ADSUJSR 13 (2) (2025) 59-69
Adamawa State University Journal of Scientific Research (ADSUJSR)
ISSN: 2705-1900 (Online); ISSN: 2251-0702 (Print)
https://www.adsujsr.adsu.edu.ng

IMPROVED PID CONTROLLER FOR DRONE HEIGHT STABILIZATION

YAKUBU I*. and DANLADI A.
Department of Pure and Applied Physics
Adamawa State University, Mubi.
**Corresponding author's email: ilisonirj1@gmail.com*

Received in January 2026
Accepted in January 2026

Abstract
This research entails the development of an Improved Proportional Integral Derivative (PID) controller for drone height stabilization. Modern-day drones usually encounter stabilization issues because of environmental forces acting on the drone. These forces have the capacity to launch the drone into an uncontrollable roll, pitch, and yaw. Also, these drones are faced with height stability issues. As a result, the drones are equipped with a lot of expensive height sensors and software algorithms to keep them stable. This research is therefore carried out to show that a less expensive sensor gyroscope alone is sufficient to stabilize the height of the drone, provided that appropriate software is written. In accomplishing this research, a PID controller is developed using the Zeigler-Nicholas method. A three-dimensional model of the drone was developed using SIMULINK in MATLAB. Disturbance is the form of a push force that is also developed in SIMULINK, and at the end of the simulation, the drone is seen to handle a vertical push force along the z-axis of 50N in both directions (positive and negative z-axis) without being thrown into instability. Finally, this research has shown that the height can be stabilized without employing expensive sensors.

Keywords: PID Controller, Height Stabilization, Drone, Disturbance, Controller

Introduction

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, have become increasingly important in both civilian and military applications due to their versatility, maneuverability, and ability to operate in environments that may be unsafe or inaccessible to humans. They are widely employed in surveillance, search and rescue, agriculture, delivery services, and environmental monitoring (Mahony, Kumar, & Corke, 2012; Shakhatreh et al., 2019). Despite their potential, UAVs are inherently nonlinear and unstable systems that require precise control mechanisms to maintain stability, especially during hovering, altitude tracking, and trajectory following.

One of the most used controllers in UAV stabilization is the Proportional–Integral–Derivative (PID) controller due to its simplicity, robustness, and ease of implementation (Astrom & Hägglund, 2006). PID controllers adjust system responses by minimizing the error between the desired setpoint and the actual output, making them suitable for flight control tasks. However, conventional PID controllers face several limitations when applied to drones, including sensitivity to parameter variations, external disturbances such as wind, and difficulties in gain tuning for nonlinear and multi-input multi-output (MIMO) dynamics (Bouabdallah, Noth, & Siegwart, 2004; Castillo, Lozano, & Dzul, 2005).

To overcome these challenges, recent research has focused on improving PID controllers by integrating advanced techniques such as adaptive control, fuzzy logic, neural networks, and optimization algorithms.

These approaches aim to enhance the robustness and adaptability of UAV stabilization systems while maintaining the simplicity of PID structures (Ming et al., 2020; Sánchez, Vázquez, & García, 2018). Improved PID controllers can significantly reduce overshoot, enhance disturbance rejection, and improve trajectory tracking performance, thereby increasing the reliability and efficiency of drones in practical applications.

However, in this research work, the dynamic assembly of the drone will be developed using the powerful computing capability of MATLAB's simscape multibody software. The Simscape multibody software is a graphical programming tool used to dynamically model multibody systems. With this tool, the need to derive mathematical equations of motion for the drone will be eliminated. This has the advantage of allowing researchers from various fields, other than mathematics and engineering, to venture into research on drones since they do not possess mathematical skills. This is basically the motivation for this research work.

At the end of this research work, the complete drone will be dynamically assembled and simulated by importing computer-aided design (CAD) models of a drone in the form of Standard for the Exchange of Product data (STEP) files. Next, an improved PID controller will be designed to stabilize the altitude of the drone whenever there is a small disturbance.

MATERIALS AND METHODS

This work employed Computer Aided Design (CAD) files of different components of the drone, which were imported as Standard for the Exchange of Product data (STEP) files into the Simscape Multibody software in MATLAB. The different components of the drone include the four arms, the propellers, the brushless motors, and the top and bottom plates.

