Single Axis PID Controller for UAV

1st Mirza Wasay Baig
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
mwbaig.bee20seecs@seecs.edu.pk

2nd Muhammad Abdullah Sohail
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
masohail.bee20seecs@seecs.edu.pk

3rd Muhammad Ibrahim
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
miibrahim.bee20seecs@seecs.edu.pk

4th Wajih Hassan Raza
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
wraza.bee20seecs@seecs.edu.pk

5th Muhammad Moiz
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
mmoiz.bee20seecs@seecs.edu.pk

***Abstract*—A focused effort towards designing and implementing an effective control system for a drone's brushless motors along a single axis using the Proportional-Integral-Derivative (PID) control technique. This project aims to develop a robust control system that regulates the speed of the brushless motors along the specified a xis. T he P ID c ontrol a lgorithm, a well-established method in control engineering, is utilized to achieve accurate motor control, compensate for disturbances, and enhance overall performance.**

***Index Terms*—Control Systems, UAV, PID, Controller**

I. Introduction

Brushless motors are widely used in drone propulsion systems due to their superior performance characteristics. However, achieving precise control over these motors is essential for stable flight, accurate positioning, and responsive maneuverability. Conventional control systems often lack the necessary robustness and responsiveness to handle varying flight conditions, external disturbances, and dynamic load changes. This results in reduced flight stability, compromised maneuverability, and limited control accuracy, negatively impacting the overall performance of the drone.
We have designed and implemented a PID controller for a multi-copter UAV with a single axis, and this can be extrapolated to multiple design types such as quadcopter, hexacopter, and coaxial.

Identify applicable funding agency here. If none, delete this.

II. Literature Review

Before moving on to the functionality of our one axis drone, we must understand the three main parameters of PID controller.

A. Proportional Control

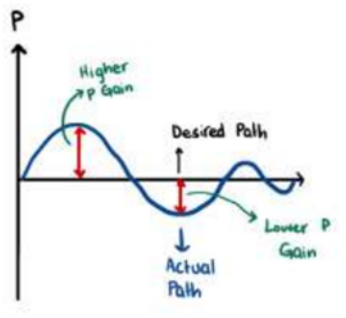


Fig. 1. Proportional Control

The proportional control as its name suggests, provides a gain output which is proportional to the magnitude of the error calculated by the comparison of desired operation and the actual operation. Higher the value of error, Higher the contribution of the P parameter in the controller.

B. Integral Control

The integral control aims to minimize the steady state error of the system. Even if there are no oscillations in the

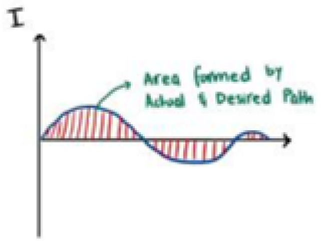


Fig. 2. Integral Control

system, the system might stabilize itself in a steady state stil being slightly deviated from its desired operation. In integra control, the error value is being added to the error of the previous iteration (calculating area formed by the deviations, which makes the error keeps accumulating Hence, in normal operation, the parameter "I" contributes much less than P and D parameters and provides a low value of gain at its output.

C. Derivative Control

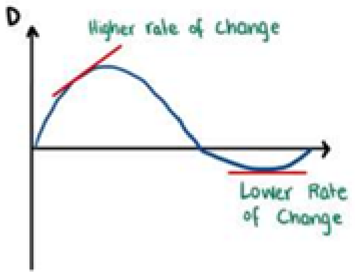


Fig. 3. Integral Control

It may be assumed that parameter P is alone enough to eliminate the errors of a system. However, this is not the case. There may be a scenario when the system keeps on progressing forward, however with a high magnitude of oscillations. Here the D parameter comes into play. This parameter provides a gain output which depends on the rate of change of the error. Thus, it aims to minimize the fluctuations in the system.

III. Proposed Method

A. Block Diagram

Firstly, we designed the block diagram of the controller project which would enable us to better understand the project. For the control system, an initial reference angle is set for the

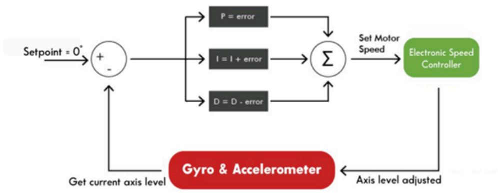


Fig. 4. Simplified Block Diagram of System

drone arm. If any distortion is produced in the drone arm, the gyro sensor value is fed as an input to the PID block which generates the desired motor speeds for the two BLDC motors which try to restore the original arm position. The motor speed value depends on how much distortion is produced in the arm.

