Self Balancing Robot

Presented by Group Monday A

01

Design Lab - 15 MAY 2025

The principle

The self-balancing robot operates on the fundamental principle of an inverted pendulum and feedback system, where the system dynamically adjusts itself to maintain an upright position. At the core of the robot is the MPU-6050 sensor which continuously measures the robot's pitch (tilt) angle, providing real-time orientation data. The raw data from the MPU-6050 is filtered using a complementary filter to obtain an accurate estimate of the tilt angle, which is crucial for maintaining balance.

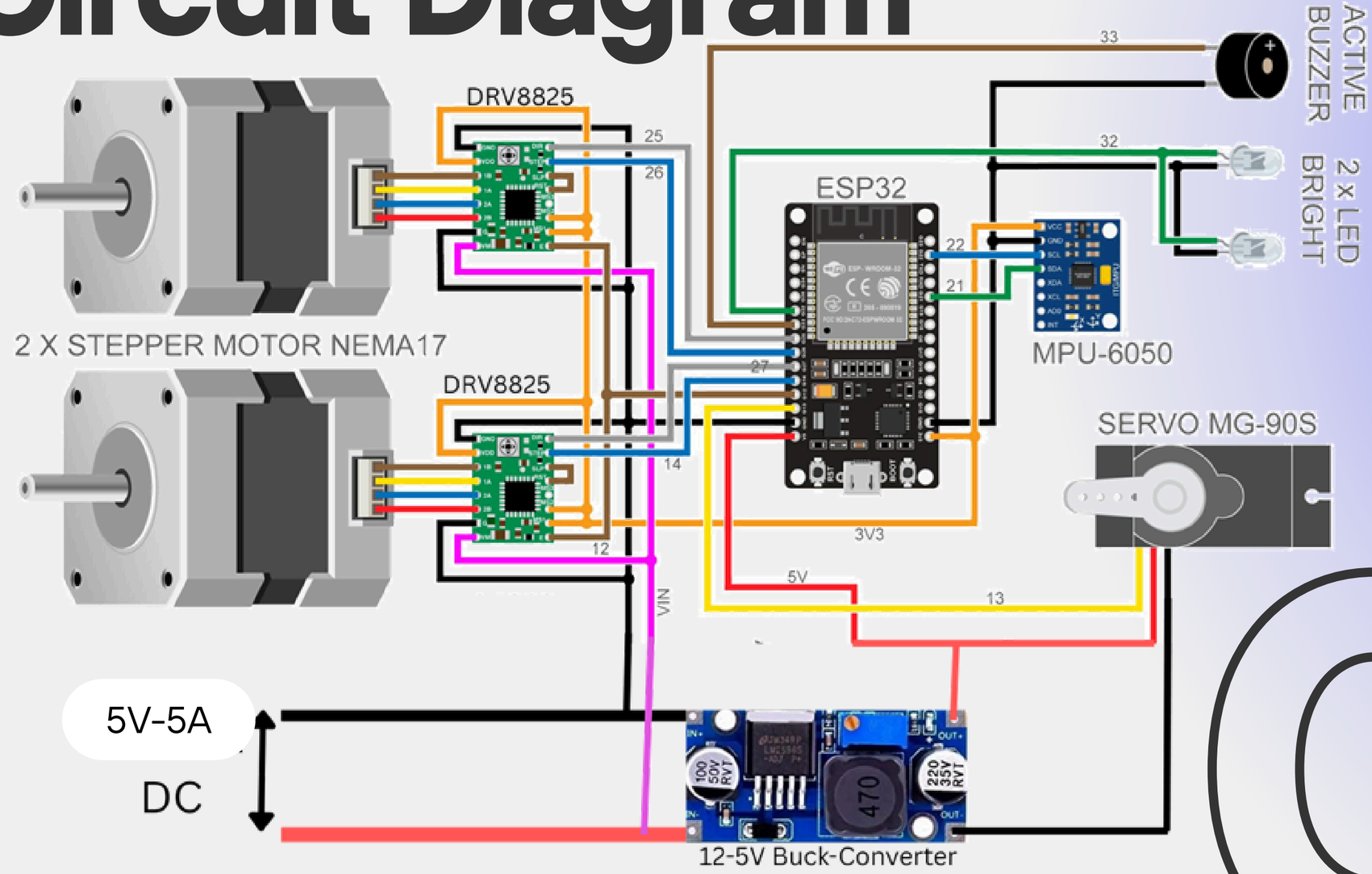
The balance is achieved using a Proportional-Integral-Derivative (PID) control algorithm, which takes the pitch angle as input and determines the corrective motion required. The PID controller calculates an error value as the difference between the desired upright position (setpoint) and the current tilt angle. It then processes this error through proportional, integral, and derivative terms to compute a control signal that is sent to the stepper motors at the wheels. By appropriately adjusting the speed and direction of these motors, the robot moves forward or backward to counteract any tilt, thus maintaining balance. This feedback loop enables the robot to self-correct and stabilize itself in real time without human help.

15 MAY 2025

-

Design Lab

Circuit Diagram



02

15 MAY 2025

-

Design Lab

Getting Acurrate Pitch angle

To balance the robot effectively, we need a stable and precise pitch angle. This is achieved by combining the strengths of the accelerometer and gyroscope in the MPU6050 sensor using a technique called sensor fusion.

