# Drone Stability Simulation Using ROS and Gazebo

**Chapter** · July 2021

DOI: 10.1007/978-981-16-2164-2_11



**4 authors**, including:

Rajesh Kannan Megalingam
Amrita Vishwa Vidyapeetham - Humanitarian Technology (HuT) Labs
**275** PUBLICATIONS **2,294** CITATIONS
SEE PROFILE

Darla Vineeth Prithvi
Amrita Vishwa Vidyapeetham
**4** PUBLICATIONS **47** CITATIONS
SEE PROFILE

# Drone Stability Simulation Using ROS and Gazebo


**Rajesh Kannan Megalingam, Darla Vineeth Prithvi, Nimmala Chaitanya Sai Kumar, and Vijay Egumadiri**

**Abstract** In the recent year, Unmanned Aerial Vehicles (UAV) which are generally called Drones or Quadcopters have gained a lot of attention towards researches and companies because of their wide range of capabilities in different sectors such as in military and public. The PID control algorithm which runs in the ROS environment controls the 3-D modelled quadcopter in Gazebo. The main focus is to make the quadcopter stable and to hover at the desired point in Gazebo by tuning PID values in the ROS environment. A plot juggler has been used to obtain the trajectory analysis and to analyse the tuning errors by the trial and error method for each quadcopter motion such as roll, pitch and altitude. Through this framework, certain parameters of the drone are obtained such as settling and rise time with PID tuning.

**Keywords** PID · ROS · Gazebo · Pitch · Roll · Yaw · Altitude · Rise time · Settling time

## 1 Introduction

In olden days where transmitting information is quite often a problematic task and also takes a large amount of time to communicate. Today's technology has made communicating things simpler as it takes place in a fraction of seconds. In this jam-packed busy schedule, knowing what's happening around us and how things are going to change is extremely important. In the field of communicating information like disaster and safety management, transport, monitoring traffic updates, aerial photography, agriculture, traffic monitoring, logistics, meteorological purposes and

R. K. Megalingam (✉) · D. V. Prithvi · N. C. S. Kumar
Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India
e-mail: rajeshm@am.amrita.edu

V. Egumadiri
Department of Electrical Engineering, University of South Florida, Florida, USA
e-mail: egumadiri@usf.edu



M. Bianchini et al. (eds.), *Advanced Computing and Intelligent Technologies*,
Lecture Notes in Networks and Systems 218,
https://doi.org/10.1007/978-981-16-2164-2_11

in the military sector, an unmanned aerial vehicle (UAV) simply known as Drone (or) Quadrotor has a wide range of applications.

Mainly in the field of survey and rescue, drones have a greater advantage due to their wireless technology. Predominantly, drone stability performs a significant role in real-time video assistance, but in general, due to its unstable nature, execution and testing in the real world, it can lead to loss or damage to equipment and the environment. In this paper, our main goal is to analyse the stability and various errors to stabilize the drone at a destination coordinate autonomously using a Proportional–Integral–Derivative (PID) control system in Gazebo open source, where the position is detected using WhyCon markers by the overhead camera. PID algorithm controls the position feedback error and the drone. Gazebo is a 3D simulator consisting of a physical engine for illumination, inertia, gravity and all types of common sensors, and Robotic Operating System (ROS) serves as the interface for the drone.

## 2 Problem Statement

Drones have been used in the research field and also in commercial applications. An experienced pilot is needed to control the drone. It is a difficult task to hover a drone steadily at a fixed point by manual operation. This paper presents the simulation and implementation of a UAV in a Gazebo environment using a robotic operating system. The drone simulation in the paper helps to understand the tuning of PID flight controllers with less stability error and the characteristics of drones using plot juggler in ROS.

## 3 Related Works

In the research work [1], the authors did the stability simulation analysis for a 3-D modelled drone. CATIA software has been used to obtain the parameters such as mass and the moment of Inertia in x, y and z axes. A detailed study about the stability simulation for basic drone motions such as roll, pitch, yaw and altitude has been mentioned with parameters such as settling time and overshoot percentages. Settling time of roll and pitch is 1.419 s, yaw is 2.327 s and altitude hold is 6.339 s.

