PITCH PERFECT: ME2400 Course Project Report

Name	Roll Number	Contribution
Devansh Mishra	ME22B016	Tuning + Fabrication
Srivatsan A	ME22B017	Tuning+Simulink
V. Eshwar Sai	ME22B034	Arduino+Fabrication
Aldis Daniel G	ME22B070	Arduino+Tuning

Aim:

To employ a PID control model on a physical system and fine-tune the coefficients to achieve smooth steady-state behaviour of the process model.

Objective:

To build a self-balancing beam pinned at one end and with a motor and propeller attached at the other end. The beam should:

- Remain horizontal and should be able to balance a ball bearing's ball in the center of beam.
- Be a fully contained unit except for power which should be delivered externally (for safety).
- be able to balance at any requested pitch angle within +30° to -30° of horizontal

Chassis Design and Component Placement:

Component List:

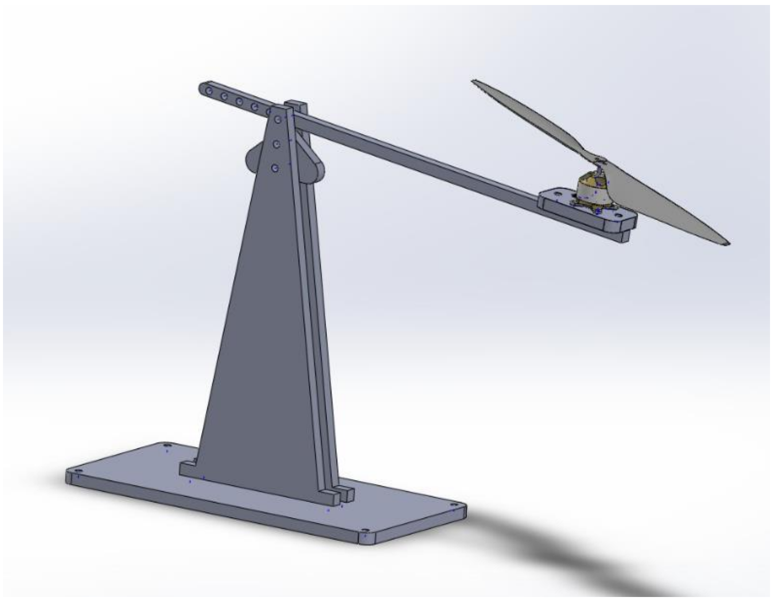
- A2212 BLDC 1000kV Brushless Motor
- 30A SimonK Electronic Speed Control
- 1045 Propeller
- MPU-6050 IMU ACCEL/GYRO 3-AXIS I2C 24QFN
- Arduino Uno SMD R3

The aim of the chassis design was to provide a sturdy base with ample space for placement of electric components and wiring. Additionally, the design should be compact and easy to transport around. Hence a wide base was chosen

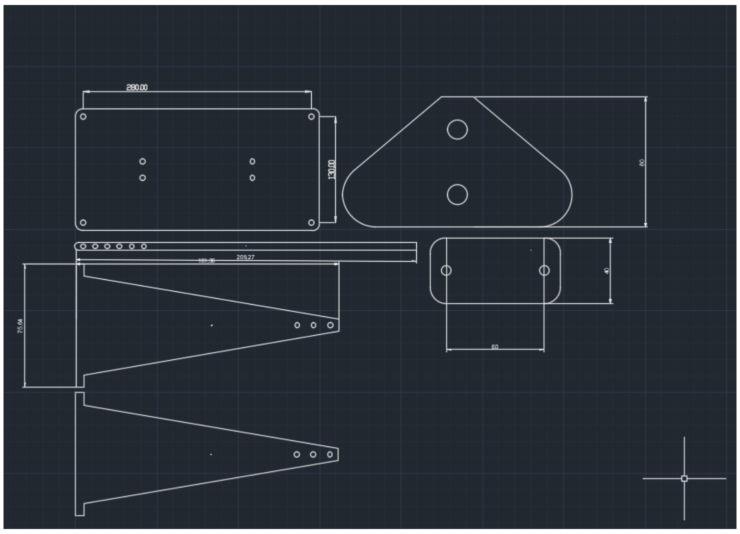
The beam was chosen to be of 200mm to:

a) Reduce overall weight to make sure the motor (which was yet to be procured at this point of time) will be able to lift the beam with its operating voltages
b) A longer beam would've led to more shaking inducing errors in the readings from the IMU

