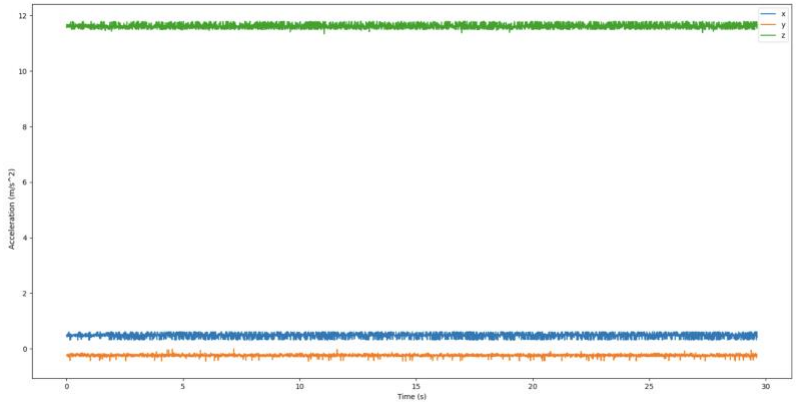


Figure 23: Linear Acceleration from Accelerometer before Calibration

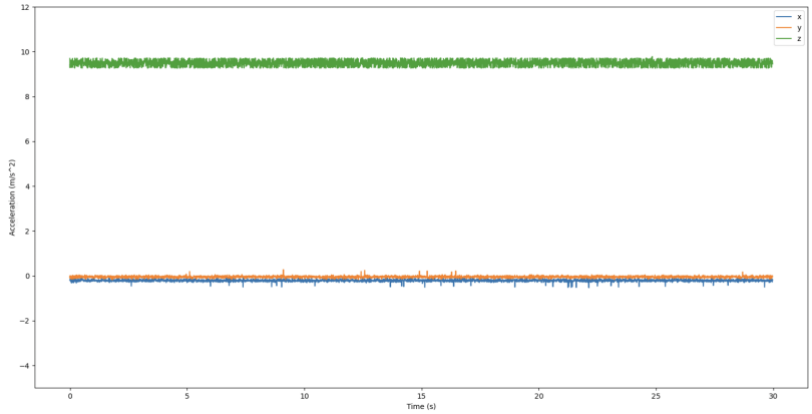


Figure 24: Linear Acceleration from Accelerometer after Calibration

Indeed, it is possible to obtain the platform's orientation by utilizing one of the sensors inside the MPU6050 module. However, though the calibrated gyroscope and accelerometer present better data than the uncalibrated data, the calculated orientation through the gyroscope might drift over time. In contrast, the calculated orientation by the accelerometer, which is prone to noise, might provide an inaccurate result. Moreover, it is visible that the obtained data from the gyroscope and accelerometer after calibration are intrinsically noisy. Therefore, Kalman Filter was implemented to acquire a more precise platform orientation based on the proposed equations in Chapter 3. Figure 25 shows the graph of the obtained orientation under steady conditions. It is manifest that the obtained result after Kalman Filter was applied indicates a significant enhancement in isolating the noise of each sensor. Moreover, the obtained result does not drift over time, indicating higher stability. Dynamic Testing was performed by repeatedly rotating the platform in each roll and pitch direction during testing time, as shown in Figures 26 and 27. The filter showed an outstanding performance in estimating the angle of the platform, at the same time isolating the noise of the environment.

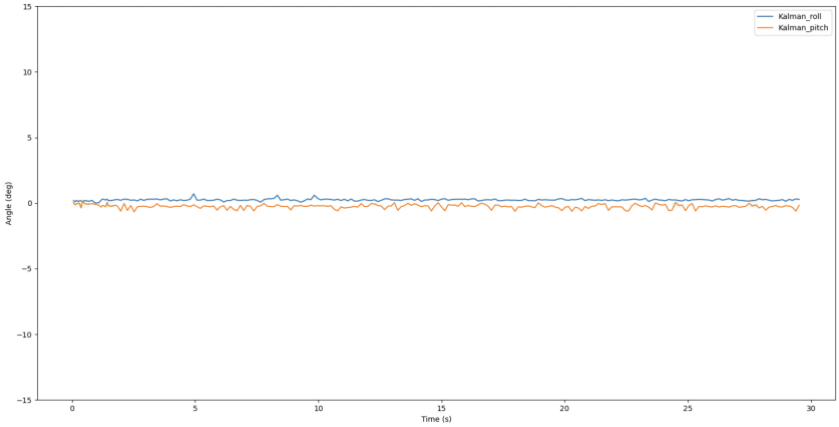


Figure 25: Kalman Filter Static Test

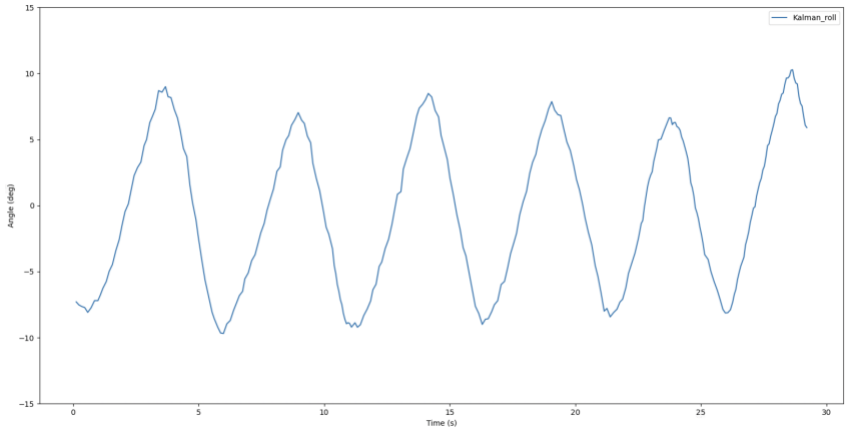


Figure 26: Kalman Filter Roll Dynamic Test

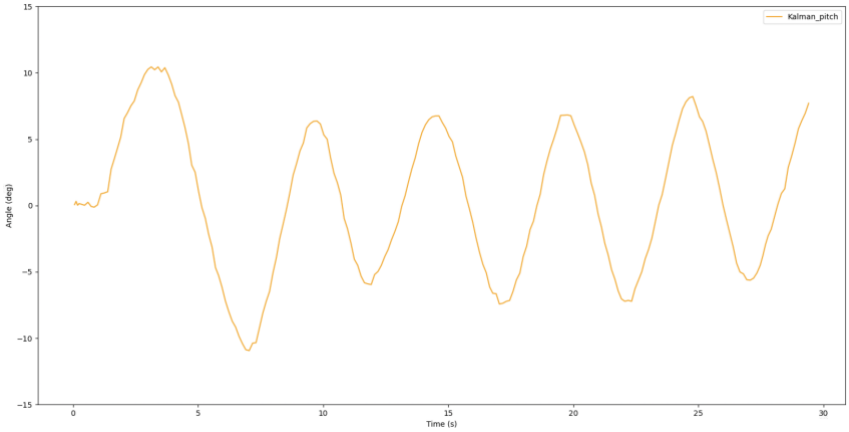


Figure 27: Kalman Filter Pitch Dynamic Test

5.2. End-Effector Stress/Strain Analysis

Due to the cost of the selected material for the actual model, the load testing was not able to be performed physically. However, to verify whether the obtained value of the thickness of the End-Effector design in Chapter 3 can withstand the load with minimal deflection, a simulation regarding Stress/Strain Analysis was performed through SolidWorks. The material of the End-Effector was defined as the AISI 347 Annealed Stainless Steel for the simulation. Moreover, the simulation was performed under a load of 2943 N, which is equivalent to 300 kg mass. The load was placed on top of the defined circular region, as shown in Figures 29 to 31, which was assumed as the base of the construction robotic arm. The property of the material used for the simulation is shown in Figure 28. Figure 29 shows the result of stress analysis for the End-Effector. From the analysis, it was evident that there was no point in the