A personal computer with MATLAB release 2016a installed, version R2016a (9.0.0.341360) was used for simulation purposes. The Simscape version 4.0 multi-body toolbox was also essential and was installed with MATLAB for this research work.

The following steps are employed to actualize throne control height stabilization.

Importing components of the drone

Firstly, the top plate of the drone was imported. This was accomplished by opening the Simscape Multibody software in MATLAB and invoking the Solid Block Graphics user interface (GUI). In the geometry section of the GUI, the shape of the solid was set to "from file," the file type was set to STEP, and the location of the STEP file was then provided (as a path or directory) in the "file name" section. The mass of the top plate was set at 15g.

After importing the drone top plate, it was then placed in a three-dimensional Cartesian coordinate system using the rigid transform block. The rigid transform GUI was invoked by typing "rigid transform" and pressing Enter. In the rigid GUI, the exact location of the top plate in the three-dimensional space was specified.

The offset z-axis is set to 3.9. This is because in this design, the width of the drone arm was exactly 3.9 cm, and the bottom plate will be placed under the arm. This means that the z-distance between the top plate and the bottom plate would be 3.9 cm, and this distance will accommodate the arms of the drone.

Therefore, the top plate model was obtained by joining the solid block and the rigid transform block.

The conn1 block was used for interfacing with the arms of the drone and the bottom plates.

The model for the drone's bottom plate was developed using the same steps as those used for the top plate. This time, the rigid transform block was not utilized. This decision was made because the z-distance between the top and bottom plates had already been accounted for in the rigid transform applied to the top plate model. The Simulink model for top and bottom plate is in Figure 1 below and Figure 2 is the Three-dimensional rendition of the top and bottom