In the research paper [2], the gradient method has been proposed to tune the proportional–integral–derivative (PID) controller parameters for a quadcopter model named as AR Drone. This paper mentioned experimental results as well as simulations of the autotuning of PID for AR drones in two ways: one is the waypoint navigation and the other is the leader–follower formation control.

The authors in [3] use a PID-based drone available in MATLAB and Simulink for the simulation. The study deals with the basics of drone dynamics and external forces that act on the drone and also develop autonomous navigation and trajectory planning for drones. Newton–Euler equations have been used for the dynamics. The papers

also showed detailed simulations and results to showcase the difference between the PD and PID controllers.

In the research paper [4], the authors present the simulation, modelling and implementation of a quadrotor unmanned aerial vehicle. The study focuses on the quadcopter dynamics and to further develop the autonomous navigation system, autonomous trajectory planning for quadcopter, and efficient control algorithms. The Newton–Euler method has been used for quadcopter dynamics and the quadcopter was developed in MATLAB and Simulink. The author also compared the MATLAB model with the commercially available quadcopters such as the adopter and MultiWii.

The authors in [5] presented an unmanned aerial vehicle (UAV) which can detect humans. The video capture by the UAV will be collected and processed in MATLAB using a human detection algorithm. A Pixhawk flight controller has been used as a quadcopter controller.

In [6], the study focuses on making a quadcopter from scratch. Arduino Mega, Raspberry Pi, GPS module, Flysky CT6B transmitter and FS-R6B receiver are used in the making of the drone and OpenCV with optical flow and Canny Edge Detection algorithms were also used for the location and the size of the objects. The author successfully implemented and tested a quadcopter which is able to fly from one point to another point autonomously and able to detect and award obstacles in its path.

The authors in [7] present a simulation framework for the position control and trajectory tracking of the Gazebo model of the Crazyflie 2.0 quadcopter. The control algorithm which runs in the MATLAB and Simulink® environments controls the position of the quadcopter in the Gazebo simulator, through Robot Operating System (ROS) interfaces. The goal here is to establish the connection between the ROS-enabled Gazebo quadcopter model and the Simulink environment and thereby sending and receiving the control commands between both the environments. The main focus of this paper is to establish a connection between ROS and Gazebo to control the drone.

The research paper [8] presents an area for testing the quadcopter capabilities and also presents a proper environment setup to train the users. The developed arena consists of different setups with certain objects such as rings and poles and with different paths and conditions for the user to gain knowledge on stability and on the performance of the quadcopter. In total, there are five arena setups mentioned. In the results, the average time taken by the drone to complete each arena is also mentioned.

In the research work [9], the authors implemented, simulated and proposed a mathematical model to tune the wheels encoders, PID controllers and inertial measurement unit (IMU) for multi-terrain robots. The authors used a robotic operating system to interface with Gazebo to simulate the multi-terrain robot. The authors also simulated the robot in different conditions with and without PID values to showcase the system stability.

In [10], the authors developed a 3-D model bot in SOLIDWORKS and simulated it in Gazebo of the Robot Operating System (ROS). A gamer's steering wheel is used to control the movement and speed of the robot. The robot was tested in different environments such as rough terrain, asphalt planes and ramps.

# 4 Drone Modelling

## *4.1 System Architecture*

Figure 1 represents the block diagram of the entire system of this research work. The master controller for the whole system is ROS which subscribes and publishes various nodes and also communicates with the drone and other Gazebo parameters. As the drone changes its momentum to reach the destination and to acquire stabilized state, the overhead camera module detects the WhyCon markers placed above the drone as shown in Fig. 2. According to the sensor, camera and physical parameters of the drone, the correction value is calculated based on PID terms and feedback

