Final Project of Mechatronics: Simulation and control of a 1 DOF system

Sharif University Of Technology, Mechanical Engineering Department

FEBRUARY 13, 2024

Mehdi Zarei

Table of Contents

Simulation and control of a 1 DOF system

1- Introduction

In this project, we aim to simulate and control a 1 Degree of Freedom (1DOF) system. Figure 1 illustrates the components of the system.

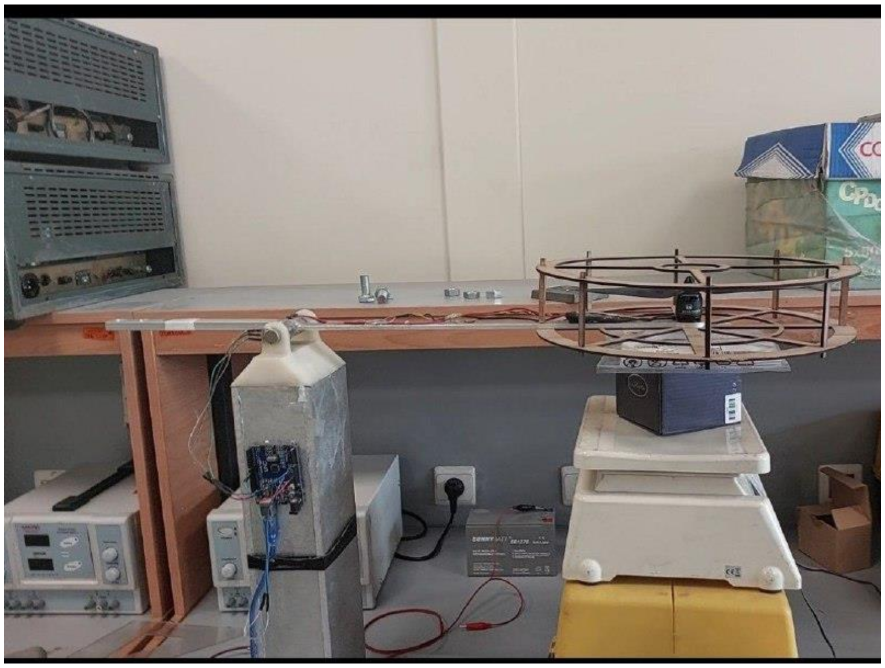


Figure 1 1DOF system

The system comprises an aluminum rod with a revolute joint mounted on a stand. It incorporates a Tiger Air motor, a propeller, an Electronic Speed Controller (ESC), a potentiometer to determine the angle of rotation of the rod, and an Arduino board responsible for controlling the rod's angle.

As the motor rotates, it generates a thrust force on the rod, influencing the rod's angle. The angle of the rod is dynamically adjusted based on this thrust force, and the magnitude of the thrust force is directly correlated with the speed of the motor.

The primary objective of this project is to achieve and maintain a horizontal state of the system through the modulation of PWM signals sent to the Electronic Speed Controller (ESC) to control the motor speed. Additionally, we aim to optimize the control of the rod's angle, striving for minimal rise time and overshoot. The robustness of the system to external disturbance forces is another key goal, ensuring that the system remains stable and in the horizontal state under varying conditions.

2- *Simulation the system in MATLAB*

For better understanding of the system's behavior, we simulated the dynamic model of the system using MATLAB. We had two options for simulation. The first option involved manually writing the equation of motion for the system, generating its state space, and subsequently implementing control. The second option was to model the entire system in SIMSCAPE MATLAB. This approach entailed creating each part of the system in SOLIDWORKS and importing them into MATLAB. With this method, there was no need to manually derive the equation of motion, and importantly, it provided a more precise simulation.

- ***Creating SolidWorks Parts***

After obtaining the dimensions and material specifications for each part, we proceed to develop a 3D model for each component using SolidWorks. This process ensures an accurate representation of individual parts, laying the foundation for a detailed and comprehensive system model.