- Accelerometer gives the angle based on gravity but is noisy.
- Gyroscope provides smooth angular velocity but drifts over time.

We use a complementary filter to blend both:

- It relies more on gyroscope for short-term precision.
- It uses the accelerometer to correct long-term drift.

This fusion results in a pitch angle that is both responsive and stable, allowing us to maintain the robot's balance in real-time.

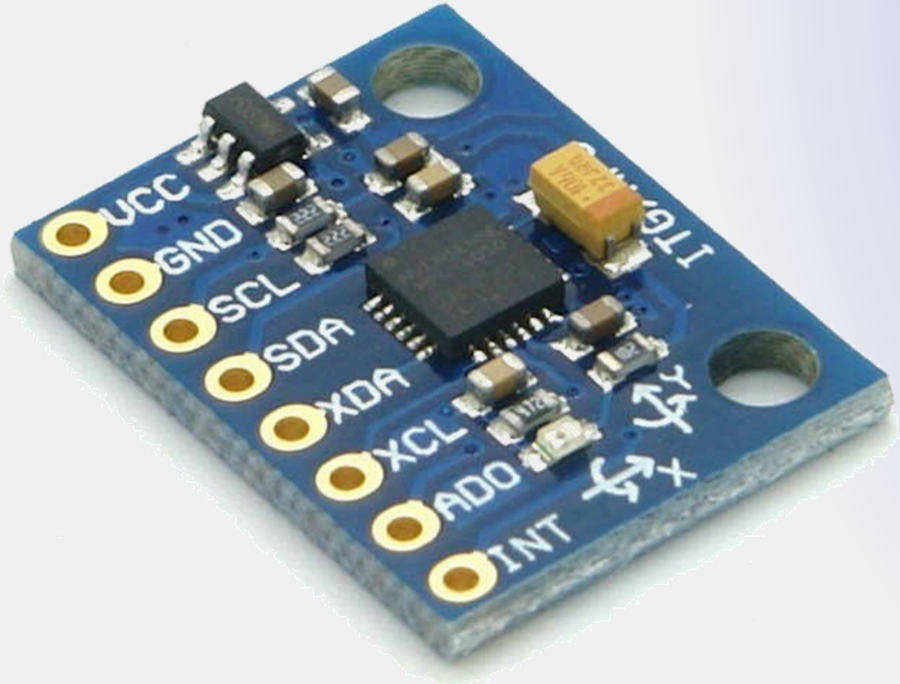
03

```
// Calculate pitch from accelerometer
float pitchAcc = atan2(-accelX, sqrt(accelY * accelY + accelZ * accelZ)) * RAD_TO_DEG;

// Integrate gyroscope data to get pitch
pitchGyro += gyroY * dt;  // gyroY in deg/s, dt in seconds

// Combine using complementary filter
pitch = 0.96 * pitchGyro + 0.04 * pitchAcc;
```

MPU-6050

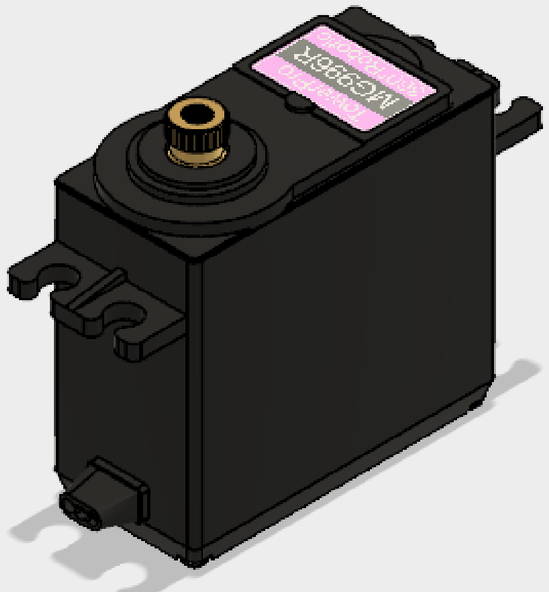


04

Servo MG 996R

The servo motor has been used for the parallel plate at the top of the frame, which would have remained parallel to the ground irrespective of the tilt.

To control the servo motors, we used the pitch angle directly and introduced a proportional opposite pitch rotation. We slowed this response, in order to not produce any jerkiness in the balancing of the top plate.



P-I-D Control

We added Proportional Integral Control to the Stepper Motors to ensure that the frame remained relatively balanced.