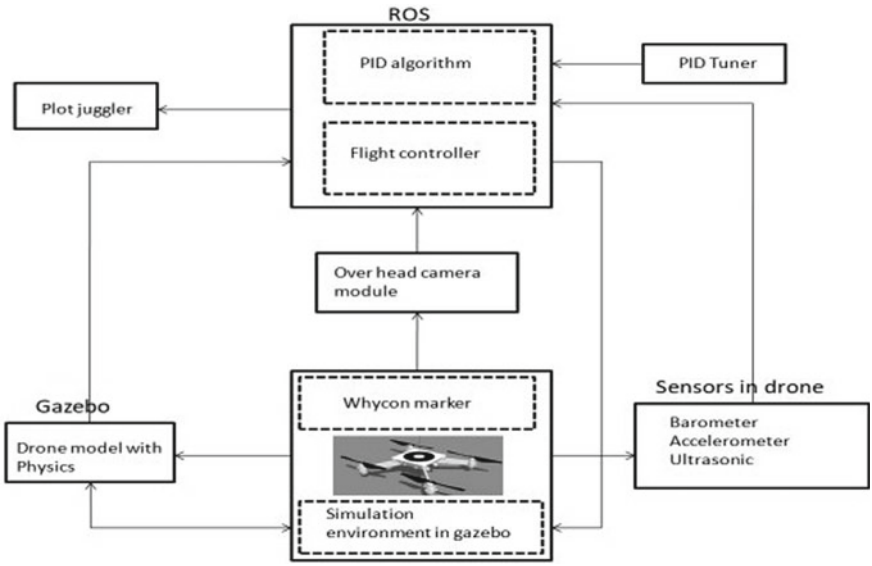


**Fig. 1** System overview—block diagram

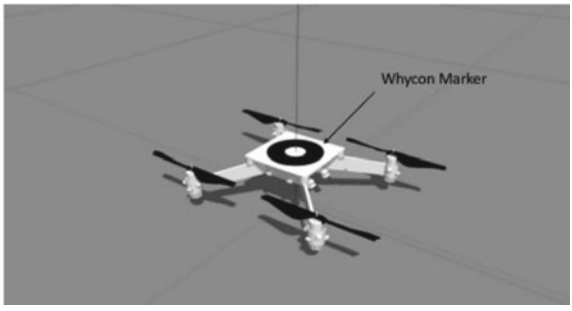


**Fig. 2** Drone in the Gazebo simulation environment

to the system to reduce the error value. PID algorithm and flight controller script communicate with the drone. Using plot juggler, graphs with respective altitude, pitch, roll errors and time are taken. By tuning appropriate values of Kp, Ki and Kd of Roll, Pitch, Yaw and Altitude parameters, the drone can reach the destination point with less settling time and less stability error.

## *4.2 Drone Motion*

Quadrotors have a major asset over other types of unmanned aerial vehicles; they can float without difficulty in confined space within a short period of time, can easily take off and land. They maintain the concept of the thrust and steering function by four propellers attached with independent motors at the respective edges of diagonals from the centre of the drone as shown in Fig. 2.

There are four degrees of freedom in which the drone can acquire stable movement. Rotating the four propellers in the below-mentioned directions results in the following movements:

(1) **Thrust/Throttle**: By increasing or decreasing the angular velocity of four motors, drone height can be increased or reduced. To attain vertical lift of the drone, each motor spins in the opposite direction of the adjacent motor, as shown in Fig. 3.

(2) **Pitch**: Changing the angular velocity of front and back motors as shown in Fig. 4 results in the drone movement to move forward or backward. Simply increasing the speed of backside motors as compared to the front side effects the drone to move forward and vice versa.