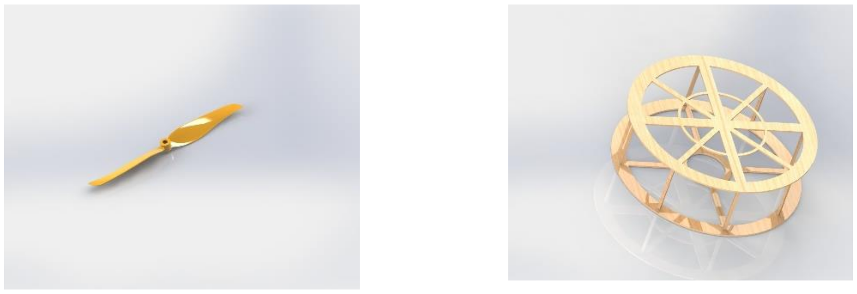


Figure 2 SolidWorks parts of the system

- ***Building the system in SIMSCAPE MATLAB***

After designing each component in SOLIDWORKS, we proceed to construct our dynamic system in MATLAB using SIMSCAPE. This entails connecting various components to each other through their respective joints, resulting in a unified and comprehensive representation of the entire system.

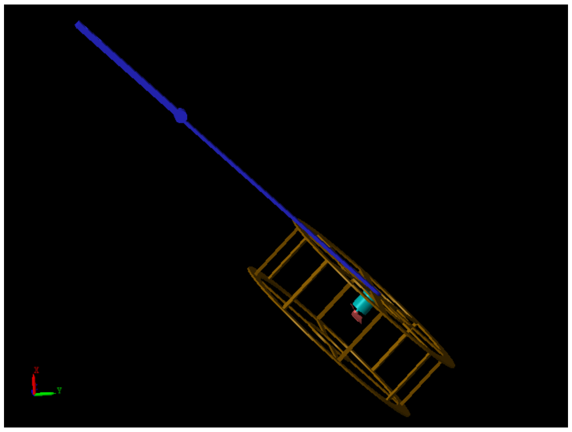
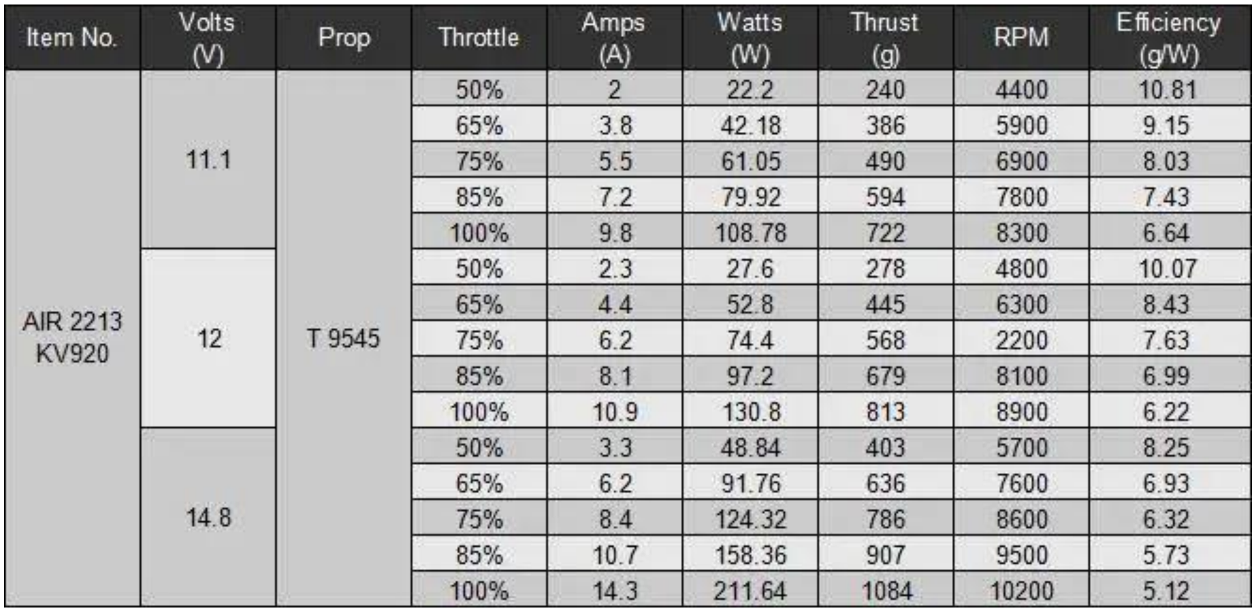


Figure 3 multibody model of the system

- ***Understanding the interaction between Electrical and Mechanical System***

To comprehend the impact of the electrical system on the mechanical system during simulation, a crucial aspect is understanding how PWM signals affect the motor and subsequently generate thrust force. In this project, we employ the AIR 2213 920 KV T-MOTOR along with a 20 A ESC and a propeller. It's imperative to note that the trust force generated by a motor is significantly influenced by its control speed circuit and the characteristics of the propeller.