B. Simulation

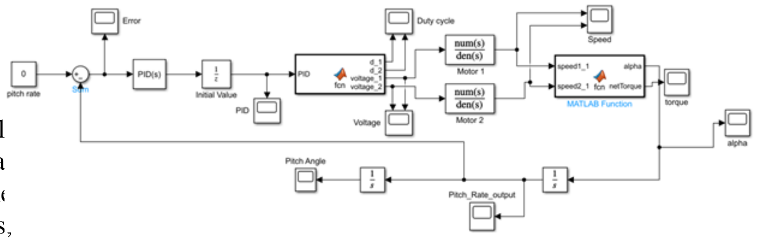
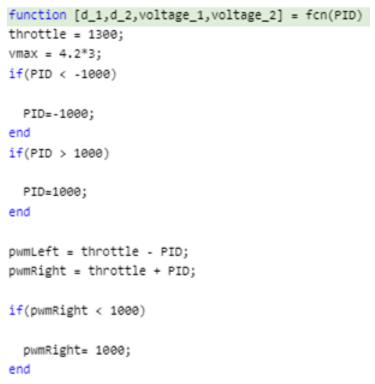


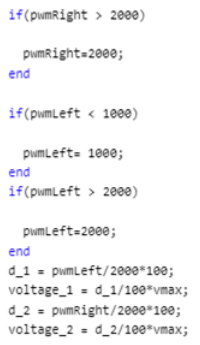
Fig. 5. Simulink Model of the Control System

As you can see from the simulation, the reference pitch angle of the drone arm is set to 0. We have designed a closed system which is responsible for maintaining this desired angle. Initially, the initial error value is generated using the unit delay block. Then, this value is fed to the PID block which outputs the error gains. Depending on the error gains, the duty cycle and voltage values of the motor 1 and 2 are calculated. The inside calculation of the values is given in below:



```
function [d_1,d_2,voltage_1,voltage_2] = fcn(PID)
throttle = 1300;
vmax = 4.2*3;
if(PID < -1000)

  PID=-1000;
end
if(PID > 1000)

  PID=1000;
end

pwmLeft = throttle - PID;
pwmRight = throttle + PID;

if(pwmRight < 1000)

  pwmRight= 1000;
end
```

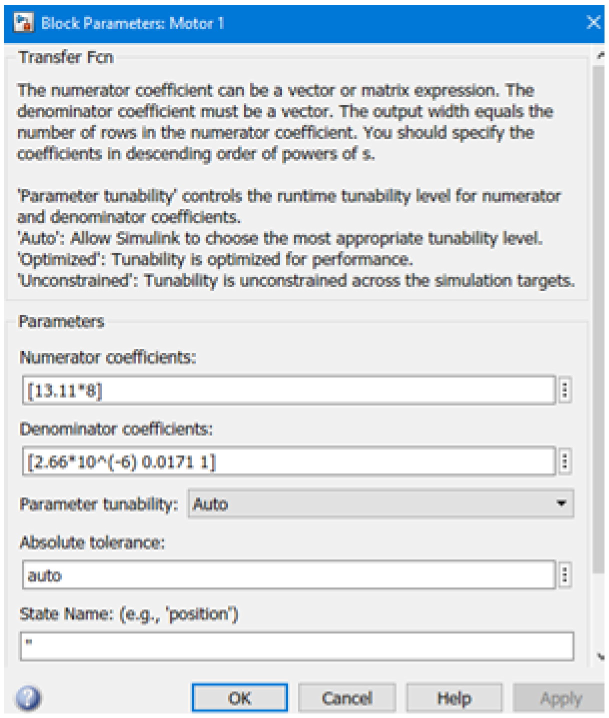
Fig. 6. Code inside MATLAB function block



```
if(pwmRight > 2000)

  pwmRight=2000;
end

if(pwmLeft < 1000)

  pwmLeft= 1000;
end
if(pwmLeft > 2000)

  pwmLeft=2000;
end
d_1 = pwmLeft/2000*100;
voltage_1 = d_1/100*vmax;
d_2 = pwmRight/2000*100;
voltage_2 = d_2/100*vmax;
```

Fig. 7. Code inside MATLAB function block - Continued

These values are passed to a motor transfer function block which produces the motor speed depending upon the voltage values. The transfer function values for the motors are given in Figure 8.