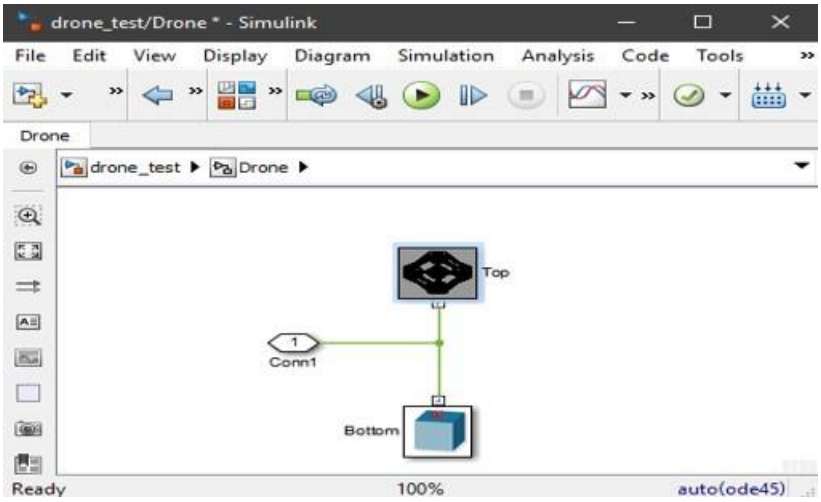

Figure 1: Simulink model for the top and bottom plate

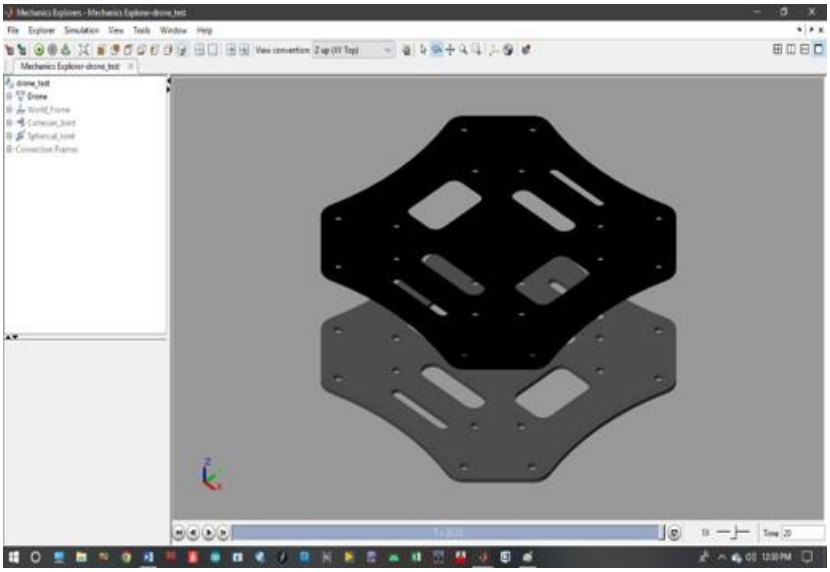

Figure 2: Three-dimensional rendition of the top and bottom

The drone arms were modeled using Simulink (like the Simscape Multibody software). This process followed the same steps used for assembling the top plate, except that in the "file name" section of the GUI, we provided the appropriate path or directory to the STEP file of the drone arm. A rigid transform block was then used to position the drone arm correctly between the top and bottom plates.

The remaining three arms of the drone were modelled in like manner, the only difference being in the rigid transform specification where each arm was specified such that it fits at a 90-degree spacing between each other around the z axis.

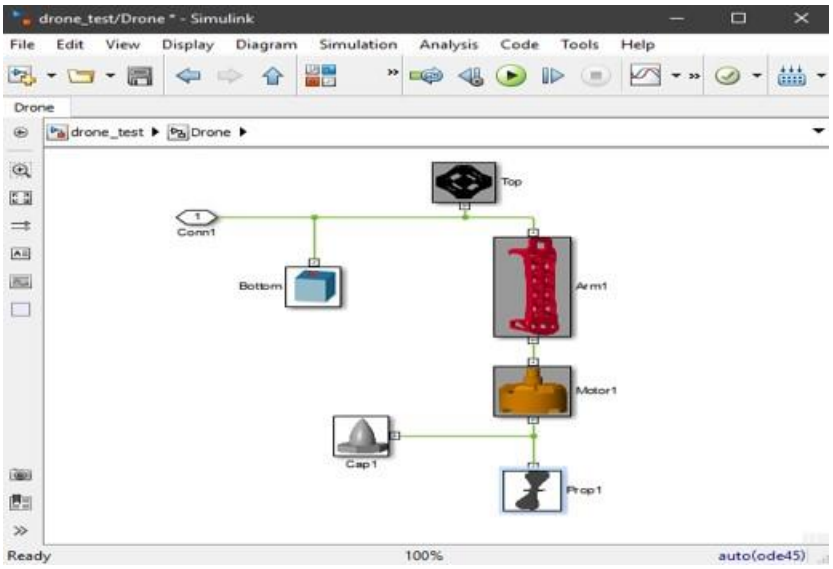

Figure: 3 show the Simulink model of the drone with one arm

The brushless motors, propellers, and the cap that fastens the propellers to the brushless motors were imported and assembled by specifying the paths or directory of their STEP files and using the rigid transforms to specify their positions in three-dimensional space such that they fit according to the dimensions of the drone. The remaining part of the drone was completed by replicating the arm design of figure 3, three times. The only difference is in the rigid transform specification. The rigid transform was specified such that the arms, the brushless motor, and the caps were all equally spaced at 90 degrees from each other