For this specific motor-controller-propeller combination, we leverage valuable information provided in Figure 4, depicting a table derived from experimental data conducted in the lab. This table offers insights into the relationship between PWM signals, thrust force, and motor speed, providing essential parameters for a more accurate simulation.



Item No.	Volts (V)	Prop	Throttle	Amps (A)	Watts (W)	Thrust (g)	RPM	Efficiency (g/W)
AIR 2213 KV920	11.1	T 9545	50%	2	22.2	240	4400	10.81
			65%	3.8	42.18	386	5900	9.15
			75%	5.5	61.05	490	6900	8.03
			85%	7.2	79.92	594	7800	7.43
			100%	9.8	108.78	722	8300	6.64
	12		50%	2.3	27.6	278	4800	10.07
			65%	4.4	52.8	445	6300	8.43
			75%	6.2	74.4	568	2200	7.63
			85%	8.1	97.2	679	8100	6.99
			100%	10.9	130.8	813	8900	6.22
	14.8		50%	3.3	48.84	403	5700	8.25
			65%	6.2	91.76	636	7600	6.93
			75%	8.4	124.32	786	8600	6.32
			85%	10.7	158.36	907	9500	5.73
			100%	14.3	211.64	1084	10200	5.12

Figure 4 Electrical system Specifications

- ***Curve fitting for Trust-Speed Relationship***

Using data from Table 4 and recognizing the general relationship that thrust is often a function of the motor speed with a polynomial of power 2, we can perform curve fitting to establish the Trust-Speed curve.

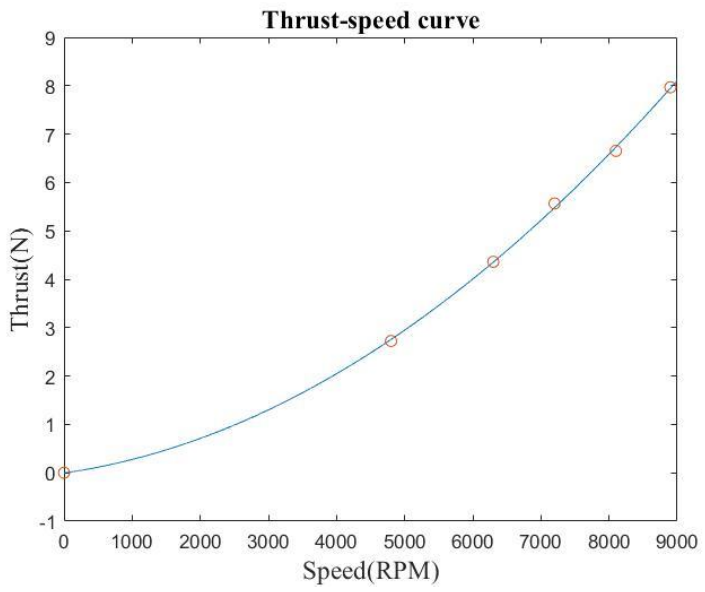


Figure 5 Trust Speed Curve

- ***Curve fitting for Speed-Throttle relationship***

In numerous motors, the relationship between PWM signal and speed is not fully linear. By leveraging the data in Table 4, we can employ curve fitting techniques to establish this intricate relationship