Block Parameters: Motor 1

Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

'Parameter tunability' controls the runtime tunability level for numerator and denominator coefficients.
'Auto': Allow Simulink to choose the most appropriate tunability level.
'Optimized': Tunability is optimized for performance.
'Unconstrained': Tunability is unconstrained across the simulation targets.

Parameters

Numerator coefficients:
[13.11*8]

Denominator coefficients:
[2.66*10^(-6) 0.0171 1]

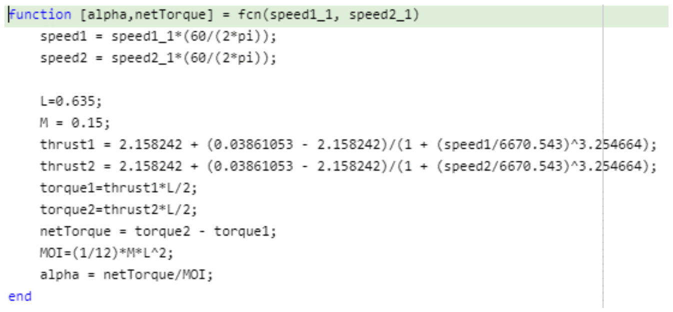
Parameter tunability: Auto

Absolute tolerance:
auto

State Name: (e.g., 'position')
''

Fig. 8. BLDC Motor Transfer Function

The output motor speeds are fed to a function block which produces angular acceleration and torque as an input by using these values. In calculation, first the desired thrust values of the motors are calculated and then using these thrust values net torque is calculated. We then calculated the moment of inertia for the straight rod. In the end, we determined the acceleration by dividing the net torque with the moment of inertia. The function block is attached in Figure 9.



```
function [alpha,netTorque] = fcn(speed1_1, speed2_1)
    speed1 = speed1_1*(60/(2*pi));
    speed2 = speed2_1*(60/(2*pi));

    L=0.635;
    M = 0.15;
    thrust1 = 2.158242 + (0.03861053 - 2.158242)/(1 + (speed1/6670.543)^3.254664);
    thrust2 = 2.158242 + (0.03861053 - 2.158242)/(1 + (speed2/6670.543)^3.254664);
    torque1=thrust1*L/2;
    torque2=thrust2*L/2;
    netTorque = torque2 - torque1;
    MOI=(1/12)*M*L^2;
    alpha = netTorque/MOI;
end
```

Fig. 9. Code inside the second MATLAB function block

We used an integral block to get pitch angle as an output from the pitch rate.

IV. Hardware

After getting successful simulation results, we decided to move on to the hardware implementation. The figures below shows the main components used in the hardware:

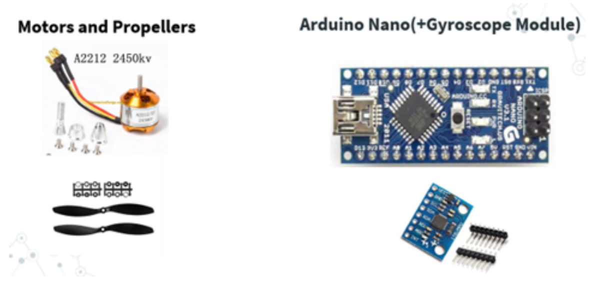


Fig. 10. Main hardware components used

- Gyro Sesnor: For monitoring the axis level.
- Arduino Nano: For receiving sensor input and controlling motor speed.
- BLDC Motors: Act as a system plant / actuator.

V. Results and Discussions

We were successfully able to implement our one-axis PID controller on both simulation and hardware. We will now discuss our results of our design.

A. Simulation Results

We were able to observe the changes in various parameters of the model on Simulink.

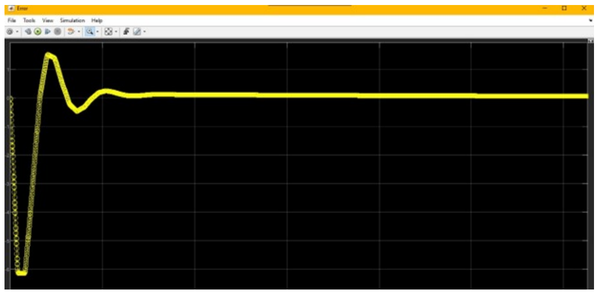


Fig. 11. Error Graph on Simulink

The graph in Figure 11 shows the error changes in time. The error is calculated by the difference in the calculated and the expected pitch angle (set at the start of the simulation). As visible in the graph, our error oscillates around the origin until our PID controller sets our output pitch angle to the same value as expected, and our error becomes zero.