End-Effector exceeding the yield strength of the AISI 347. Therefore, a load of 2943 N was proved to be in the range of the yield point where the plastic deformation of the material begins.

Moreover, the deflection analysis shown in Figure 30 indicated that the maximum deflection is where the center of gravity of the base of the construction robotic arm is located. The maximum deflection in the simulation was 0.1063 mm which did not exceed the established setpoint of 3 mm in Chapter 3. The amount of strain, as shown in Figure 31, was also minimal. The simulation results indicated that the performed calculation in Chapter 3 regarding the End-Effector design was acceptable though it was assumed to have rigid support at both ends. However, upon the implementation of the actual model, physical load testing is required to examine the actual mechanical stability.

Property	Value	Units
Elastic Modulus	1.95000001e+11	N/m^2
Poisson's Ratio	0.27	N/A
Shear Modulus	7.700000134e+10	N/m^2
Mass Density	8000.000133	kg/m^3
Tensile Strength	654999998.5	N/m^2
Compressive Strength		N/m^2
Yield Strength	275000000.9	N/m^2
Thermal Expansion Coefficient	1.7e-05	/K
Thermal Conductivity	16.3	W/(m·K)
Specific Heat	500	J/(kg·K)
Material Damping Ratio		N/A

Figure 28: AISI 347 Annealed Stainless Steel Material Property

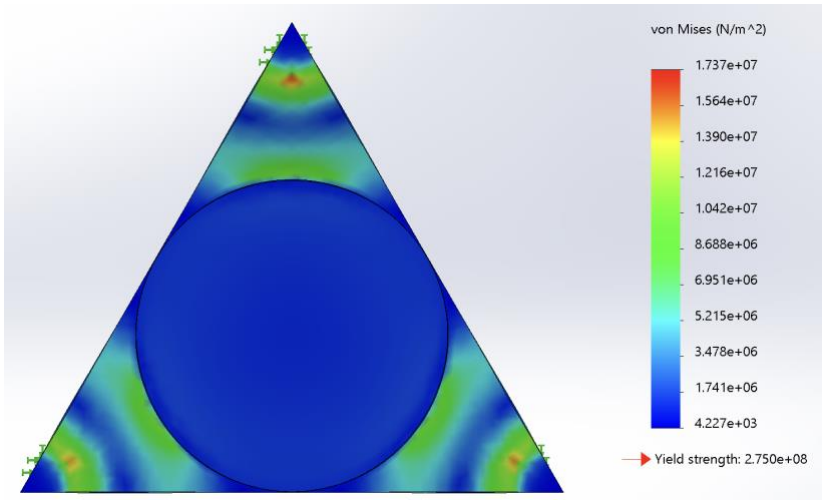


Figure 29: Stress Analysis of the End-Effector with 2943 N Load

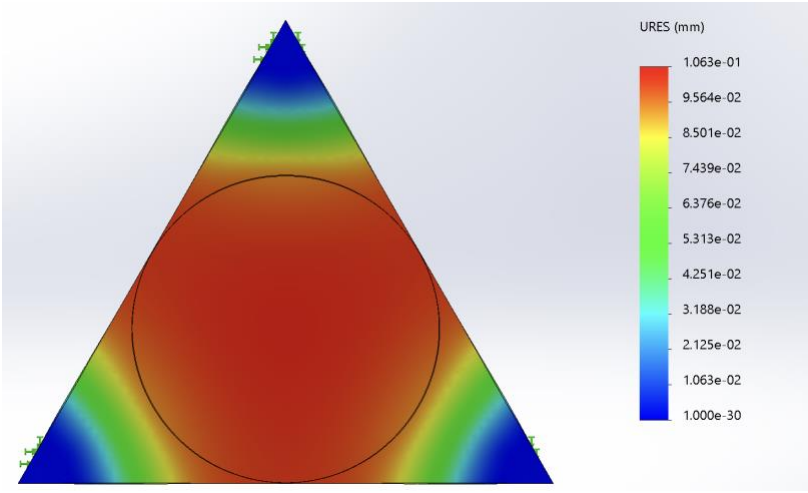


Figure 30: Deflection Analysis of the End-Effector with 2943 N Load

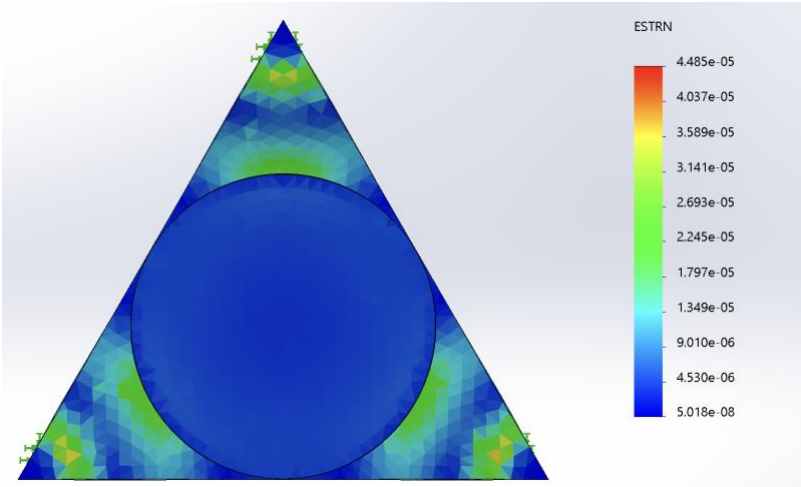


Figure 31: Strain Analysis of the End-Effector with 2943 N Load

5.3. System Response Time

Upon the assembly of the prototype, the settling time to compensate for the roll, pitch, and roll and pitch was measured. The test measured the settling time of the platform for Auto-Balancing both with and without 3kg mass. The PID gains were adjusted accordingly by examining the physical response and settling time. To measure the settling time accurately, a Python module, "time," was used to measure the time the loop exits when the platform is balanced. To track the system response, such as overshoot, the "print()" statement was utilized to output the current orientation of the platform throughout the loop. For the following tests, the time constant inside the low-pass filter for preventing derivative kick, τ, was set to 3.