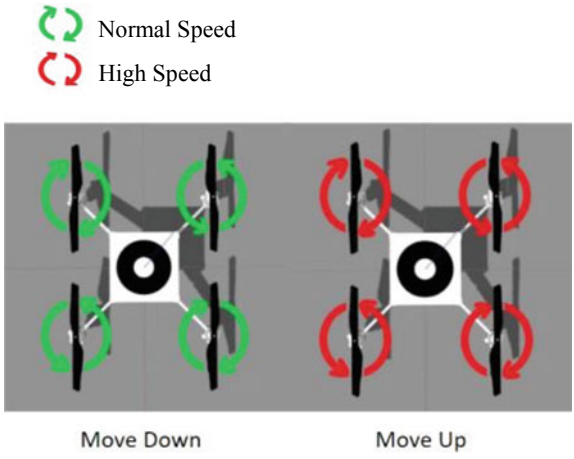


**Fig. 3** Throttle movement

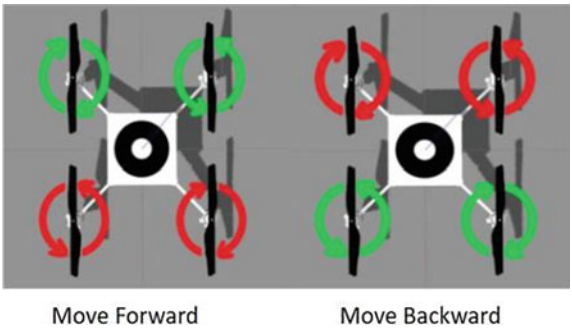


**Fig. 4** Pitch movement

(3) **Roll**: To move the drone left or right, we have to change the angular velocity of the right side or left side motor. In this rolling movement, the pair of left-sided motors rotates with high speed compared to that of the right-sided motors which effects the drone to bend to the right side as shown in Fig. 5.
(4) **Yaw**: The drone yaws to the right or the left by changing the speed of the alternating motors as shown in Fig. 6.

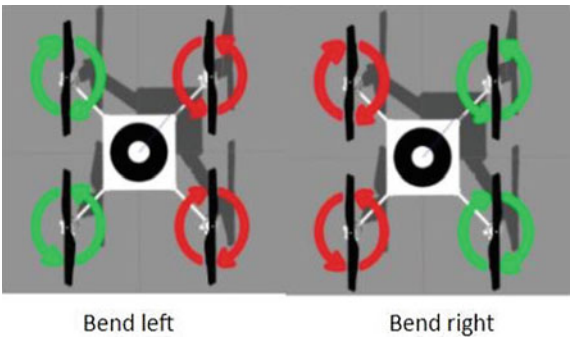


**Fig. 5** Roll movement

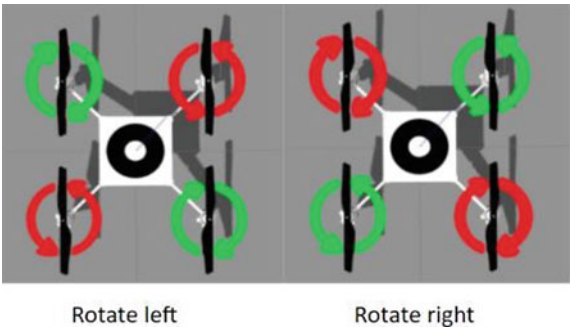


**Fig. 6** Yaw movement

### *4.3 Sensors*

There are mainly three sensors used in this drone, in order to maintain its 'pose' inflight and stability. A drone highly depends on sensors that continuously monitor the drone's 'attitude'. Based on the feedback from sensor values, the motor spin speed adjusts and shifts the drone in flight to remain stable.

(1) **Accelerometer and Gyroscope**: The accelerometer is employed to sense the acceleration of the drone within the X, Y and Z-axis. The gyroscope is used to measure the tilt in pitch, roll and yaw of the drone. MPU6050 sensor consists of a 3-axis accelerometer and a 3-axis gyroscope. It helps us to measure acceleration, velocity, orientation, displacement and other motions related to drones. Mostly, this sensor is used in self-balancing robots and hand-gestured robots.
(2) **Barometer**: Barometers are used to measure air pressure and detect the height of the drone from the ground. The barometer can easily detect large enough heights.
(3) **Ultrasonic sensor**: Ultrasonic sensors emit ultrasonic sound waves and convert reflected waves into electric signals, which are extremely useful when flying less than 50 cm from the ground, whereas the barometer gives an accurate distance beyond 50 cm.