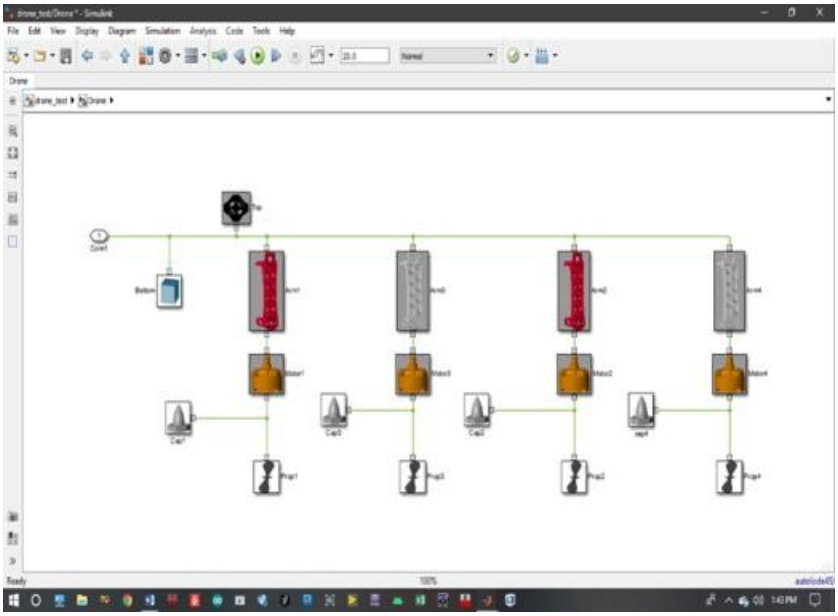

Figure 4: Simulink assembly

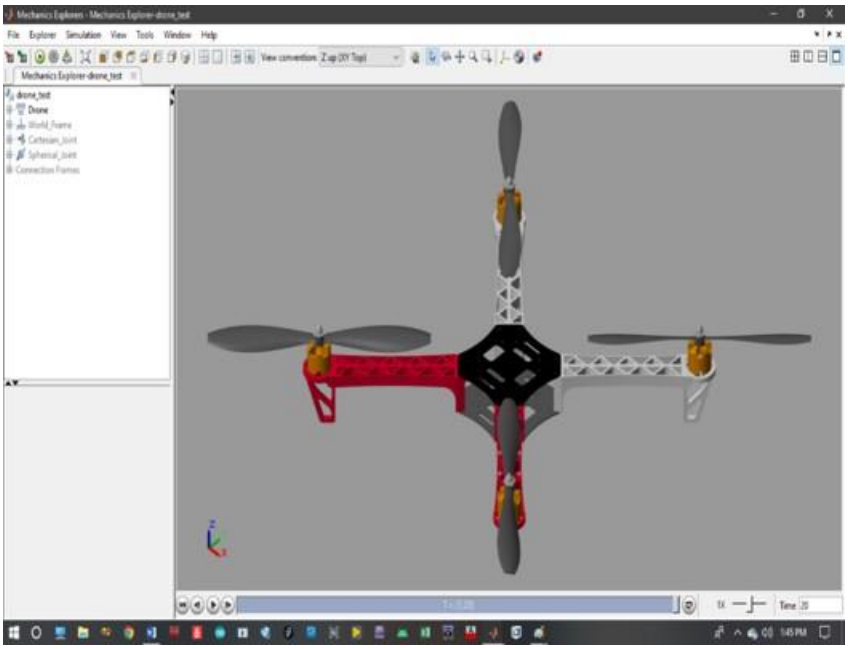

Figure 5: Three-dimensional rendition of the assembled drone

Attaching the Simulink or Simscape multibody joints

A drone possesses six degrees of freedom (DoF) of motion. A drone was therefore capable of rotating about an x axis (roll), rotating about a y axis (pitch), rotating about a z-axis (yaw), and translating towards all three axes of the Cartesian coordinate system. In order to give the drone that was assembled in Figure 5 these characteristics, Simulink joints were attached to appropriate sections of the Simulink model. The joints that will be appropriate in this research work, to give the drone the six DoF of motion were the spherical joints and the Cartesian joints of the Simscape multibody software. The spherical joints give the drone the roll, pitch, and yaw characteristics, while the Cartesian joints give the drone the translational characteristics. Also, a revolute joint was used to give the propellers of the drone the rotational motion during simulation.

The revolute joint is attached in this design by invoking the revolute joint block and connecting it as shown in Figure 6.

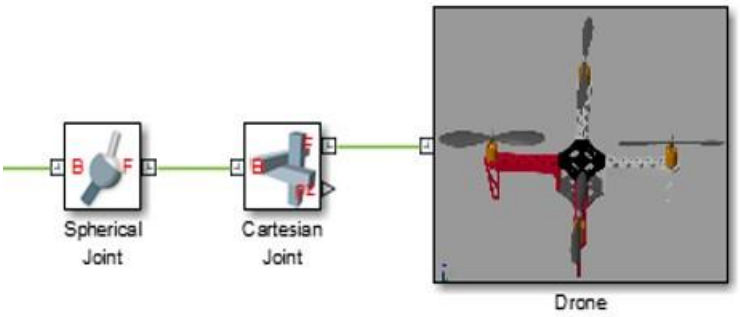

Figure 6: Attaching Cartesian and spherical joints

Designing the height stabilization of the PID controller

The PID controller was designed using the PID block in Simulink software as shown in Figure 7. First, a constant block was added to input the desired height of the drone. Next, a summation block was incorporated into the system. This summation block receives the current height of the drone from the Cartesian joint and calculates the difference between the current height and the desired height. This difference is then fed to the PID controller, which adjusts the thrust of the drone's motors accordingly, either increasing or decreasing the thrust—to maintain the drone at the specified height.