The settling time of the platform was measured without mass. The initial derivative gain of the system utilized 0.0005. However, some test cases revealed an overshoot of ± 1 degree from the setpoint, which delayed the settling time. To come up with the issue, the derivative gain was increased to 0.001. The increased derivative gain decreased overshoot to ± 0.5 degrees, at the same time, decreasing the settling time, though it showed a minor difference. Therefore, the PID gains for the system without mass were set in Table 5. The results obtained using the PID gains as Table 5 are shown in Figure 32.

Table 5: Optimal PID gains for platform without mass

K_p	0.15
K_i	0.01
K_d	0.001

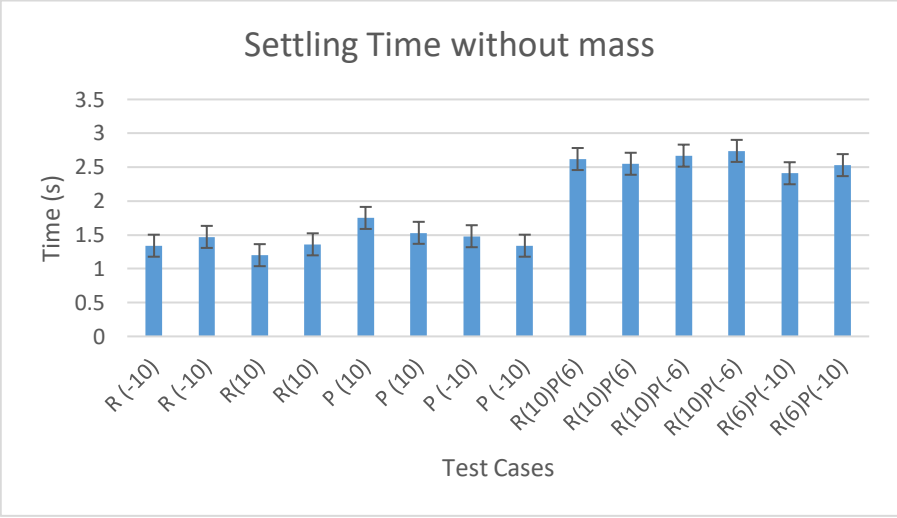


Figure 32: Settling Time without mass

The system used initial PID gains, as shown in Table 5, to obtain the settling time when 3 kg mass is placed on top of the platform. The system showed a significant decrease in settling time. Therefore, the proportional gain was increased to come up with a faster response of the system. Though the proportional and integral gain was increased, no physical oscillations were detected. Thus, the derivative gain used for the platform without mass was used. Figure 33 shows the system's settling time with PID gains, as shown in Table 6 with 3 kg mass.

Table 6: Optimal PID gains for platform with 3 kg mass

K_p	0.2
K_i	0.02
K_d	0.001

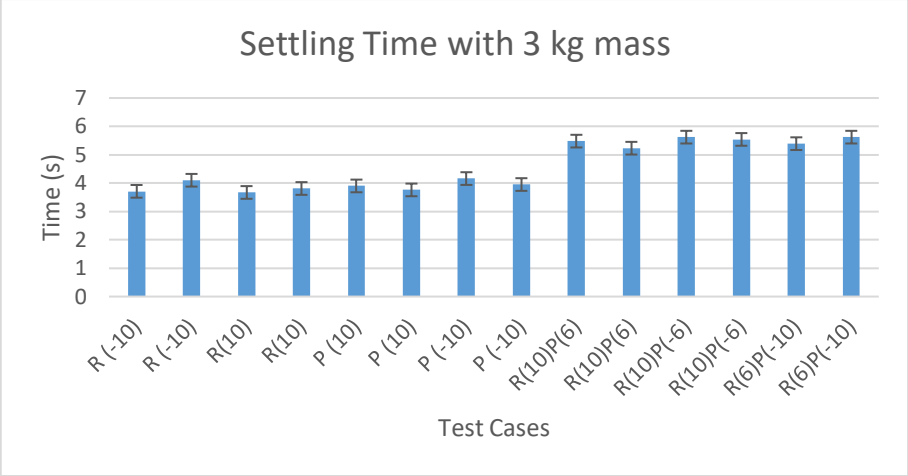


Figure 33: Settling time with 3 kg mass

```
roll is positive 6
motor1 is on (CCW)
pos1: 70.95311875518668
adjusting roll
roll is negative -1
motor1 is on (CW)
pos1: 67.35245587454503
adjusting roll
roll adjusted
pos1,  67.35245587454503  pos2,  0.0  pos3,  0.0
adjusted in: 1.204071044921875  seconds
Exiting
```

Figure 34: System Response Tracking

6. APPLICATIONS

The following chapter will introduce the Auto-Balancing Platform in real-life usage, especially in construction applications. As implied, the main objective of the following project was to make an Auto-Balancing Platform on top of the scissor lift to ensure the reliability and safety of the Construction Robotic Arm. Thus, the Auto-Balancing Platform will serve as the basis of the Construction Robotic Arm working on top of the scissor lift.

Figure 35 shows the Construction Robotic Arm to be placed on top of the Auto-Balancing Platform. For the actual implementation, the square plate at the base of the Construction Robotic Arm shown in Figure 35 will be replaced with the proposed Auto-Balancing Platform. For the actual implementation, the gearbox should be attached to the pinion to orient the construction robotic arm. Considering the mass of the Construction Robotic Arm of 300 kg, each actuator should carry a 100 kg load, as mass is distributed evenly across the End-Effector. Moreover, to avoid speed reduction as much as possible, a gear ratio of 2:1 is recommended. Assuming the pitch diameter of the pinion is 0.1 m, the required torque for each motor is as follows:

$$T_{driven} = 0.05 \times 2 \times 100 \times 9.81 = 98.1 \text{ Nm} \quad \text{..... (56)}$$

$$T_{driver} = T_{driven} \times \frac{1}{2} = 49.05 \text{ Nm} \quad \text{..... (57)}$$

Therefore, a motor of 50 Nm is needed to implement an actuator through a rack and pinion mechanism for the actual application. Moreover, load testing of the End-Effector should be performed before the actual application to ensure mechanical stability.

Aside from serving as a basis for the Construction Robotic Arm, the developed prototype can be utilized as an End-Effector of the Robotic Arm. The degrees of freedom the proposed Auto-Balancing Platform provides 3-DOF, which is sufficient for most of the applications, accounting for the fact that the human wrist has 3-DOF as well.

Figure 35: Construction Robotic Arm [12]

7. CONCLUSIONS

The project developed a prototype of the Auto-Balancing Platform for construction applications. The platform achieved a stable and accurate orientation using the low-cost MEMS IMU sensor module. Several Methodology was implemented to develop the prototype. The platform utilized Kalman Filter to fuse the incoming data from the accelerometer and gyroscope to obtain reliable feedback. Moreover, PID control was implemented to achieve a prompt and accurate system response. As no encoder was utilized for the prototype, Inverse Kinematics was derived from tracking each actuator's traveled distance based on the sensor's given orientation.