Figure 12 shows the graph of Motor speeds on Simulink, it can be observed that the initial speed of both the motors is not equal. This is because our drone controller is oscillating in both directions and is not stable initially. The higher motor speed at one side means that our drone is tilted towards its direction. However, it can be observed that the motor speed for both motors become equal, which shows that our PID controller has stabilized our one axis drone at the 0 degree angle.

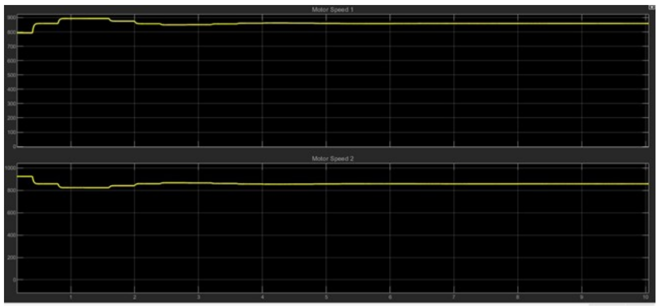


Fig. 12. Graph of Motor Speeds on Simulink

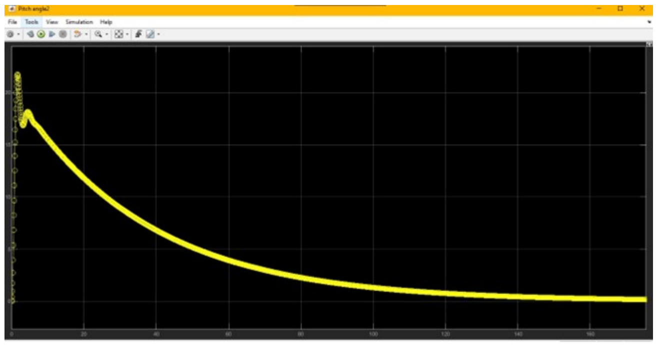


Fig. 13. Output pitch angle on Simulink

The graph in Fugure 13 shows the pitch angle graph observed on our Simulink model. We can clearly observe that with time, our pitch angle reduces to the desired output value, which is set to 0 degrees. Thus, we can conclude that our Simulink model can stabilize our drone at this angle.

B. Hardware Results

After completing our hardware implementation, we verified the operation of hardware through Arduino. Through our Arduino code, we set our PID gain values and observed the results. After a few hits and trial iterations, we were able to compute our desired PID gain values. Using those parameters, we stabilized our PID drone on hardware, which was observed in the demonstration.

VI. Challenges Faced

We faced some decent challenges on the journey of completing this project both during simulation and hardware. However, we were successfully able to implement feasible solutions to overcome these challenges. The main challenges faced were:

- Choice of Variable of Analysis: While making the simulation, first we performed our analysis with the pitch angle and were not getting the desired results. Later, we came to know that the gyro sensor provides an output of pitch rate instead of pitch angle. Thus, by changing the variable of analysis to pitch rate we were able to get the correct results.
- Choice of Motor Speed: The motors we used were 2450kV and after providing them full throttle, we burnt an electronic speed controller (ESC). Thus, we had to limit the speed of the motor for correct functioning.
- Faulty Components: We also faced difficulties with the manufacturing fault of the components. The motors were faulty as they had a lot of friction with the mounting screws. We had to make the use of washers to make gap between screws and motors.

VII. Conclusion

Overall, this project provided summed up all the concepts taught to us in Linear Control Systems. We were able to fully grasp the understanding of how to design a controller for a given plant. Our plant was the BLDC motor and first of all we performed an open loop analysis to determine its step response. Moving on, we specified our design requirements and got a controller transfer function performing our analysis on MATLAB Control System Designer. After modelling our system on Simulink, we moved on to implement it on hardware which provided successful results.

Acknowledgment

We would like to extend our gratitude to Dr. Farid Gul and Sir Yasir Rizwan for teaching us the concepts of Linear Control Systems in an easy to understand and efficient manner which played a key role in successful implementation of this project.