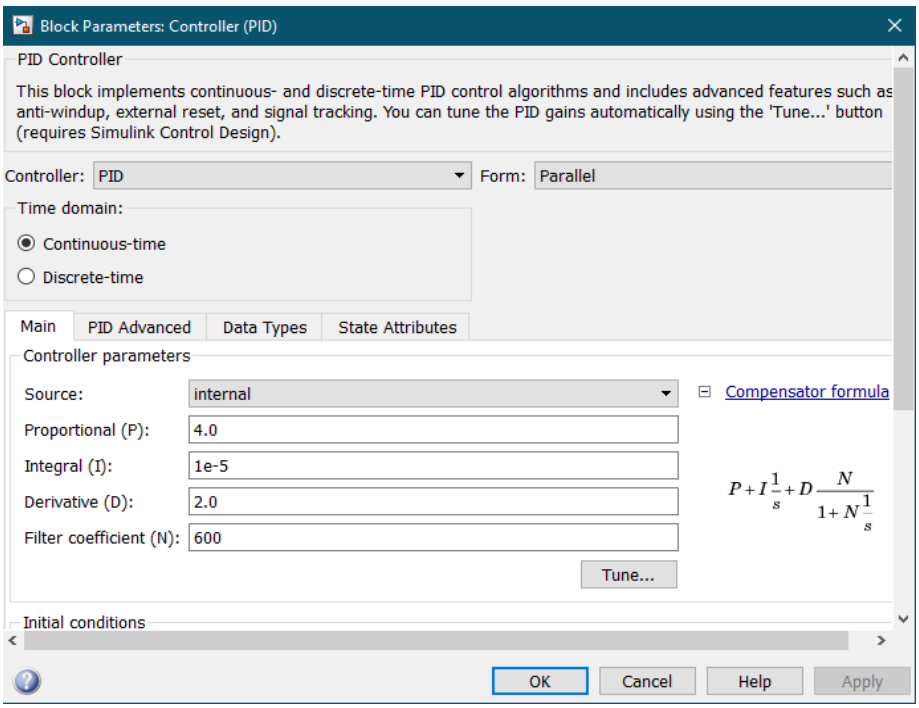

Figure 7: PID controller designed

The equation implemented for the PID algorithm was as thus:

$$PID = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}} \quad (1)$$

Where:

P is the proportional gain, I is the integral gain; D is the derivative gain and; N is the filter coefficient that filters out noise; S in the equation signifies frequency domain.

From Figure 7, it is seen that: P = 4.0; I = 1 x 10^{-5}; D = 2.0 and N = 600

Simulation of the Drone

The mechanism configuration block sets mechanical and simulation parameters that apply to an entire system were connected. The gravity parameters were set. Without this block, the effect of gravity will not be felt on the drone.

The solver configuration block defines solver settings used for simulation. These solver types include the Backward Euler solver and the Trapezoidal rule. The simulation requires at least one of these settings to run.

The world frame block was used to attach the drone to the world (Mechanical ground) through the spherical and Cartesian joints. This was appropriate since the six DoF of motion of the drone would be in the world.

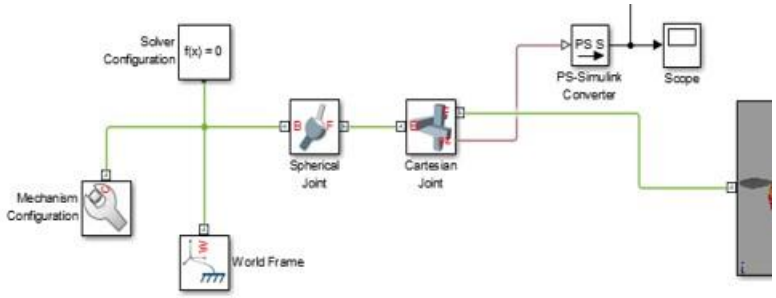

Figure 8: Adding the solver configuration, world frame and mechanism configuration blocks.