Instead of using the electrical cylinders in the market, which are slow, the platform utilized an actuator using the rack and pinion mechanism to compensate for the angular deviation of the platform expeditiously. Furthermore, the project investigated the required thickness of the End-Effector of the actual model using the moment area method and SolidWorks Stress/Strain analysis.

To conclude, the developed prototype based on the proposed Methodology could quickly and accurately respond to the angular deviations. However, in some instances, the incoming feedback from the IMU had an error of ± 0.5 degrees. Therefore, to minimize the error, a higher precision IMU could be used in future versions of the model instead of the low-cost MEMS sensors.

8. FUTURE WORKS

The actual model of the Auto-Balancing Platform works closely with the Construction Robotic Arm sitting on top. It is highly recommended to integrate both control systems to reduce conflicts. Moreover, rather than using a DC or Continuous Rotational Servo Motor, which cannot control the rotational angle, implementation through a stepper motor will be able to increase the accuracy by moving the joints accordingly based on the obtained feedback from the sensors. Lastly, the in-depth mechanical stability of the platform should be performed to ensure no mechanical failures occur during the operation.

REFERENCE:

[1] B. Dasgupta and T. S. Mruthyunjaya, "The stewart platform manipulator: A Review," *Mechanism and Machine Theory*, vol. 35, no. 1, pp. 15–40, 2000.

[2] I. Arun Faisal, T. Waluyo Purboyo, and A. Siswo Raharjo Ansori, "A review of accelerometer sensor and gyroscope sensor in IMU sensors on motion capture," *Journal of Engineering and Applied Sciences*, vol. 15, no. 3, pp. 826–829, 2019.

[3] C. M. Graney, "Coriolis effect, two centuries before Coriolis," *Physics Today*, vol. 64, no. 8, pp. 8–9, 2011.

[4] N. P. Anh, B. H. Hoang, N. T. Hoai, and D. Duy, "A model of 3-DOF stewart auto-balancing system for low-cost and high reliable embedded applications," *2021 15th International Conference on Advanced Computing and Applications (ACOMP)*, 2021.

[5] P. Gui, L. Tang, and S. Mukhopadhyay, "MEMS based IMU for Tilting Measurement: Comparison of complementary and Kalman filter based data fusion," *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, 2015.

[6] M. Haid and J. Breitenbach, "Low cost inertial orientation tracking with Kalman filter," *Applied Mathematics and Computation*, vol. 153, no. 2, pp. 567–575, 2004.

[7] D. Doan, H. H. Bui, D. X. Nguyen, and M. T. Nguyen, "An implementation of low-cost auto-balancing embedded system for safety mechanisms," *2022 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, 2022.

[8] C. Yang, Q. Huang, J. He, H. Jiang, and J. Han, "Model-based control for 6-DOF parallel manipulator," *2009 International Asia Conference on Informatics in Control, Automation and Robotics*, 2009.

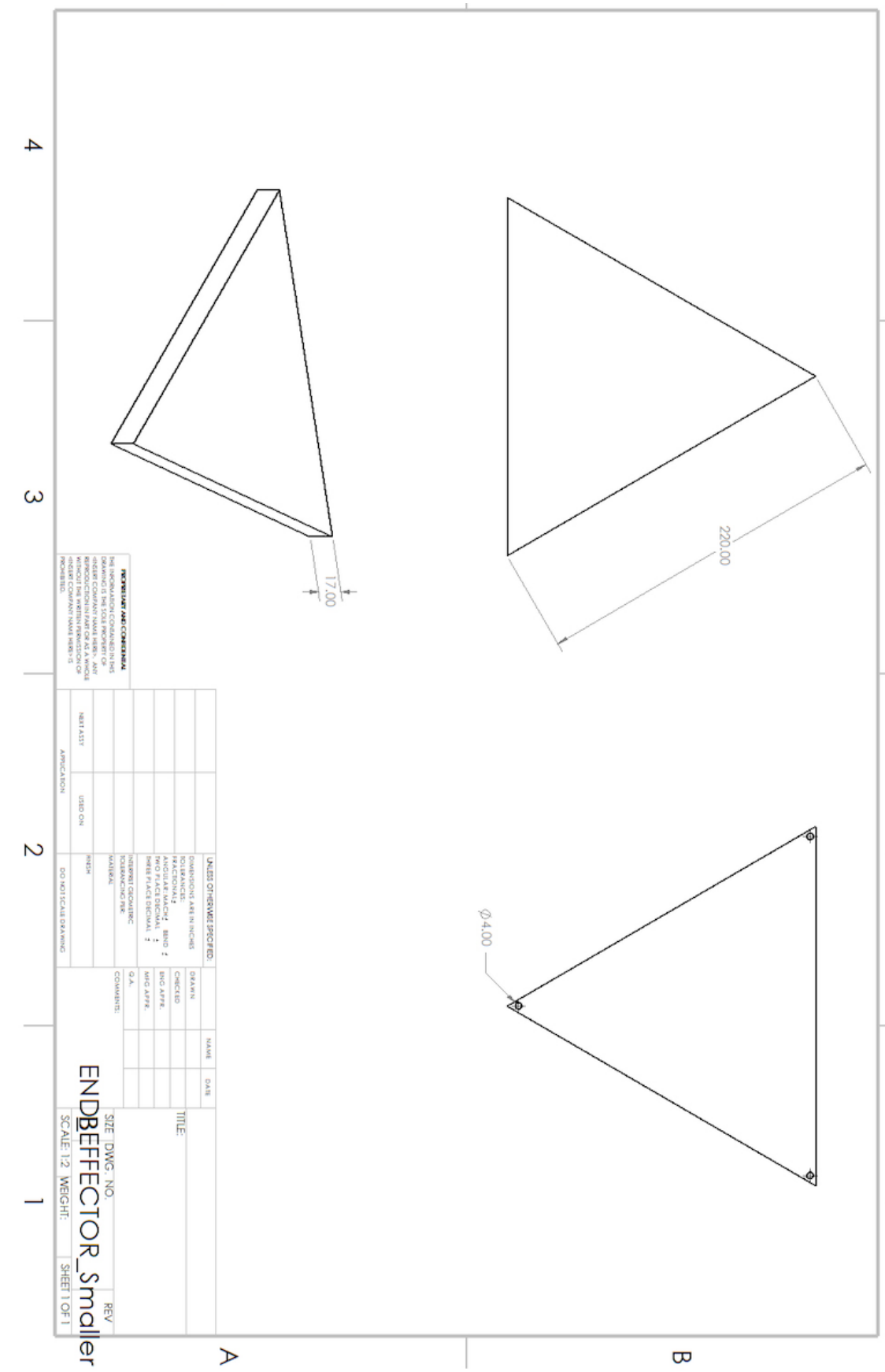
[9] Genie, "Genie® GS™-2632 & GS-2646," GS-2632 & GS-2646 – Product Specifications – 2022 – EN-US, Aug. 2022

[10] From Thermo Fisher Scientific – Materials & Structural Analysis, F. Formulaction, F. E. Instruments, F. P. S. Corp, F. Z. Corporation, and F. T. Olsen, "Properties: Stainless steel - grade 304 (UNS S30400)," *AZoM.com*, 05-Apr-2023. [Online]. Available: https://www.azom.com/properties.aspx?ArticleID=965. [Accessed: 10-Oct-2022].