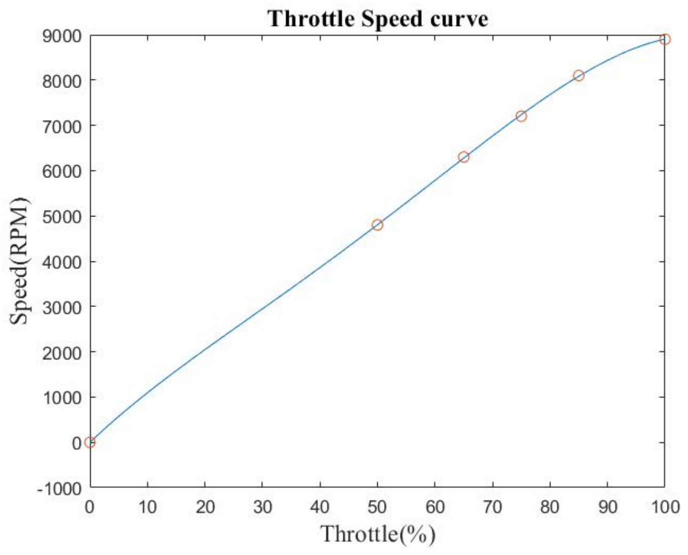


Figure 6 Speed-Throttle curve

- ***Implementing Controller for the system***

At this stage, the mechanical system is fully defined, and specifying the desired motor angle further solidifies its configuration. It is crucial to emphasize that the trust force of the motor is calculated using the Trust-Speed curve, which was established through curve fitting using the available data.

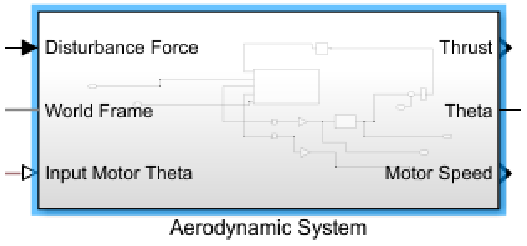


Figure 7 Aerodynamic System

The input motor angle can be precisely specified by adjusting the PWM signal. By referencing the Throttle-Speed Curve and integrating the speed over time, we can effectively determine the angle of the motor

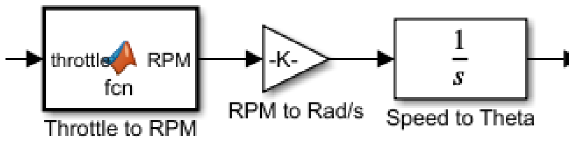


Figure 8 PWM to Motor Angle Block Diagram

Now, it's time to implement the controller, which takes the error angle of the system relative to the desired angle. The controller's role is to generate the appropriate PWM signal, which is then sent to the Electronic Speed Controller (ESC). It's crucial to note that the controller is saturated and incapable of generating signals above 100% or below 0%. This limitation ensures that the generated PWM signal remains within the acceptable range, preventing issues related to signal saturation