Components of PID:

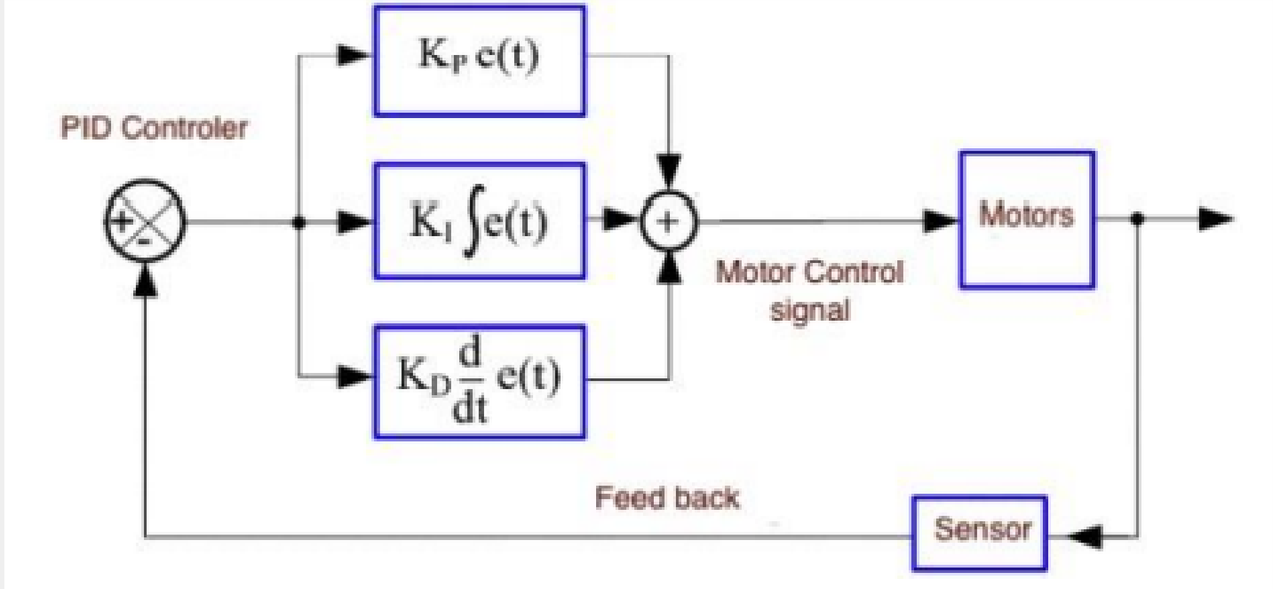
1. Proportional (P)
   - Reacts to the current error.
   - Bigger error → stronger correction.
   - Helps respond quickly.
2. Integral (I)
   - Reacts to the accumulated past error.
   - Helps eliminate long-term bias (drift).
   - Can cause overshoot if too strong.
3. Derivative (D)
   - Reacts to the rate of change of error.
   - Predicts and slows down the system before overshooting.
   - Helps smooth the response.

07

Design Lab - 15 MAY 2025

P-I-D

```
error = setpoint_angle - pitch_angle;
integral += error * dt;
derivative = (error - previousError) / dt;
previousError = error;
output = Kp * error + Ki * integral + Kd *
                                derivative;
```



Movement Control of the Bot (Forward/Backward)

To move a self-balancing robot forward or backward, we adjust the setpoint of the system, which represents the desired tilt angle of the bot. In a stationary balanced state, the setpoint is typically zero, meaning the bot is perfectly upright. When we want the bot to move forward, we shift the setpoint slightly in the forward direction, causing the control system (usually a PID controller) to interpret this as an error since the bot is now leaning behind the desired angle. As a result, the motors drive the wheels forward to bring the bot back to the new setpoint. This motion creates forward movement. This dynamic balance and correction process allows the bot to continuously move while staying upright. The key is that the bot always tries to "fall" in the direction we want it to go, and the motors act to prevent it from actually falling, creating movement.