## 5 PID Controller

In this paper, the drone model in Gazebo uses a PID flight controller. PID is a combination of three control systems such as proportional, integral and derivative. These control systems are combined in such a way that they produce a control signal. As shown in Fig. 7, the PID controller uses a control loop feedback system to continuously calculate the difference between the desired set point and measured

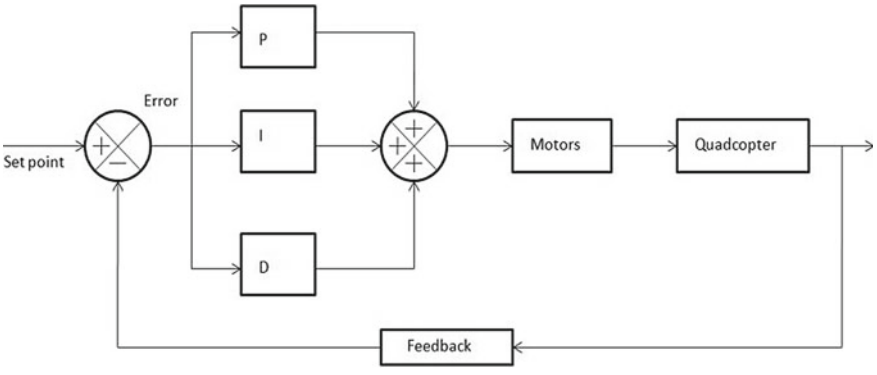


**Fig. 7** The overall system block diagram with PID

value or current value to generate an error signal. Based on the generated error signal, the controller will alter the system output in such a way that it will minimize the error. Otherwise, it will reduce the error to zero. The controller in the drone gets the error signal by calculating the difference between the gyroscopic sensor data and the rotation speed of motors. The controller will minimize the error signal by changing the speed of the motors to keep constant output based on the input of the sensor data.

From Fig. 7, it is clear that the error is being continuously altered by three controllers P, I and D. The summation result of the three controllers are being fed to motors and thus enable the drone to fly. Due to the feedback loop in the controller, this process continues until the drone attains stability.

## *5.1 P Controller*

Figure 8 shows the P controller block diagram. P controller gives an output proportional to the measured error at that time. Proportional gain (Kp) determines how intense the flight controller works to correct the error and to reach the set point. In the P controller, a fine-tuned proportional gain (Kp) is responsible for a snappy and sharp control of the drone. The more is the Kp, the lesser is the steady-state error but if the Kp is too high, it causes overshoot in the drone and thus causes instability. The downside of Kp is even when the drone reaches a stable state if the error is too small, the output will be negligible and the drone will never reach the set point (Fig. 8)

$$\text{Output} = \text{Kp} \times \text{error} \tag{1}$$

## *5.2 PD Controller*

The PD controller block diagram is shown in Fig. 9. The PD controller can predict the future error by analysing the rate at which the drone is approaching a set point