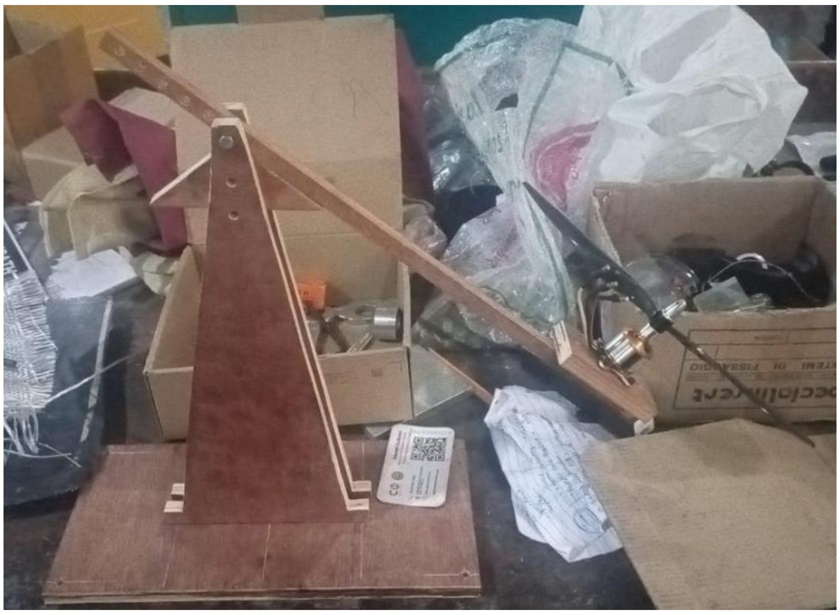
Also, counterweights were added to make the centre of mass to lie very near to the hinge point.



CAD File for CNC Machining:

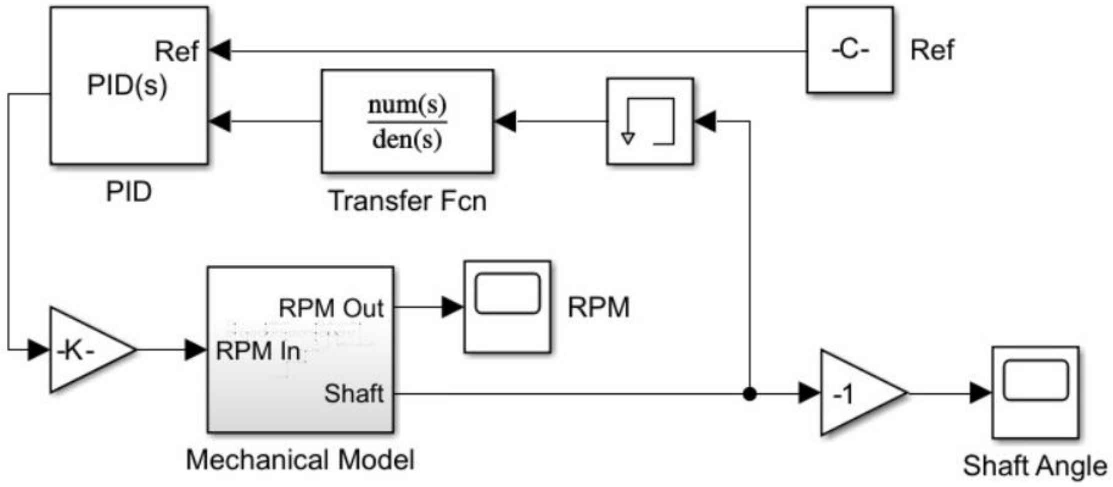


Finalised Chassis Design:

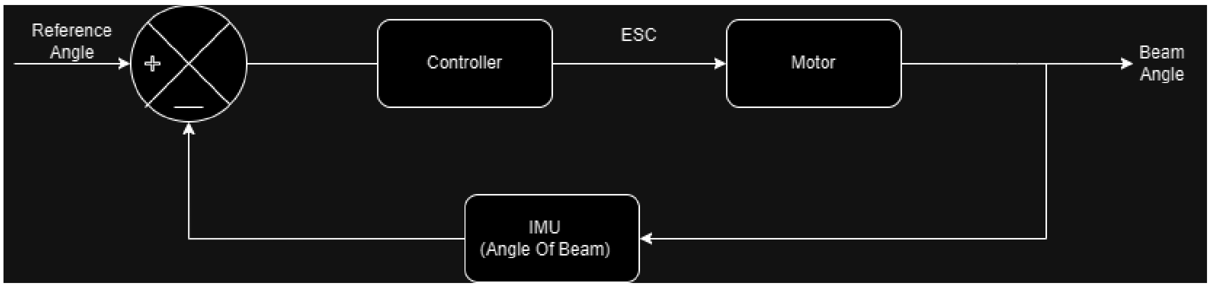


Controller design and Tuning Method:

Simulink Model:

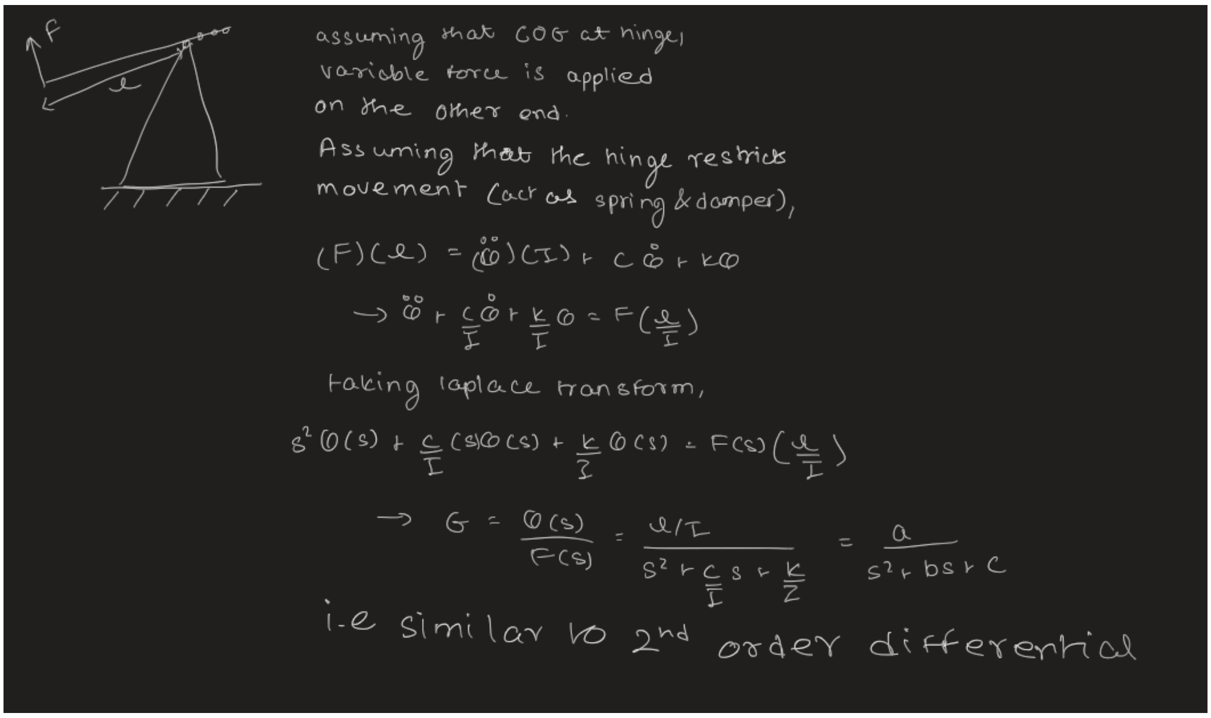


Overall Control Loop:



Transfer Function:

The rough transfer function was derived as shown:



assuming that COG at hinge, variable force is applied on the other end.

Assuming that the hinge restricts movement (act as spring & damper),

$$(F)(l) = (\ddot{\theta})(I) + c\dot{\theta} + k\theta$$

$$\rightarrow \ddot{\theta} + \frac{c}{I}\dot{\theta} + \frac{k}{I}\theta = F\left(\frac{l}{I}\right)$$

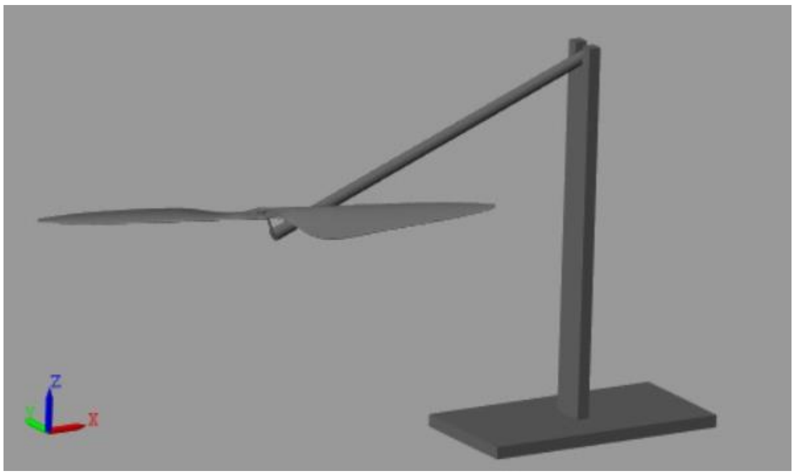
taking laplace transform,

$$s^2\theta(s) + \frac{c}{I}(s)\theta(s) + \frac{k}{I}\theta(s) = F(s)\left(\frac{l}{I}\right)$$

$$\rightarrow G = \frac{\theta(s)}{F(s)} = \frac{l/I}{s^2 + \frac{c}{I}s + \frac{k}{I}} = \frac{a}{s^2 + bs + c}$$

i.e similar to 2nd order differential

This was used in tuning a rough MATLAB SimScape model using the Simulink library. The SimScape model used is as follows:



Controller Design and Arduino Code:

While the aim initially was to have complete PID control, we realised that integral control was making the system unstable. Hence, the control was effectively PD. While beginning with relatively high Kp, we realised that reducing its value resulted in reduced oscillations, albeit settling time did increase.

The values of Kp, Kd and Ki were determined experimentally through trial and error. Fine tuning was performed based on observation of the response.

S No.	Kp	Kd	Ki
1	1	0.1	0.1
2	0.5	0.1	0
3	0.01	0.1	0
4	0.001	0.1	0
5	0.0005	0.1	0
6	0.00025	0.1	0

The control loop measured error as the difference between the reference angle and the pitch angle (as a weighted average between the output from accelerometer and gyroscope).

Since our final design employed a very lightweight beam, the running motor induced vibrations within the IMU, producing extremely noisy raw data. To this end, the error was processed as a running average to smoothen out the readings.

The entire implementation is attached with the report.

Conclusion:

The project was a wonderful introduction to mechanical control systems. The final model is able to balance itself at given angle with a small steady state error. However, the settling time was compromised to achieve this