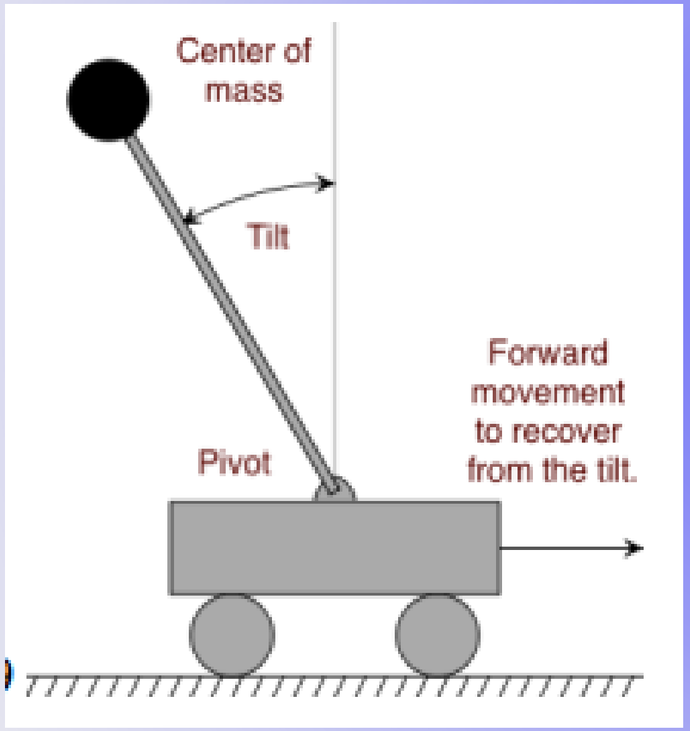
15 MAY 2025

-

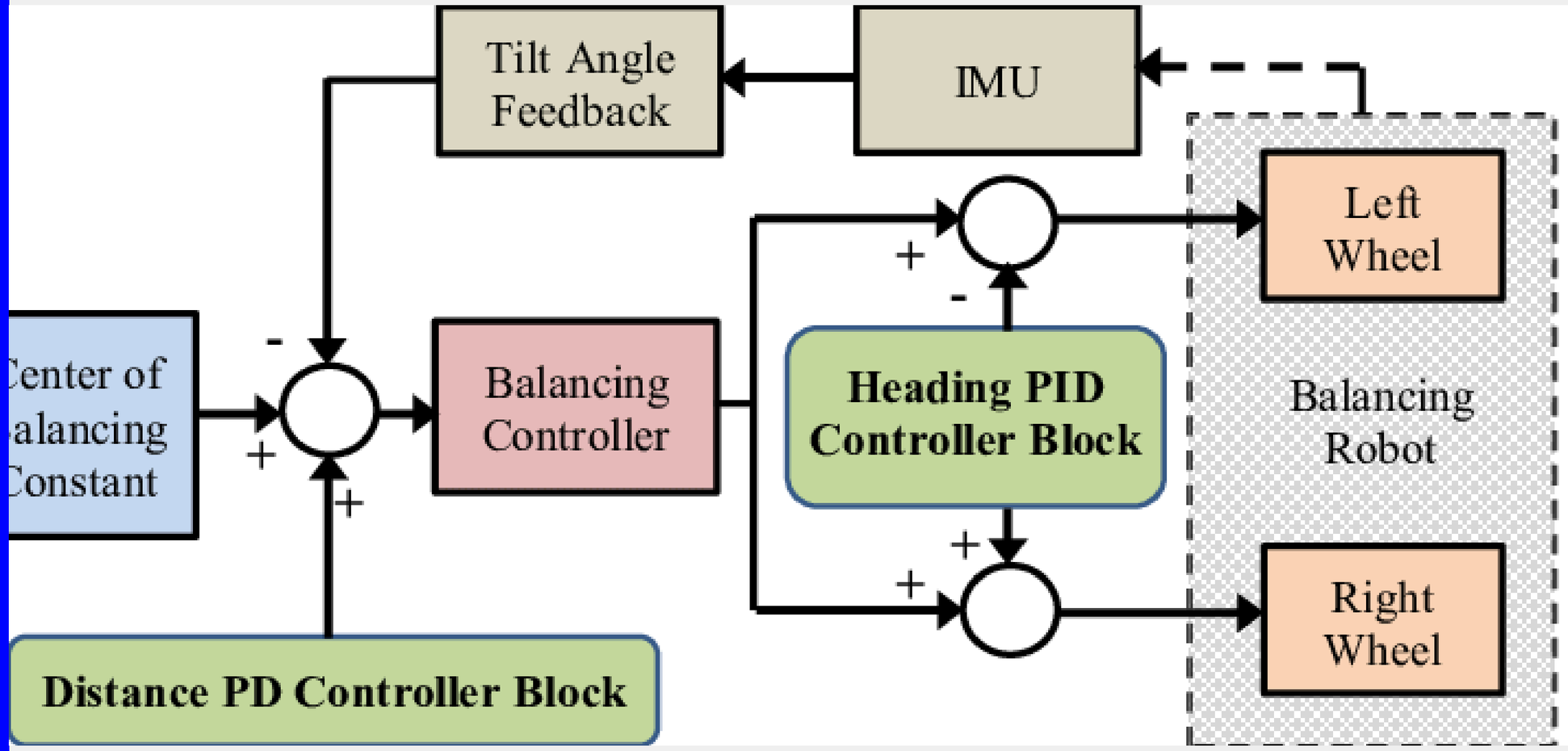
Design Lab

Movement Control of the Bot (Turning)

To turn the bot, we create a differential in wheel speeds. By increasing the speed of one wheel while keeping the other slower or even stationary, the bot rotates around the slower wheel. For example, speeding up the right wheel while slowing the left causes the bot to turn left. This method allows the bot to pivot or make smooth turns while maintaining its balance through continuous feedback control.



08


Tilt Angle Feedback
IMU
Left Wheel
Center of Balancing Constant
Balancing Controller
Heading PID Controller Block
Balancing Robot
Right Wheel
Distance PD Controller Block
+
-
+
+
-
+
+

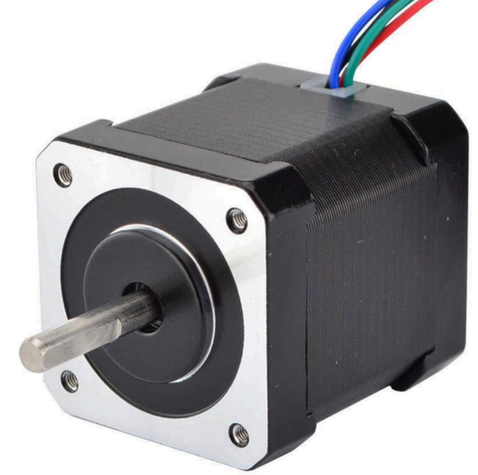
10

Design Lab

-

15 MAY 2025

Stepper Motor



NEMA 17

Working:

A stepper motor moves in discrete steps by converting electrical pulses into mechanical rotation. Each pulse advances the motor shaft by a fixed angle, allowing precise control over position and speed. It typically operates using electromagnetic coils arranged in phases, which are energized in a sequence to rotate the shaft incrementally.