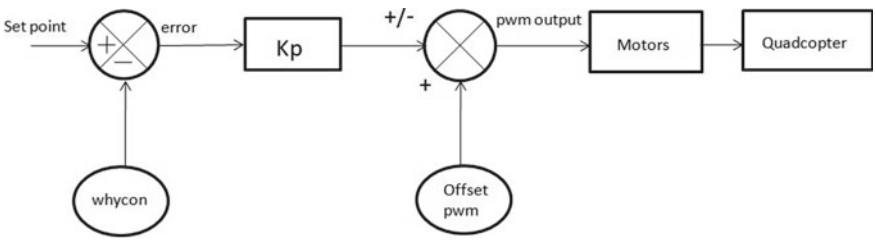


**Fig. 8** Depicts the P controller block diagram interfacing with the quadcopter

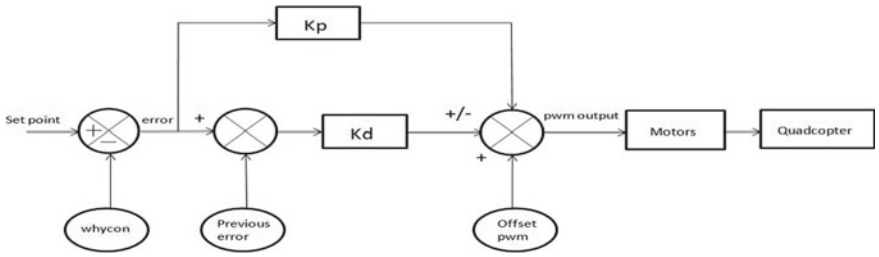


**Fig. 9** Depicts the PD controller interfacing with the quadcopter

and changes controller output with the rate of change of error with respect to time. The PD controller helps to decrease the output of P and I when the drone is near to the point to decrease the aggressive changes in the drone. Here, differential gain will work as a shock absorber to drones, and it reduces overshoots and oscillation caused by P and I. If the differential gain is too high, I will also cause vibrations in the drone due to the non-linearity in the system and it also causes overheat in drone motors. Overall, a fine-tuned gain gives a soft control on the drone (Fig. 9)

$$\text{Output} = \text{Kd} \times (\text{error} - \text{previouserror}) \tag{2}$$

## *5.3 PID Controller*

Figure 9 shows the entire PID controller flow diagram. The PID controller will eliminate the offset and steady-state error by integrating the error for over a period until the error gets nullified and the I controller also determines the stiffness and sturdiness of a drone to hold the set point against external forces such as wind. If integral gain (Ki) increases, the steady-state error will also decrease, but when Ki is too high overshoot will increase and the stability of the system will also degrade. Just like the P controller flight path, I controller flight path is not efficient and takes a lot of time to settle (Fig. 10).

$$\text{Output} = (\text{sum of errors}) \times \text{Ki} \tag{3}$$

$$\text{PID output} = \text{Kp} \times \text{error} + (\text{sum of errors}) \times \text{Ki} + \text{Kd}(\text{error} - \text{previouserror}) \tag{4}$$

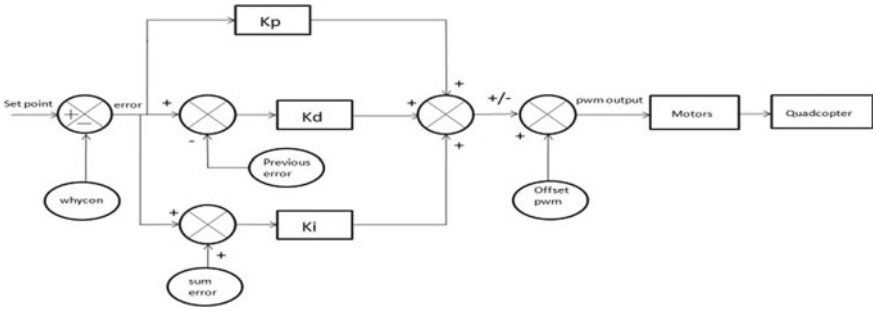


**Fig. 10** Depicts the PID controller interfacing with the quadcopter

## 6 Results and Discussion

Initially, the drone is situated at the origin [0, 0, 0] as shown in Fig. 2. When we run the PID script, the overhead camera detects the WhyCon marker attached at the top of the drone. To reach the destination coordinate, the drone gradually changes its four motors' speed as explained in the drone motion section. The PID script starts calculating errors from the current position coordinates to the previous coordinate position. Based on the position errors, the altitude, pitch and roll values change. By the trial and error method of tuning, the simulation parameters such as Kp, Kd and Ki values of Roll, Pitch, Throttle and settling time are obtained as shown in Table 1.

The graph below in Fig. 11 shows where the drone starts at 19 s and reaches the destination at nearly 24 s and hovers steadily with minimal error. For testing, the destination point in this experiment is [2, 2, 20]. In Figs. 12 and 13, that is altitude errors and pitch, error stabilizes in less than 4 s. In Fig. 14, it takes less than 6 s. it is clearly observed that both the altitude and pitch graph look similar because both the coordinate points of the X and Y-axis are the same. Here, the Yaw error is neglected because there is no clockwise or anti-clockwise rotation required to reach the destination coordinate.