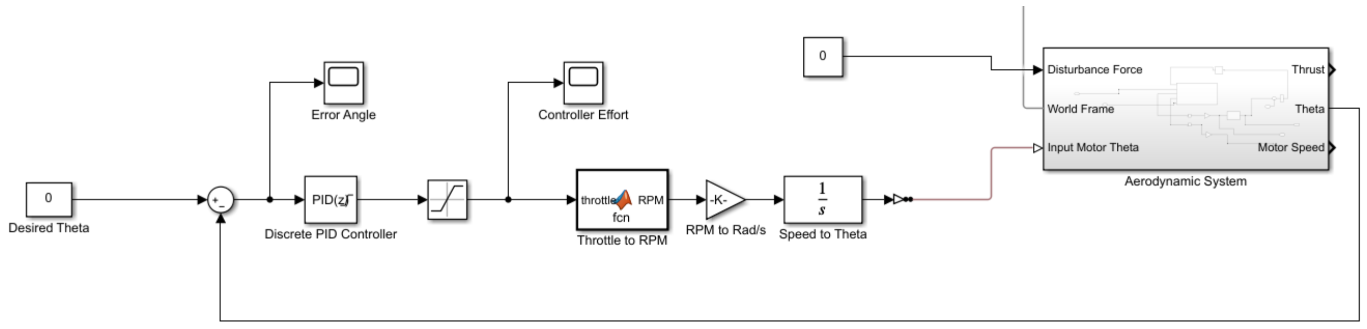


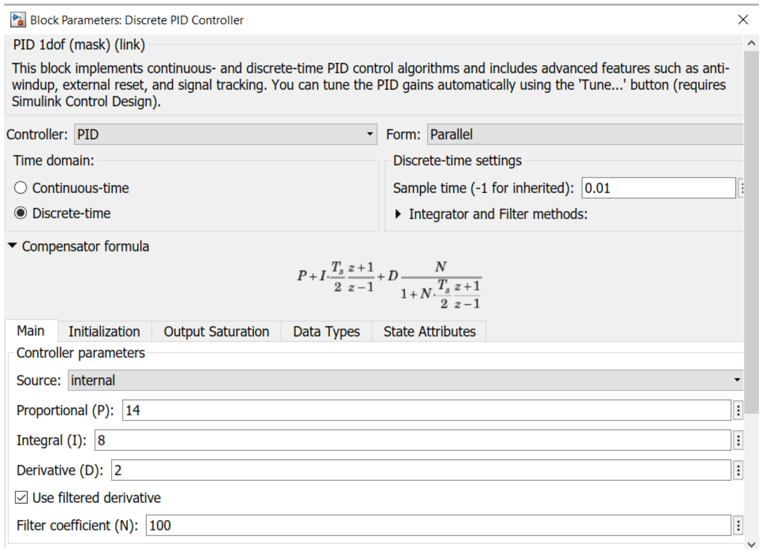
Figure 9 Closed-loop controlled System

To achieve more precise control, we engage in the optimization of PID controller parameters. This process involves fine-tuning the proportional (P), integral (I), and derivative (D) gains to ensure optimal performance. The selected PID parameters, tailored for accuracy and responsiveness, will be implemented in our system.

Furthermore, for enhanced precision, we employ a discrete PID controller with a sampling time T of 0.01 seconds. This choice ensures compatibility with Arduino's capabilities, allowing it to generate control signals within the specified time frame.

- ***Results of Simulation for the Controlled System***

Using the PID controller parameters specified in Figure 10, our simulation achieved impressive results, including a settling time of 1 second and an overshoot of 0.5 degrees. The figures presented below depict the controller effort and the angle of the controlled system, showcasing the efficacy of the PID controller in maintaining precise control and stability within the system.



Block Parameters: Discrete PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID   Form: Parallel

Time domain:
- Continuous-time
- [✓ Discrete-time]

Discrete-time settings

Sample time (-1 for inherited): 0.01

Integrator and Filter methods:

Compensator formula

$$P+I\cdot\frac{T_s}{2}\frac{z+1}{z-1}+D\frac{N}{1+N\cdot\frac{T_s}{2}\frac{z+1}{z-1}}$$