Benefits over DC/Servo Motor:

- Precise Control: Ideal for accurate angular positioning.
- Repeatability: Returns to exact positions without feedback systems.
- Open-Loop Operation: No need for sensors to track position.
- High Torque at Low Speed: Suitable for applications requiring stability over speed.

Use in the Project:

We used NEMA 17 steppers with an DRV8825 driver and fed MPU6050 tilt data into a PID loop that adjusts each step pulse—this gave us the precise, real-time control needed to keep the robot balanced and responsive.

Structural analysis of our model

Overview:

To ensure the mechanical strength and reliability of our model, we performed structural analysis using Shear Force Diagram (SFD) and Bending Moment Diagram (BMD). This analysis helped us understand the internal forces and moments acting on the structure under both static and dynamic (turning/motion) conditions.

- The SFD allowed us to identify how shear forces vary along the length of the structure.
- The BMD provided insight into the distribution of bending moments, helping us locate the points of maximum bending stress.

Using the maximum bending moment from the BMD, we applied the bending stress formula:

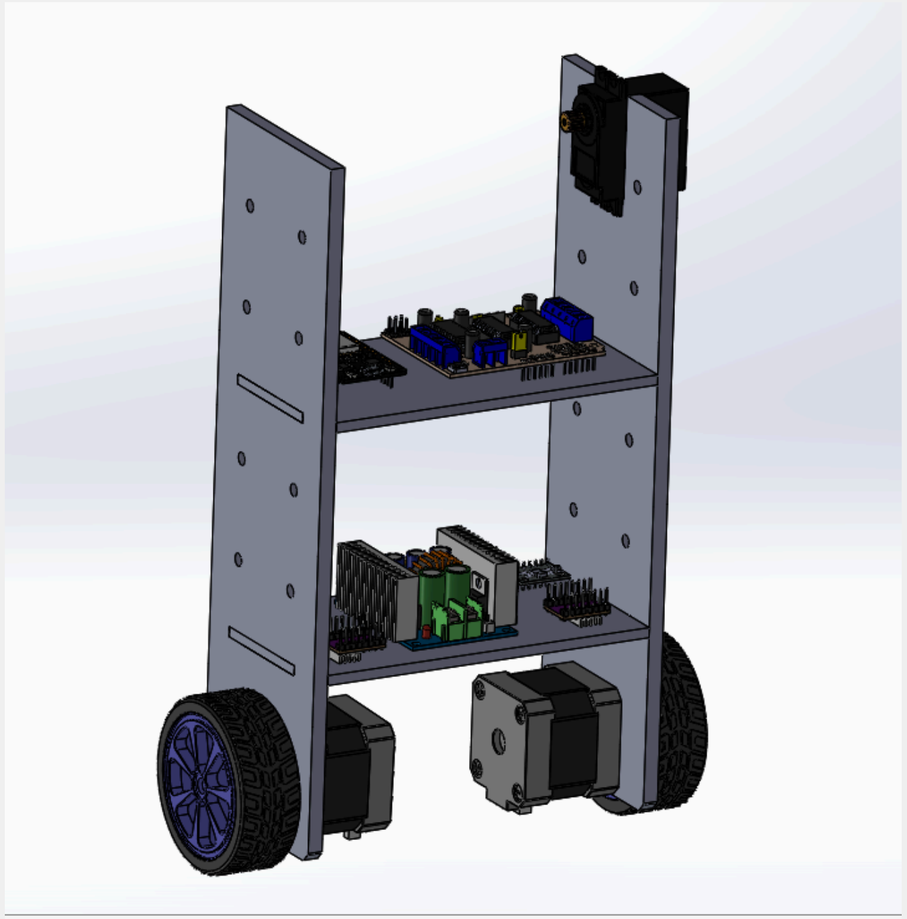
:

$$\sigma_{bend} = \frac{My}{I} \text{ where:}$$

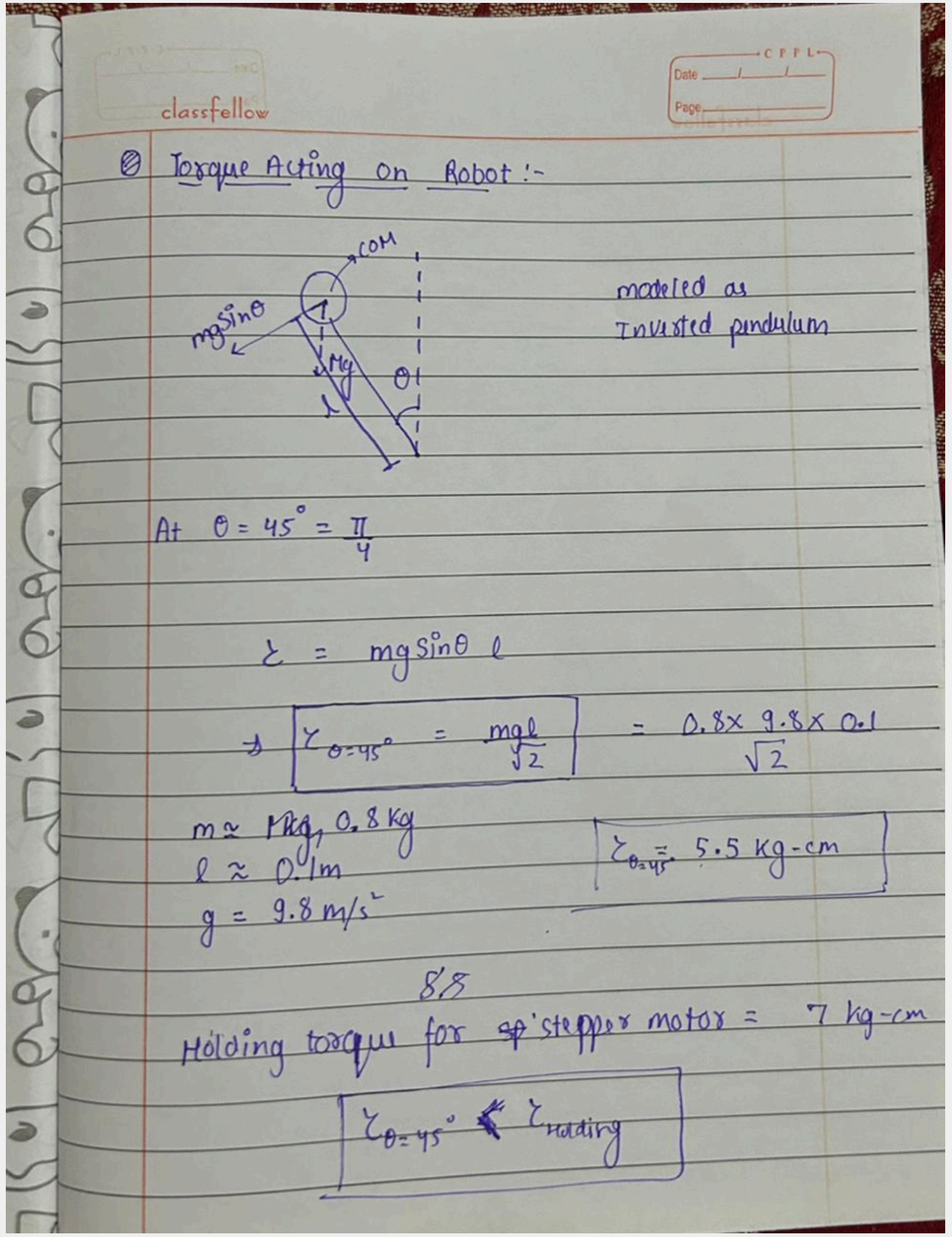
M = the internal bending moment about the section's neutral axis
y = the perpendicular distance from the neutral axis to a point on the section
I = the moment of inertia of the section area about the neutral axis

From this, we can either calculate the required thickness(t) and material dimensions to prevent structural failure or the maximum bending stress and check if it exceeds the max bending stress of the material.

By comparing results under static loading (e.g., resting condition) and dynamic loading (during turning and motion), we ensured the model maintains structural integrity in all operational states.