**Table 1** Simulation parameters

| S. no | Parameters | Kp | Ki | Kd | Settling time (Ts) (s) |
|---|---|---|---|---|---|
| 1 | Throttle | 1083 | 7 | 1410 | 4 |
| 2 | Pitch | 262 | 13 | 2555 | 4 |
| 3 | Roll | 157 | 5 | 393 | 6 |

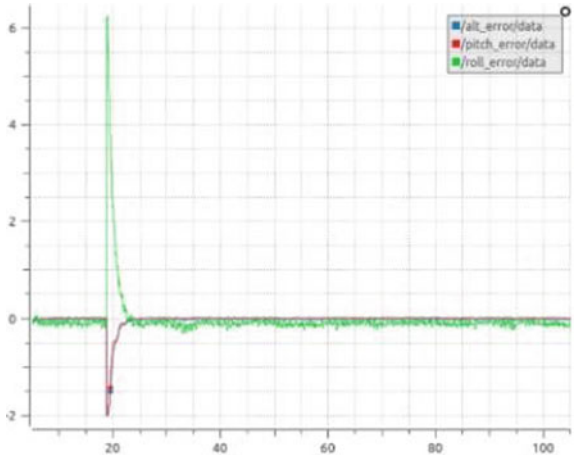


**Fig. 11** Overall error versus time

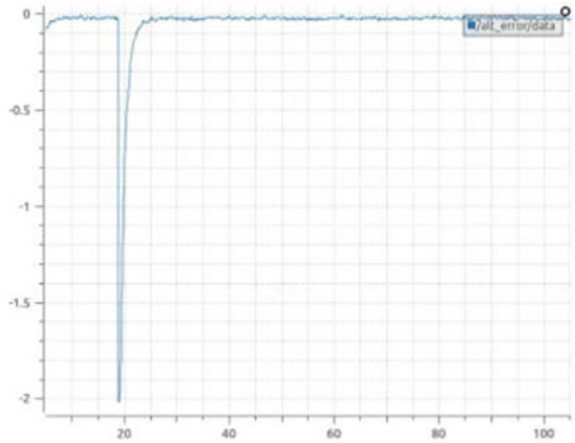


**Fig. 12** Altitude error versus time

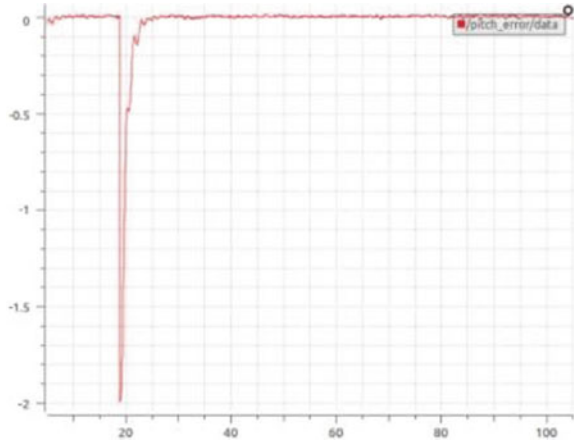


**Fig. 13** Pitch error versus time

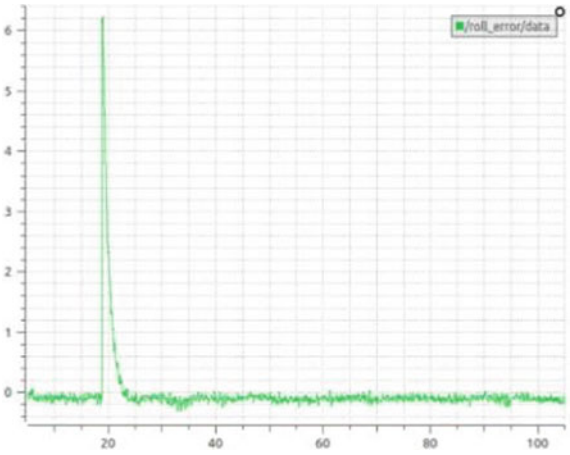


**Fig. 14** Roll error versus time

## 7 Conclusion

In this paper, we presented the simulation of drone interfacing with ROS and the Gazebo simulator, in the tests of the drone to reach the destination point and hover steadily. Due to the physical and structural characteristics of the drone it is not easy to hover the drone with comfortable stability. But required stability can be achieved by tuning the kp, ki and kd values precisely. In a fraction of seconds, the drone reached its destination point. However, there is much scope for this system improvement in the future to increase the wide range of communications by adding different types of communicating devices like cameras for telecasting. This model can be further developed by integrating with GPS for survey and rescue operations.

**Acknowledgements** We take this opportunity to express our profound gratitude and deep regards to HuT Labs, Amrita Vishwa Vidyapeetham which provided guidance and space for us to complete this work.

## References

1. Lukmana, M.A., Nurhadi, H.: Preliminary study on Unmanned Aerial Vehicle (UAV) Quadcopter using PID controller. In: 2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA), Surabaya, pp. 34–37 (2015). https://doi.org/10.1109/ICAMIMIA.2015.7507997
2. Babu, V.M., Das, K., Kumar, S.: Designing of self-tuning PID controller for AR drone quadrotor. In: 2017 18th International Conference on Advanced Robotics (ICAR), Hong Kong, pp. 167–172 (2017). https://doi.org/10.1109/ICAR.2017.8023513