Controller parameters

Source: internal

Proportional (P): 14

Integral (I): 8

Derivative (D): 2

[✓] Use filtered derivative

Filter coefficient (N): 100

Figure 10 Desired PID parameters

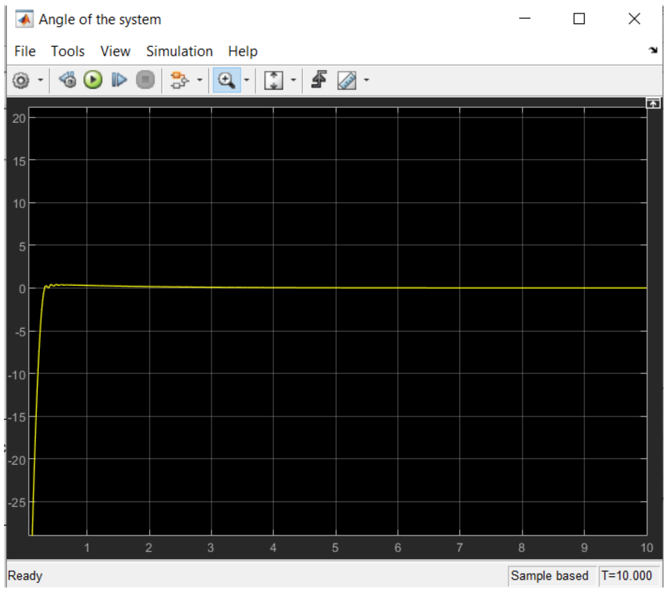


Figure 11 Angle of the System

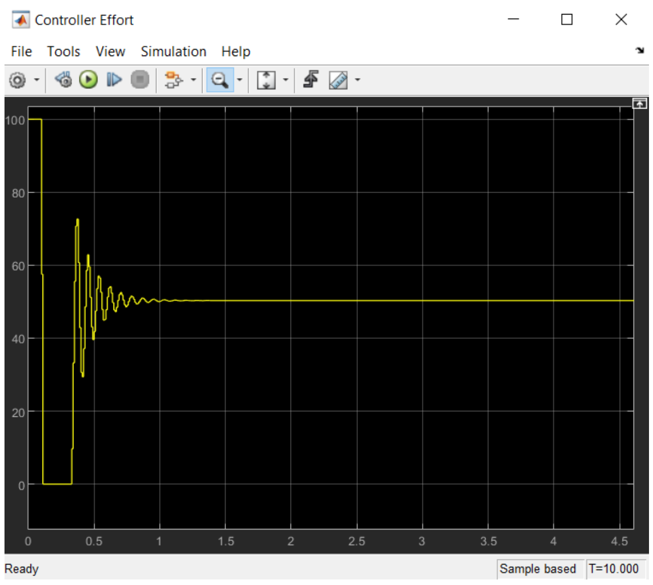


Figure 12 Controller Effort

- ***Converting Z-Transform to Difference equation***

After implementing PID controller in Z domain , we should find difference equation of it to implement in Arduino by expanding the Z transform controller and taking inverse Z we can find difference equation as follows:

$$\frac{U(Z)}{E(Z)} = P + I\frac{T_S}{2}\frac{(Z+1)}{Z-1} + D\frac{N}{1+N\frac{T_S}{2}\frac{Z+1}{Z-1}} \quad (eq\ 1.1)$$

Where $U(Z)$ is controller effort and $E(Z)$ is the corresponding error. By specifying variables as follows :

$$P = 14, I = 8, D = 2, T_s = 0.01, N = 100$$

The controller transfer function becomes:

$$\frac{U(Z)}{E(Z)} = \frac{221\,Z^2 - 428\,Z + 207}{1.5\,Z^2 - 2\,Z + 0.5}$$

Consequently, the corresponding difference equation is expressed as:

$$U(t) = \frac{2\,U(t-1) - 0.5\,U(t-2) + 207\,e(t-2) - 428\,e(t-1) + 221\,e(t)}{1.5} \quad (\,eq\ 1.2)$$

Utilizing eq1.2 allows us to implement our discrete controller in Arduino in the form of a difference equation

3. *Real implementation on the system*

- ***Reading the angle of the system by Potentiometer***

To implement our computed algorithm on the real system, we need to accurately measure the angle (θ) of the system. This involves reading the theta value using a potentiometer. By collecting experimental data utilizing tools such as the Protractor Android app to measure the system's angle and performing potentiometer analog reads on Arduino, we can curve-fit an optimal relationship between potentiometer voltage and the angle of the system.

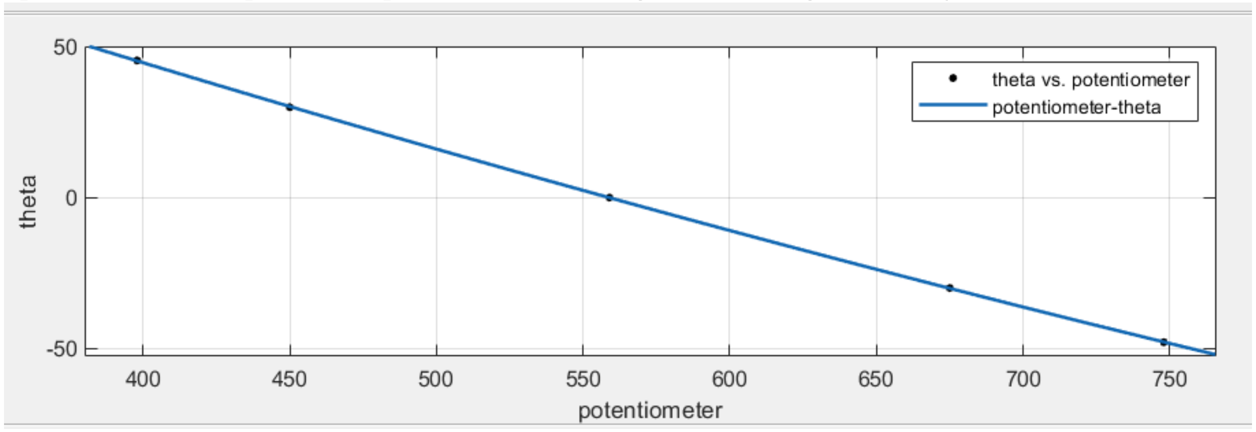


Figure 13 Calibrating potentiometer to read the angle of the system

- ***Implementing tunned PID on the system***

Despite simulating the system's behavior and implementing the tuned PID control in simulation, unfortunately, it did not function as expected when applied to the real system. The malfunction could arise from various factors:

1- Despite having nominal points for throttle-to-speed and speed-to-force relationships, our system proved highly sensitive to any alterations in these two transfer functions. It's possible that our fitted curves for these functions didn't precisely match those of the real system.
2- The tuned PID controller might be sensitive to changes in its parameters, and errors could arise during the computation of its difference equation due to numerical precision issues.