TORQUE ACTING ON ROBOT



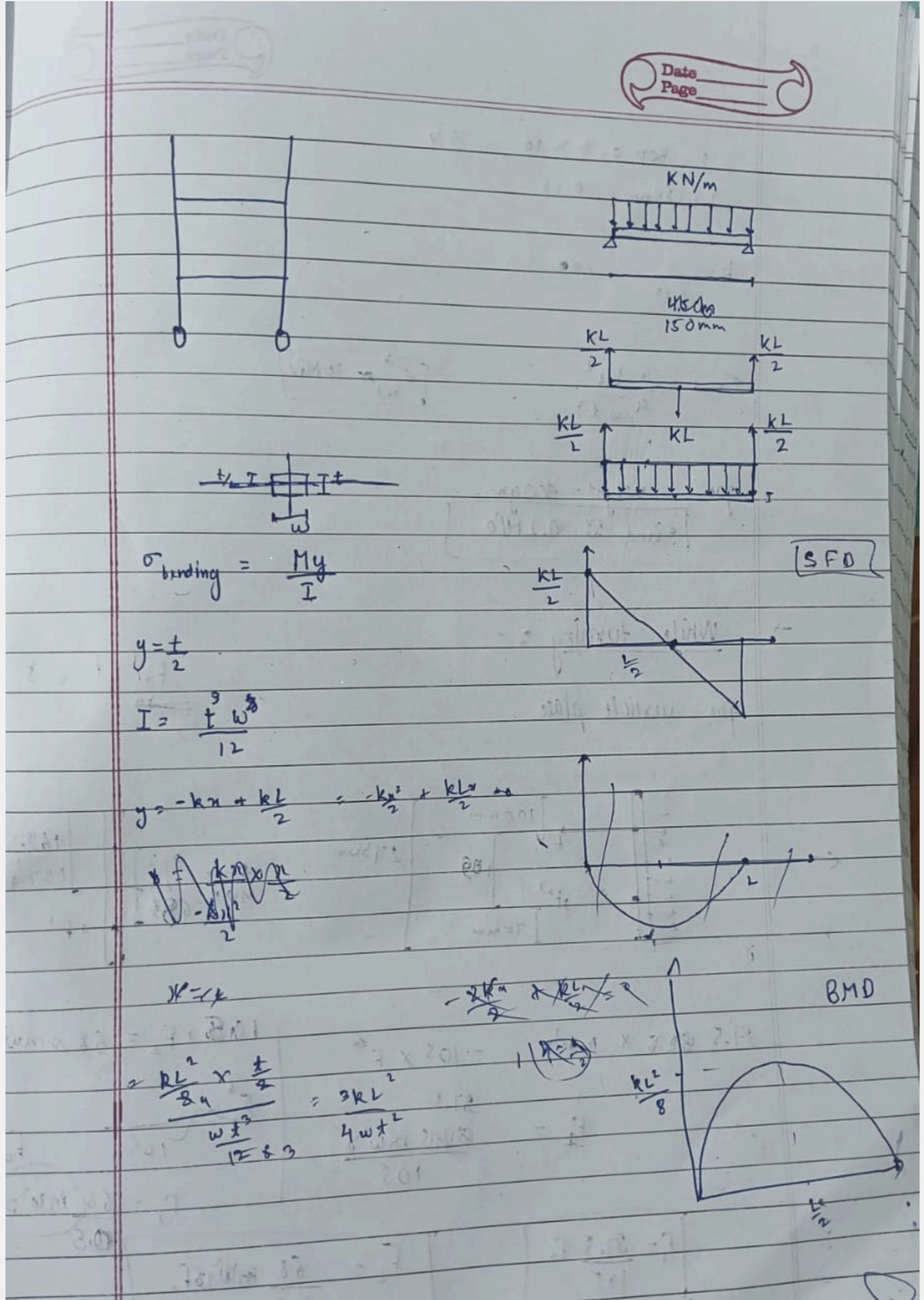
13

Design Lab

15 MAY 2025

-

Static analysis

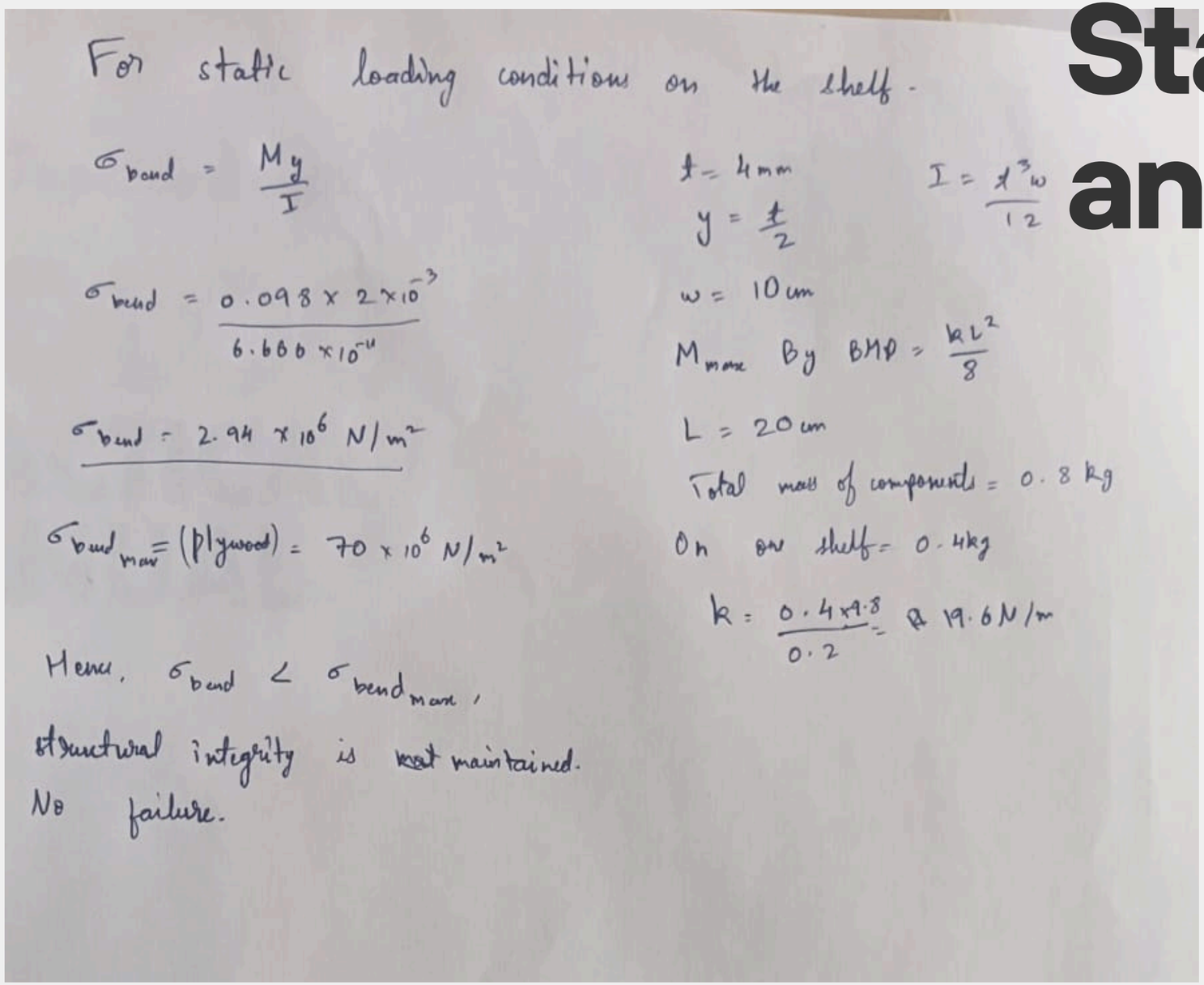


14

Design Lab

15 MAY 2025

Static analysis



For static loading conditions on the shelf.