3. Vamsi, D.S., Tanoj, T.V.S., Krishna, U.M., Nithya, M.: Performance Analysis of PID controller for Path Planning of a Quadcopter. In: 2019 2nd International Conference

on Power and Embedded Drive Control (ICPEDC), Chennai, India, pp. 116–121, doi: 1109/ICPEDC47771.2019.9036558
4. Fernando, H.C.T.E., De Silva, A.T.A., De Zoysa, M.D.C., Dilshan, K.A.D.C., Munasinghe, S.R.: Modelling, simulation and implementation of a quadrotor UAV. In: 2013 IEEE 8th International Conference on Industrial and Information Systems, Peradeniya, pp. 207–212 (2013). https://doi.org/10.1109/ICIInfS.2013.6731982.
5. Mandal, S., Biswas, S., Balas, V.E., Shaw, R.N., Ghosh, A.: Motion prediction for autonomous vehicles from lyft dataset using deep learning. In: 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, pp. 768–773 (2020). https://doi.org/10.1109/ICCCA49541.2020.9250790
6. Sharma, V.N.V.A., Rajesh, M.: Building a quadcopter: an approach for an autonomous quadcopter. In: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, 2018, pp. 1252–1258 (2018). https://doi.org/10.1109/ICACCI.2018.8554718
7. Nithya, M., Rashmi, M.R.: Gazebo—ROS—Simulink framework for hover control and trajectory tracking of crazyflie 2.0. In: TENCON 2019–2019 IEEE Region 10 Conference (TENCON), Kochi, India, pp. 649–653 (2019). doi: 1109/TENCON.2019.8929730
8. Megalingam, R.K., Raj, R.V.R., Masetti, A., Akhil, T., Chowdary, G.N., Naick, V.S.: Design and implementation of an arena for testing and evaluating quadcopter. In: 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, pp. 1–7 (2018). https://doi.org/10.1109/CCAA.2018.8777578
9. Megalingam, R.K., Nagalla, D., Nigam, K., Gontu, V., Allada, P.K.: PID based locomotion of multi-terrain robot using ROS platform. In: 2020 Fourth International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, pp. 751–755 (2020). https://doi.org/10.1109/ICISC47916.2020.9171152
10. Megalingam, R.K., Nagalla, D., Pasumarthi, R.K., Gontu, V., Allada, P.K.: ROS based, simulation and control of a wheeled robot using gamer's steering wheel. In: 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, pp. 1–5 (2018). https://doi.org/10.1109/CCAA.2018.8777569.