Creating disturbance

To demonstrate the stability effect of a PID controller, disturbance was introduced during the simulation process. This disturbance involved applying a vertical force (pushing) to the drone in either the positive or negative z-axis direction.

To design the disturbance, two main considerations were considered: the timing of the disturbance application (step time) and the desired duration of the force. This approach results in a square wave graph. To accomplish this in Simulink, the step block can be utilized. By combining two-step blocks, a square wave can be created to accurately model the disturbance.

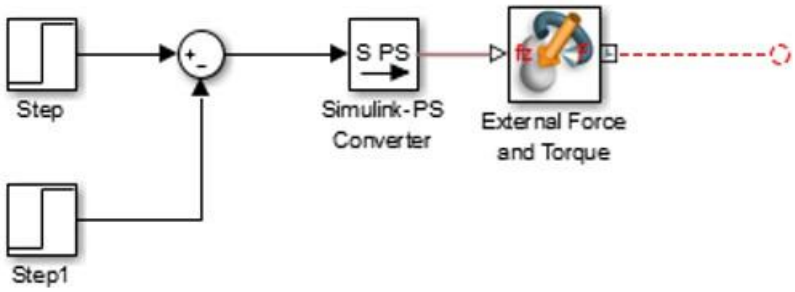

Figure 9: Disturbance model

RESULTS AND DISCUSSIONS

Table 1: Results from the simulation without disturbance

S/N	Rise time (s)	Overshoot (m)	Settling time (s)	Steady-state error (m)
1	1.2	1.08	1.6	0

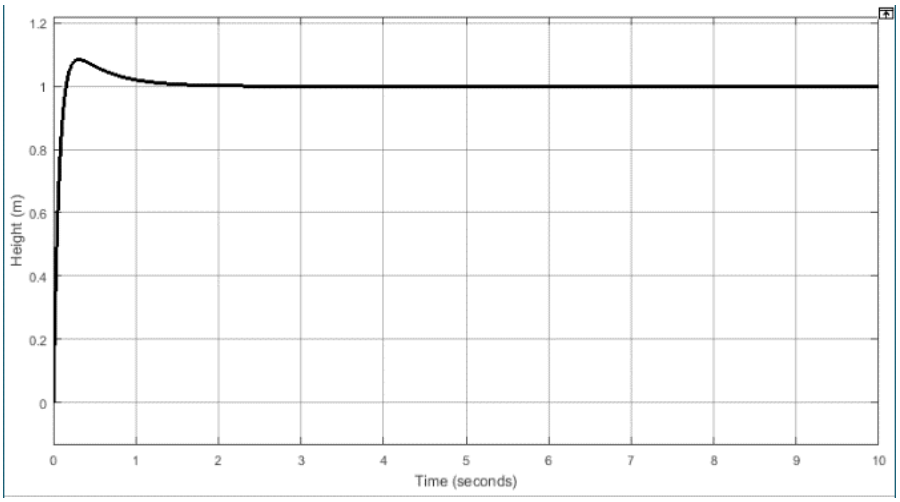

Figure 10: Simulation result without disturbance

The simulation was conducted over a period of ten (10) seconds to achieve the desired result. The target height for the drone was set at 1 meter. During this simulation, no disturbances were caused. The drone stabilized at the desired height after 1.6 seconds of flight. At that point, an overshoot of 1.08 meters was recorded, indicating that the drone initially exceeded the target height by 0.08 meters before correcting itself. The time taken to correct this error is determined by the difference between the rise time and the settling time. Once the 1.6 seconds passed, the drone reached the target height of 1 meter, resulting in a steady-state error of zero

Table 2: Result from simulation with disturbance

Number of Observation	Desired Height (m)	Force (N)	Step time (s)	Duration of contact of force	Z-axis direction
1	0.5	5	4	0.5	Positive
2	0.5	50	4	0.5	Positive
3	1	5	4	0.5	Negative
4	2.5	50	4	0.5	Negative