$\sigma_{bend} = \frac{My}{I}$

$\sigma_{bend} = \frac{0.098 \times 2 \times 10^{-3}}{6.666 \times 10^{-11}}$

$\sigma_{bend} = 2.94 \times 10^{6}\ N/m^2$

$\sigma_{bend_{max}} = (Plywood) = 70 \times 10^{6}\ N/m^2$

Hence, $\sigma_{bend} < \sigma_{bend_{max}}$,

structural integrity is not maintained.

No failure.

$t = 4mm$

$y = \frac{t}{2}$

$I = \frac{t^3 w}{12}$

$w = 10\ cm$

M_{max} By BMD $= \frac{kL^2}{8}$

$L = 20\ cm$

Total mass of components $= 0.8\ kg$

On our shelf $= 0.4kg$

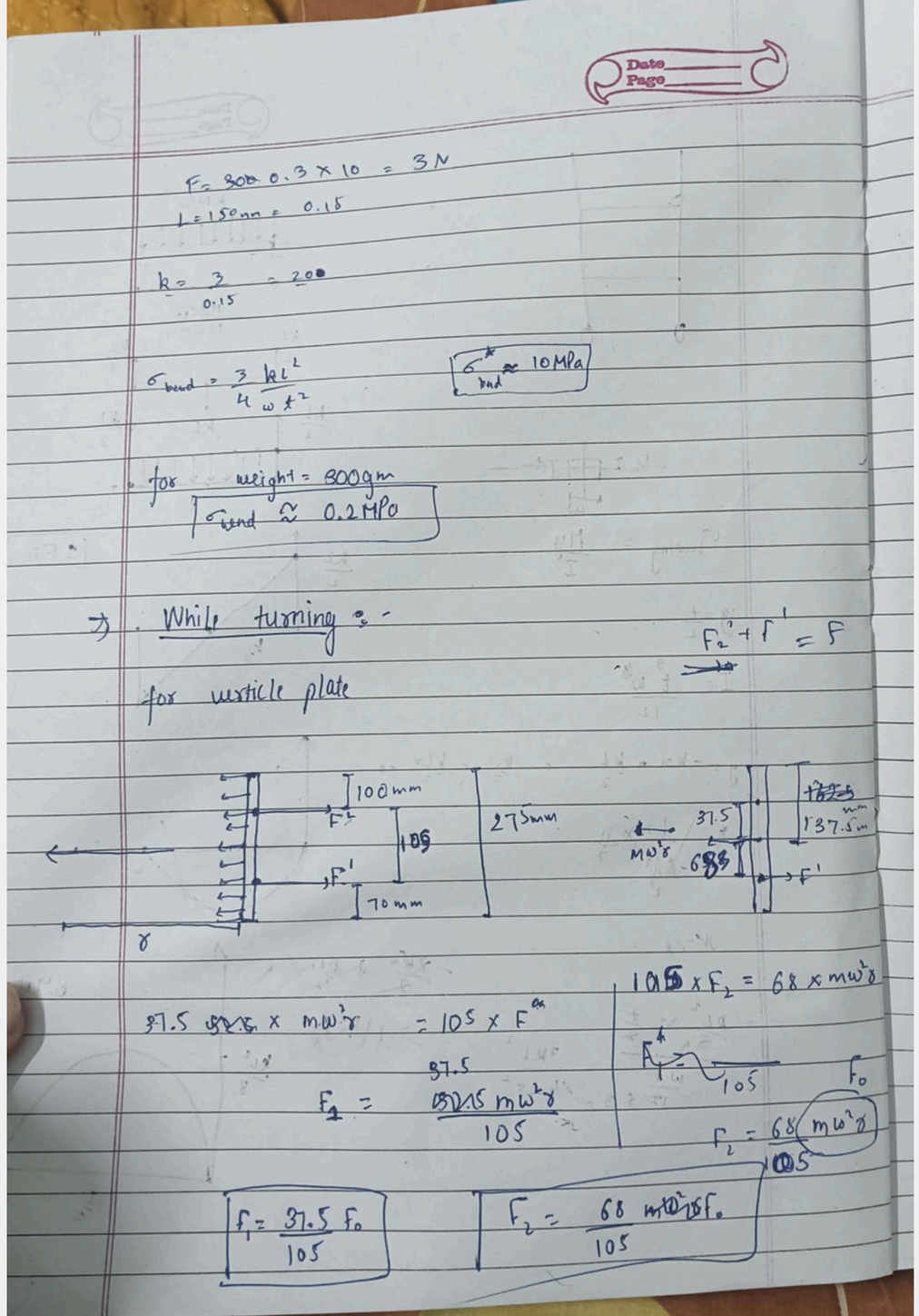
$k = \frac{0.4 \times 9.8}{0.2} \approx 19.6\ N/m$

15

Design Lab

15 MAY 2025

Analysis while turning



16

Design Lab

15 MAY 2025

THANK YOU!!