These two factors were the primary reasons for the controller's inability to effectively control the system. Additionally, there are other influencing factors such as noise in the potentiometer, variations in the time stamps for delivering controller effort to the system, potential errors in the SOLIDWORKS model and so on.

To address these issues, we considered two options:

1- Accurately measuring the throttle-to-force relationship of the actual system by varying the throttle and using a balance to measure the thrust force. Then, adjusting the simulation relationship to match the actual one and tuning the PID controller in the simulation before implementing it in reality.
2- Tuning the PID controller directly on the real system by adjusting PID parameters.

Due to time constraints and, importantly, a lack of accurate equipment for the first option, we chose the second option.

Additionally, the system exhibited a high dynamic response due to the lack of mass, which made controlling the system more challenging. To address this issue, we added extra mass to the other side of the aluminum rod to facilitate balancing the system

After several attempts with PID parameters, we were able to identify the following parameters that achieved a settling time of under 10 seconds with a maximum overshoot of 3 degrees

$KP = 0.8\ , Ki = 0.4\ , Kd = 0.25, N = 10, Ts = 10\ ms$

difference equation of the Discrete Controller:

$U(t) = 0.9523(3.3421e(t) - 6.599\mathrm{e}(\mathrm{t}-1) + 3.2581\mathrm{e}(\mathrm{t}-2) + 2\mathrm{U}(\mathrm{t}-1) - 0.95\mathrm{U}(\mathrm{t}-2))$

You can find the video of the controlled system in the zip file. Unfortunately, I was unable to record the actual angle-time curve during the implementation of the Controller. Additionally, when I attempted to record the real-time angle of the system versus time in the next session, the system encountered some issues. However, you can find the full controlling session video in the zip file. All MATLAB simulations, SOLIDWORKS parts, system specifications and the Mathematica file for finding the difference equation based on the PID parameters are included in the zip file.

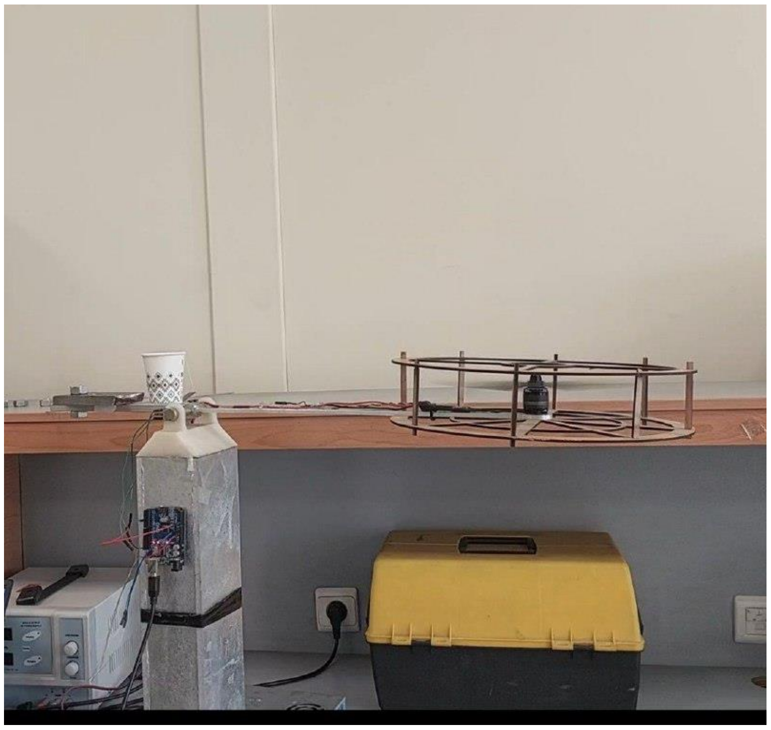


Figure 14 Controlled System