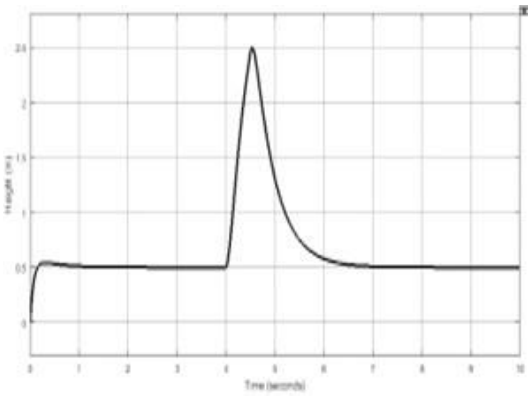

Figure 11: Simulation from observation 1

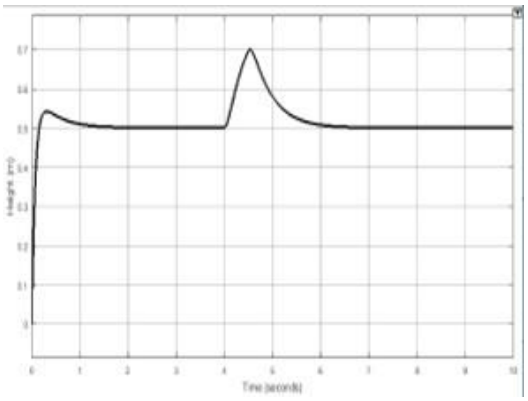

Figure 12: Simulation Graph from observation 2

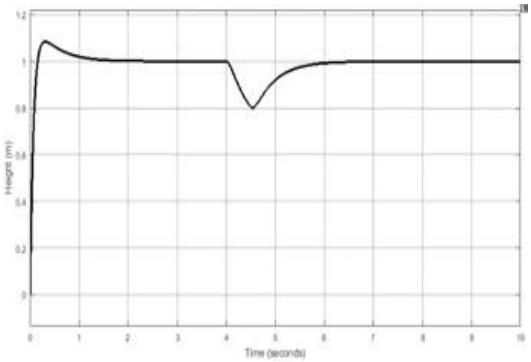

Figure 13: Simulation from observation 3.

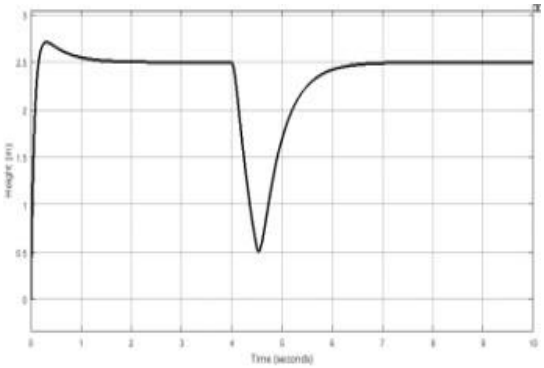

Figure 14: Simulation Graph from observation 4

From Observation 1, in Table 2 in relation to Figure 11, the target height programmed for the drone was set at 0.5 meters. The simulation runs for 10 seconds, during which a disturbance of 5 Newtons was introduced in the positive z-axis after four seconds. The graph in Figure 11 showed that height stabilization occurred 2.5 seconds after the disturbance was applied. An overshoot of 0.2 meters was observed because of the applied force. The time taken to correct the gained height was measured from the moment the force was applied to the subsequent settling time. After 2.5 seconds of the force being applied, the drone successfully reaches the desired height of 0.5 meters, resulting in a steady-state error of zero.

From observation 2, in Table 2 in relation to Figure 12, when a disturbance of 50 N is applied in the positive z-direction, after 4 seconds of running the simulation, the intended height was 0.5 m, and the duration of the applied force was 0.5 seconds.

Height stabilization occurred 3.0 seconds after the disturbance was applied. An overshoot of 2.0 m was observed due to the impact of the applied force. The time taken to correct the gained height was the difference between the moment the force was applied and the settling time after the force was removed.

After 3.0 seconds of applying the force, the drone reaches the desired height of 0.5 m, resulting in a steady-state error of zero.

From Observation 3, in Table 2 in relation to Figure 13, the programmed height for the drone was set at 1.0m. The simulation ran for a duration of 10 seconds, during which a disturbance of 5N was applied in the negative z-direction after the first four seconds. Height stabilization occurs 2.4 seconds after the disturbance is introduced, resulting in an undershoot of 0.2m. This undershoot was a consequence of the applied negative force.