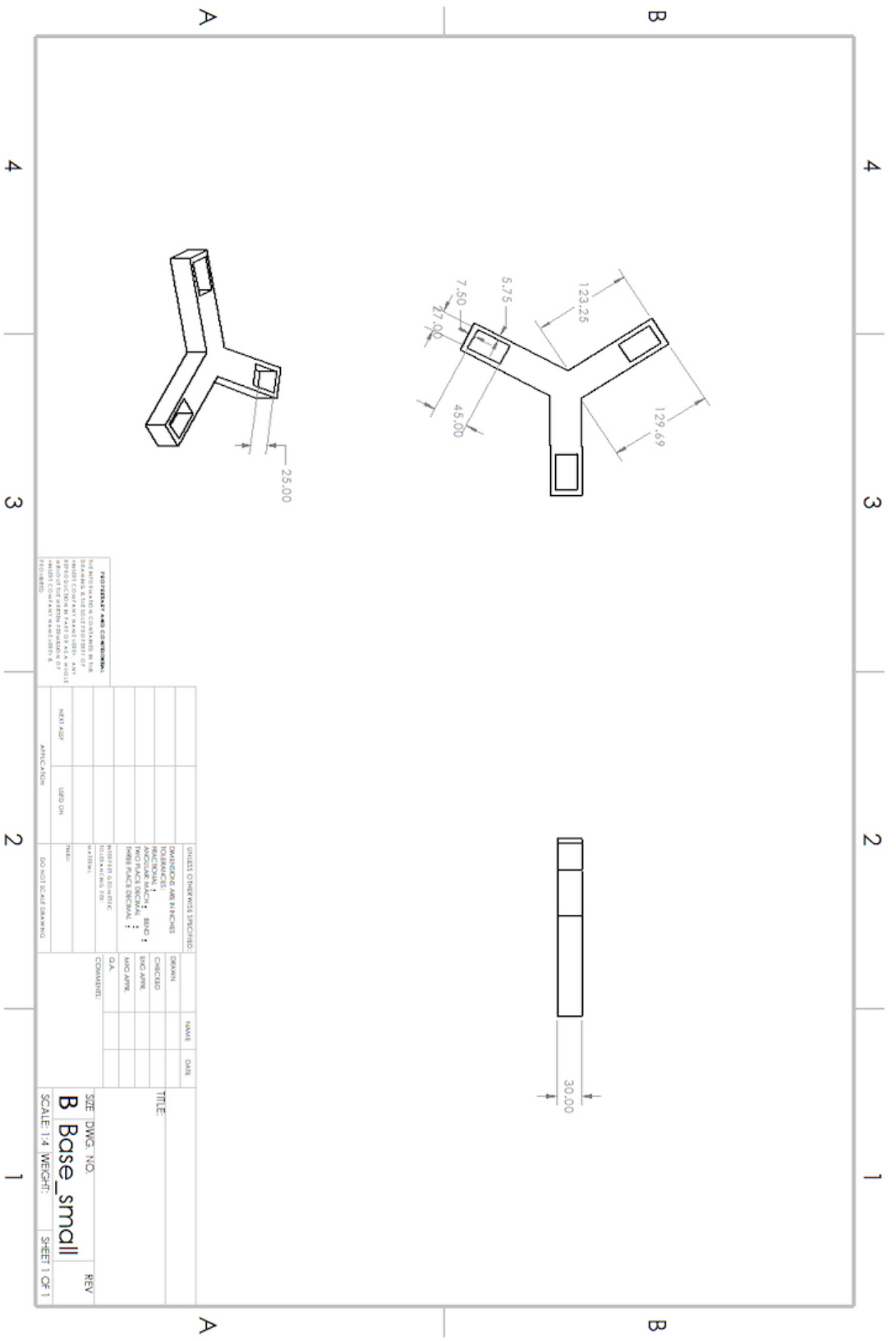
[11] InvenSense, "MPU-6000/MPU-6050 Product Specification," PS-MPU-6000A-00, Aug. 2013

[12] Jiangyin Tongli Industrial Co., Ltd., Hong Kong, working paper.