The time taken to regain the lost height was the difference between the moment the force was applied and the settling time (the time taken to stabilize after applying the force). After 2.4 seconds of the force being applied, the drone reaches a height of 1m, which leads to a steady-state error of zero.

Observation 4, in Table 2 in relation to the desired (programmed) height for the drone was set at 2.5 meters. The simulation ran for a total of 10 seconds, during which a disturbance of 50 Newtons in the negative z direction was applied after four seconds. Height stabilization occurred 2.7 seconds after the disturbance was applied, but an undershoot of 2.0 meters was observed due to the negative force. The time taken to regain the lost height was the difference between the moment the force was applied and the settling time following this application. Ultimately, after 2.7 seconds from the disturbance, the drone reached its desired height of 2.5 meters, resulting in a steady state error of zero.

CONCLUSION

In this research work, a computer model of a drone was developed. STEP files of the various parts of the drone were imported. After importing these STEP files, the rigid transform block of the simscape multibody software was used to assemble the drone as a single body. Revolute joints were added at the appropriate position in the design to simulate the rotation of the blades and thrust. Computer model, as against mathematical model, were preferred in this research work to eliminate approximations and assumptions inherent in the mathematical model. These approximations arise because of linearization usually carried out in mathematical models. Again, a computer model was preferred as issues of non-convergence of solver software during compilation were eliminated.

PID algorithm was also developed using Simulink PID block. Gains of the PID parameters are tuned using the Ziegler-Nichols method of PID tuning.

After running the simulation, it is seen that the drone was able to maintain its height with an average settling time of 2.5 seconds. Application of different magnitudes of disturbances was initiated, and the drone was seen to recover again at an average of 2.5 seconds.

Finally, it was safe to conclude that the PID designed in the research work was robust enough to handle disturbances of up to 50 N and was still able to make the drone recover from such disturbance at a reasonable time of 2.5 seconds.

ACKNOWLEDGEMENT

Authors acknowledge Department of Pure and Applied Physics Laboratory for the convenient environment and constant power supply.

Conflict of interest: Authors have declared that no conflict of interest exists.

REFERENCES

Atinga, A., Kósi, K., & Tar, J. K. (2025). *Measuring model parameter setting errors' effects in the control of an underactuated system*. *Electronics*, **14**(5), 883. https://doi.org/10.3390/electronics14050883

Ihnak, M. S. A., & Edardar, M. M. (2023). *Comparing LQR and PID controllers for quadcopter control effectiveness and cost analysis*. In 2023 IEEE 11th International Conference on Systems and Control (ICSC) (pp. 1–6). IEEE. https://doi.org/10.1109/ICSC58660.2023.10449763

Karakoc, T. H., & Özbek, E. (Eds.). (2023). *Unmanned Aerial Vehicle Design and Technology*. Springer Cham. https://doi.org/10.1007/978-3-031-45321-2

Al-Samarraie, S. A., & Gorial, I. I. (2024**).** *Assessment of FLC, PID, Nonlinear PID, and SMC Controllers for Level Stabilization in Mechatronic Systems. Journal of Robotics and Control*, 5(6)**,** 1845–1861. — Compares PID and advanced controllers for mechatronic stabilization tasks.

Satbayev, M. Zharkyn (2024). *Modeling and Control of a Quadrotor Unmanned Aerial Vehicle. Computing & Engineering*, **2**(4), 22–27. https://doi.org/10.51301/ce.2024.i4.04

Boubakir, A., Souanef, T., Labiod, S., & Whidborne, J. F. (2024). *A robust adaptive PID-like controller for quadrotor unmanned aerial vehicle systems*. *Aerospace*, **11**(12), 980. https://doi.org/10.3390/aerospace11120980

Zhao, J., Cao, Y., & Zhou, Z. (2025). *Design of a fuzzy adaptive PID control system for quadrotor UAVs. Journal of Electronic Research and Application, 9*(3), Article 10809.

Abbas, N., Abbas, Z., Liu, X., Khan, S. S., Foster, E. D., & Larkin, S. (2023). *A survey: Future smart cities based on advanced control of unmanned aerial vehicles (UAVs)*. *Applied Sciences, 13*(17), 9881. https://doi.org/10.3390/app13179881