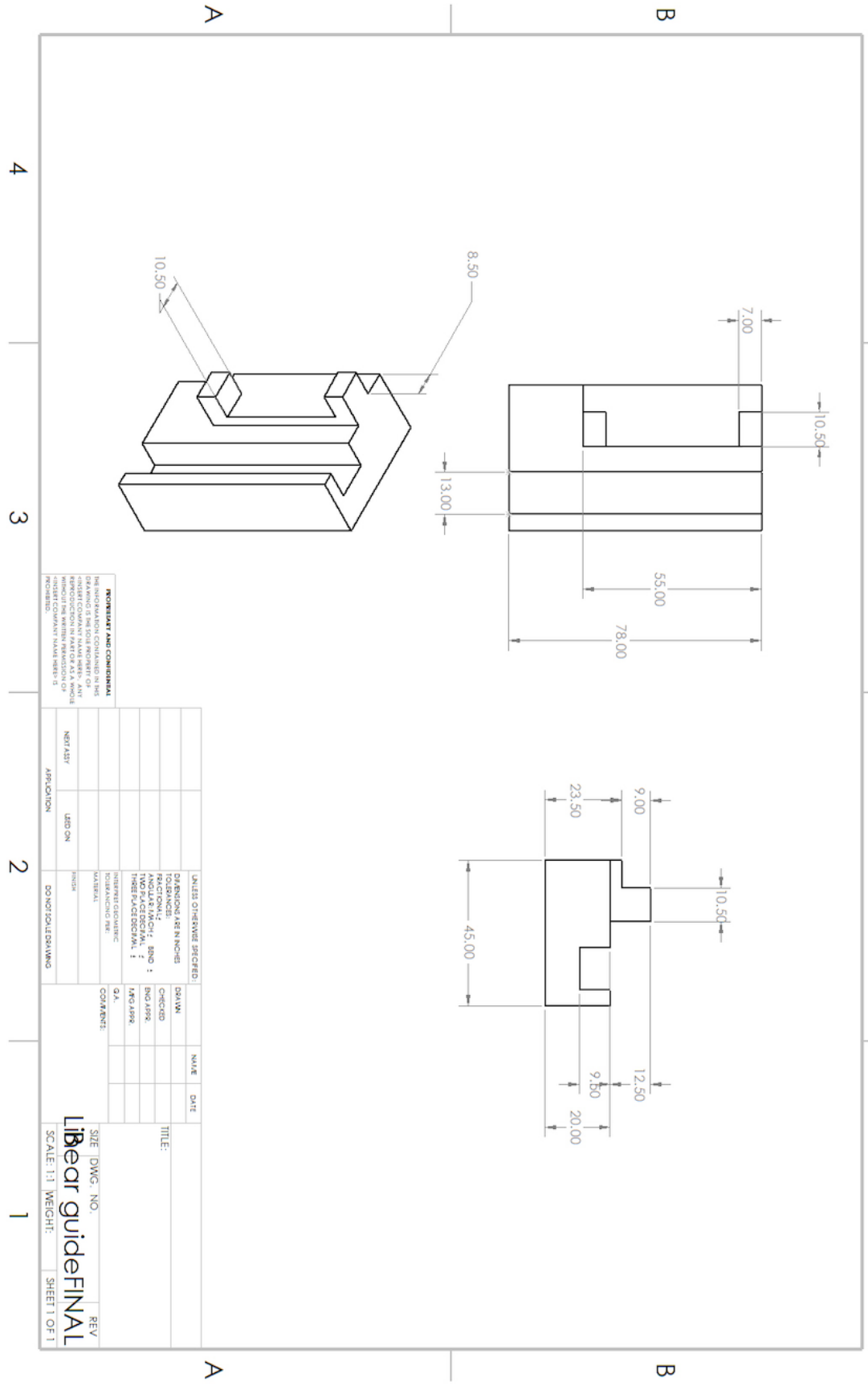
APPENDIX I CAD DRAWING OF END-EFFECTOR



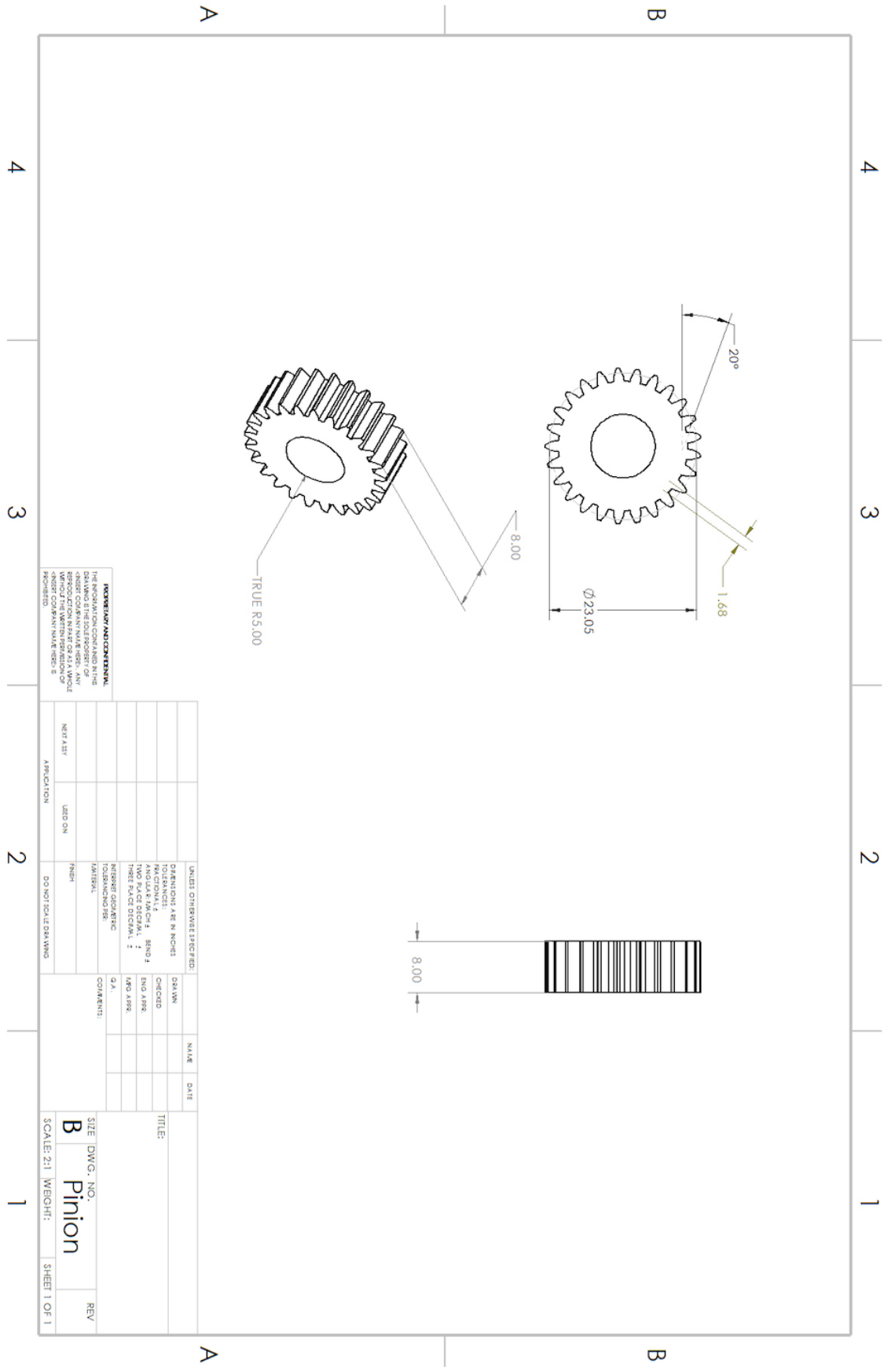
APPENDIX II CAD DRAWING OF BASE



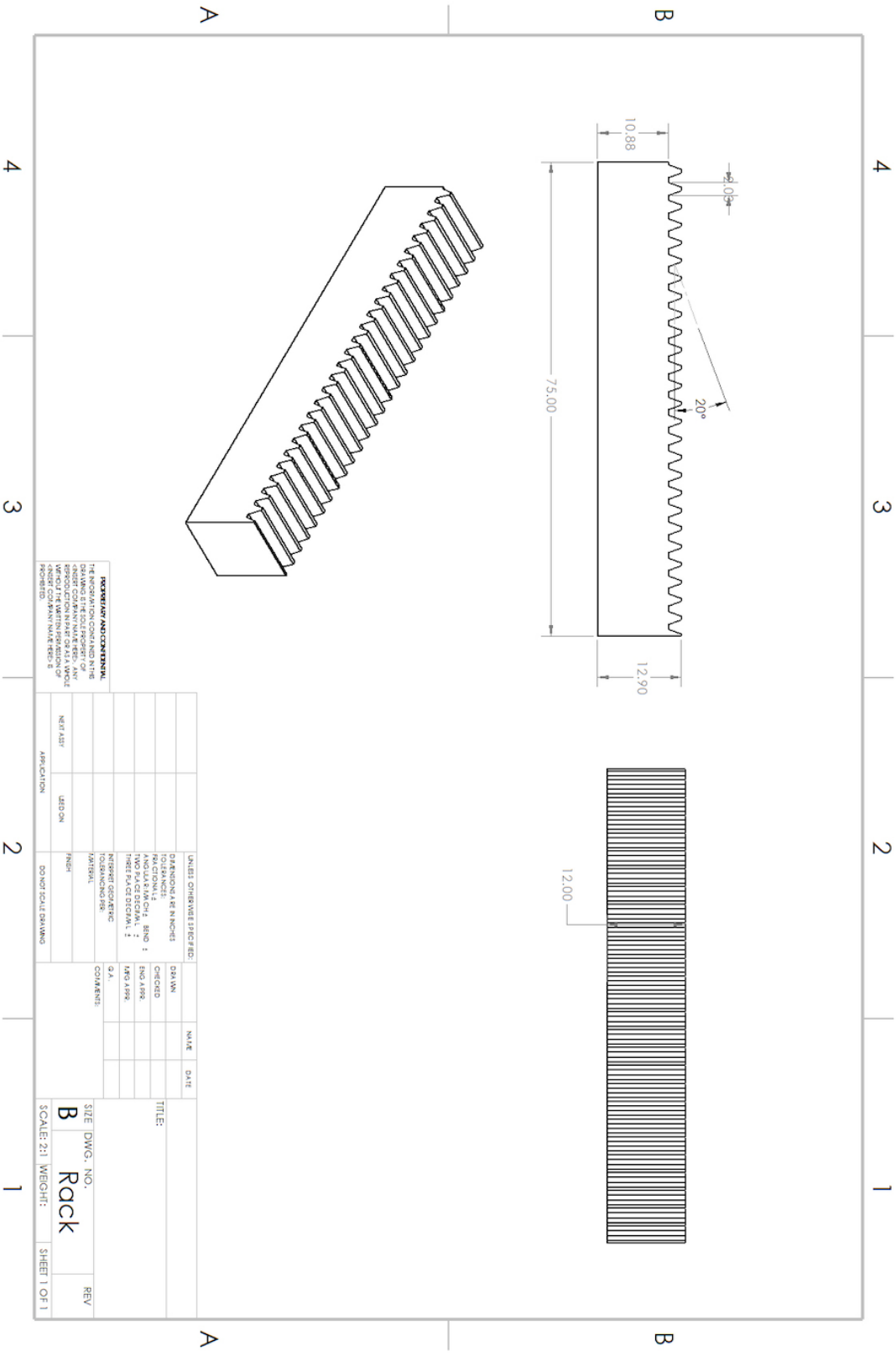
APPENDIX III CAD DRAWING OF LINEAR GUIDE



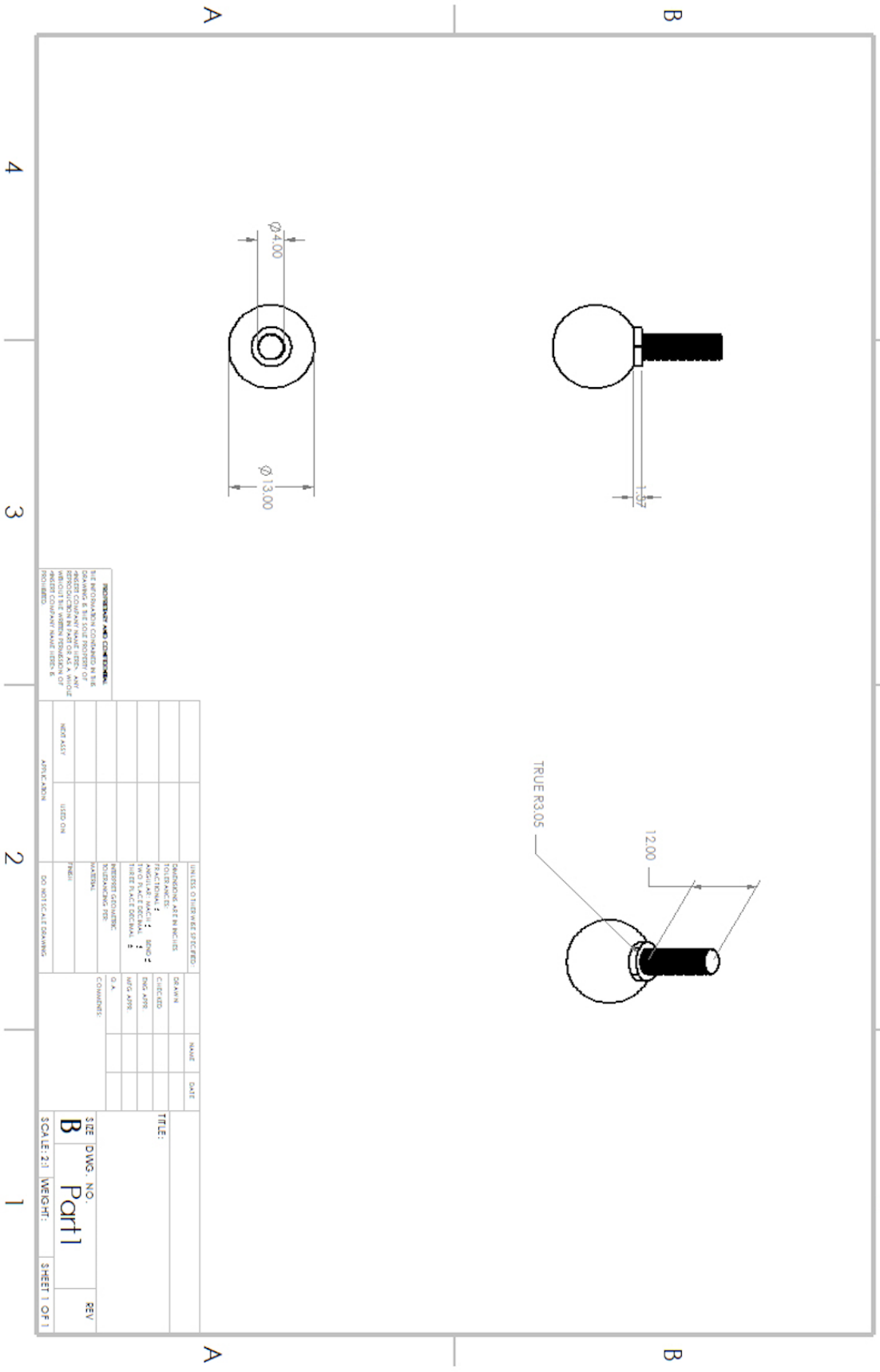
APPENDIX IV CAD DRAWING OF PINION



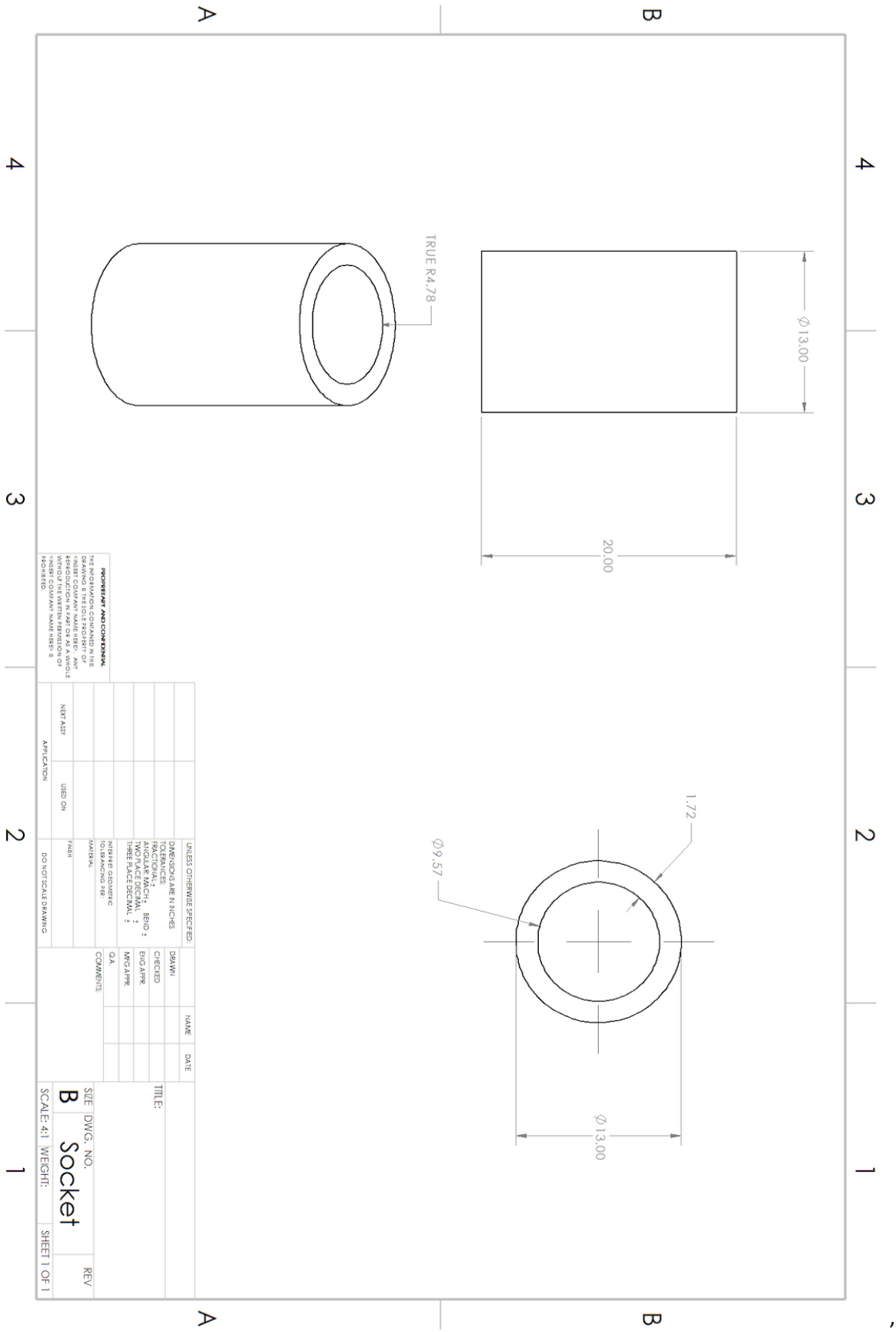
APPENDIX V CAD DRAWING OF RACK



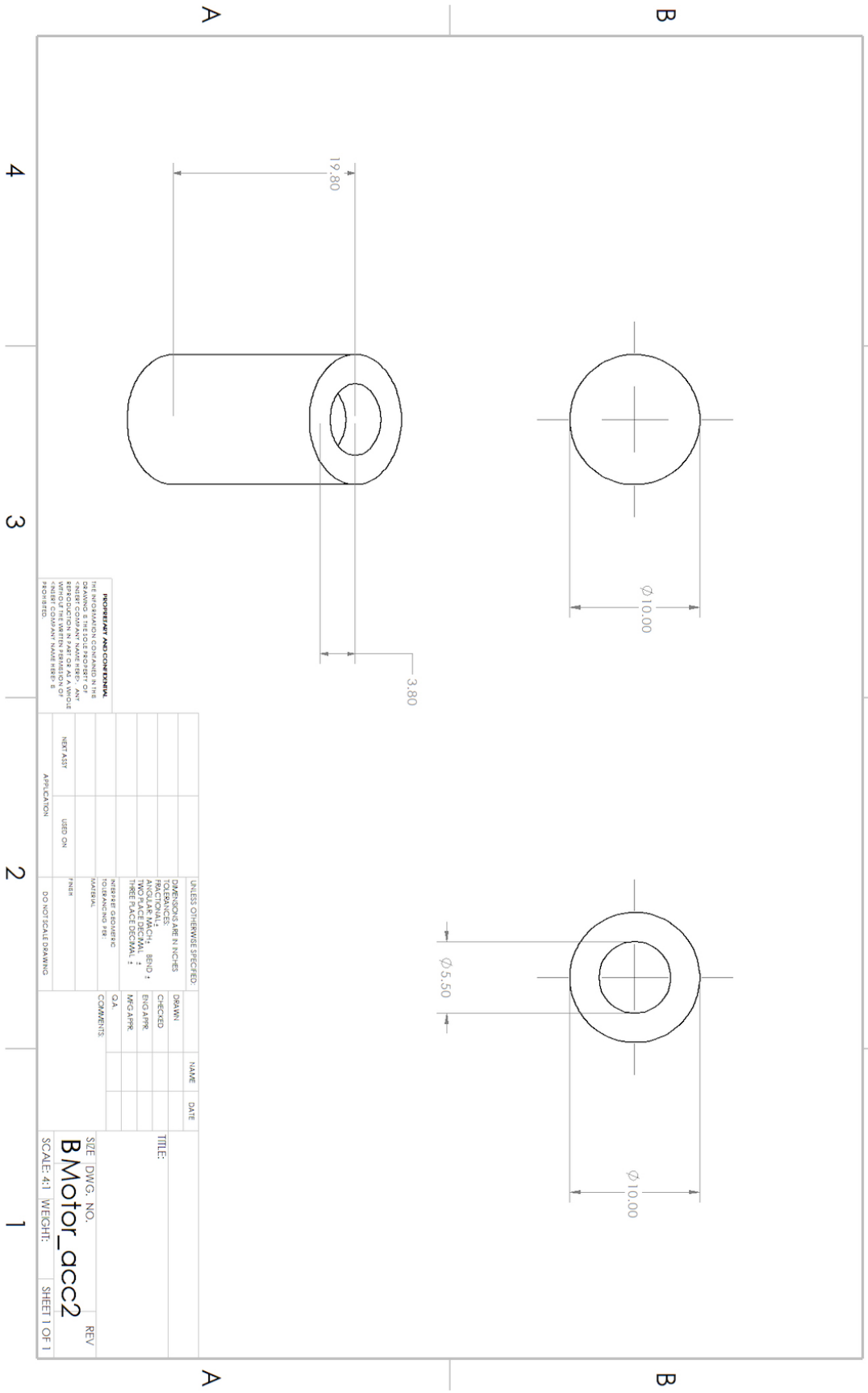
APPENDIX VI CAD DRAWING OF BALL JOINT



APPENDIX VII CAD DRAWING OF SOCKET JOINT



APPENDIX VIII CAD DRAWING OF MOTOR COUPLING PART



APPENDIX IX COMPUTER CODE

The computer code to run the prototype of the Auto-Balancing Platform is uploaded to GitHub: https://github.com/bobkim938/auto-levelling-platform