Eduvest – Journal of Universal Studies
Volume 2 Number 9, September, 2022
p- ISSN 2775-3735 - e-ISSN 2775-3727

DESIGN AND TESTING OF A PID CONTROLLER ON A PARROT MINIDRONE

Hrishitva Patel
SUNY Binghamton, New York, United States
Email: hpatel51@binghamton.edu

ABSTRACT

This paper focuses on quadcopter control design and uses the Simulink Support Package for Parrot Minidrones to test various control approaches. This project's goal is to bridge the knowledge gap between control design and implementation, making it simpler to comprehend the fundamental ideas of control theory. Obtaining a dynamic model of the quadcopter, linearizing the system around an equilibrium point, designing PID controller on the linearized system, and then verifying the controllers on the non-linear model using simulations and test flights with the actual drone are the key components of this research. The tuned controllers are validated by trajectory tracking, setpoint regulation, and disturbance rejection.

KEYWORDS Quadcopter, Parrot Drone, Support Package, PID, Simulink

INTRODUCTION

A quadcopter features four revolving propellers. To provide lift, these rotating propellers are employed. Essentially four motors and six degrees of freedom make up quadcopters (Pérez et al., 2014). Three of these six degrees of freedom are for rotational motion and three are for translational motion. Quadcopters are a type of rotorcraft that can fly independently with very little assistance from humans thanks to a highly developed control system that makes them incredibly stable. (MATLAB) (Prayitno et al., 2018).

There are two different setups for quadcopters: plus configuration and cross configuration. This division is based on direction and flight. Both arrangements employ oblique motors spinning in the same direction. Plus arrangement employs two motors to produce roll and pitch motions, whereas cross configuration uses all four rotors (Schaub & Junkins, 2003). This is the difference between the two (Gopalakrishnan, 2016).

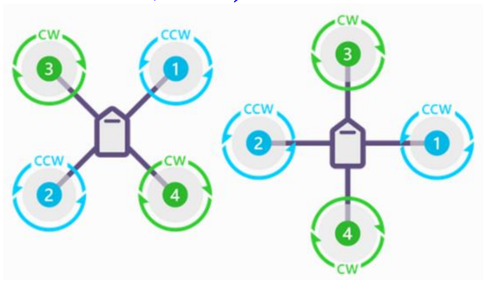

Figure 1. Configuration of motors

Quadcopters have four motors and six degrees of freedom (3 translational and 3 rotational). Quadcopters do not have an actuator for every motion. So, with 6-DOF and only four independent actuators, quadcopters are underactuated. We require it to have a six degree of freedom by using four actuators. To achieve 6-DOF, rotational and translational motions are coupled. (Fiaz, 2015) (H. Schaub a J. Junkins, 2003) (Tamayo et al., 2018)(Zhang et al., 2014), (Zouaoui et al., 2019).

RESEARCH METHOD

The following universal model presumptions are offered from various literature and research publications. Although different assumptions were used to simplify mathematical equations, these equations nonetheless produce plausible outcomes.

The four identical rotors of the quadcopter are used to control it.

- A quadcopter's structure is stiff and symmetrical, and its inertial matrix is diagonal.
- The origin is where the centre of gravity is located.
- The square of the rotor's speed determines the drag and thrust coefficients.
- The mass stays constant because the properties of the system are time invariant.
- Aerodynamic effects, including blade flutter, can be disregarded. (Musa, 2018) (Zhang, 2014)

In his investigation, Sabatino used the intricate quadrotor model. In order to assess the vectors in various frames, this model uses transformation matrices and the Newton-Euler equation for 3D motion. In this method, quadrotor equations of motion were developed using the link between force and moment balance. (Sabtino, 2015)

Rotation matrix is used to convert the velocities from body frame to inertial frame.

$$v = R.v_B$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = R.\begin{bmatrix}u\\ v\\ w\end{bmatrix}$$

Similarly, transformation matrix T is used to convert angular velocities from body frame to inertial frame

$$w = R.w_B$$

$$\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = T.\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

We can change a vector from a body frame to an inertial frame by applying the Transport theorem. The expression for forces and moments around an inertial frame is obtained. It is written as follows:
*“If the body-fixed frame is rotating with rotation vector **w**, then for any vector, **A**, $d\boldsymbol{A}/dt$ in the inertial frame is”:*

$$\left(\frac{dA}{dt}\right)_{inertial} = \left(\frac{dA}{dt}\right)_{body} + w_{B/I} \times A$$

So, by using this theorem the forces in inertial frame are given as;

$$\begin{bmatrix}F_x = m(\dot{u} + qw - rv)\\ F_y = m(\dot{v} - pw + ru)\\ F_z = m(\dot{w} + pv - qu)\end{bmatrix}$$

Similarly, by using the transport theorem for momentum about the inertial frame we get

$$\begin{bmatrix}M_x = \dot{p}I_x - qrI_y + qrI_z\\ M_y = \dot{q}I_y + prI_x - prI_z\\ M_z = \dot{r}I_z - pqI_x + pqI_y\end{bmatrix}$$

Using the rotation matrix given above, we can determine the forces and moments in a fixed frame for the body. The propeller forces produced by a quadrotor's standard configuration always point in the direction of the body frame's z-axis. Similar to how all thrust differential torques are centred on the x, y, and z axes of the body frame. The following is a simple way to express the quadrotor's weight in an inertial frame.

$$F_{inertial} = \begin{bmatrix}0\\ 0\\ mg\end{bmatrix}$$

The components of weight in body frame can be found by using rotation matrix:

$$F_{body} = R^T F_{inertial}$$

Simplifying, we have:

$$F_{body} = \begin{bmatrix}-mg[s(\theta)]\\ mg[c(\theta)s(\varphi)]\\ mg[c(\theta)c(\varphi)]\end{bmatrix}$$

The complete dynamic model of the quadrotor is given as:

$$\begin{bmatrix} -mg[s(\theta)] = m(\dot{u} + qw - rv) \\ mg[c(\theta)s(\varphi)] = m(\dot{v} - pw + ru) \\ mg[c(\theta)c(\varphi)] = m(\dot{w} + pv - qu) \\ \tau_x = \dot{p}I_x - qrI_y + qrI_z \\ \tau_y = \dot{q}I_y + prI_x - prI_z \\ \tau_z = \dot{r}I_z - pqI_x + pqI_y \end{bmatrix}$$

For input modelling, angular velocity approach is used. In this approach, the inputs are the angular velocities of the rotors. The input model is as follows

$$\begin{bmatrix} b({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2) \\ lb\left({\Omega_4}^2 - {\Omega_2}^2\right) \\ lb\left({\Omega_3}^2 - {\Omega_1}^2\right) \\ d(-{\Omega_1}^2 + {\Omega_2}^2 - {\Omega_3}^2 + {\Omega_4}^2) \end{bmatrix}$$

The matrices A and B of the linearized state-space model of the quadcopter is given as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} B$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1/m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1/I_x & 0 & 0 \\ 0 & 0 & I/I_y & 0 \\ 0 & 0 & 0 & I/I_z \end{bmatrix}$$

Pid Control On A Parrot Minidrone

MIT created the Simulink Support Package for Parrot Mini-drones. Programs for unique flight control systems are created using it. As long as the inputs and outputs of the FCS block remain the identical, this package allows us to replace the quadcopter's control part while maintaining all of the drone's standard operational features. Any control system we develop is incorporated into the FCS block and communicates with the other firmware components of the mini-drone. The Simulink model is shown in the figure down below.

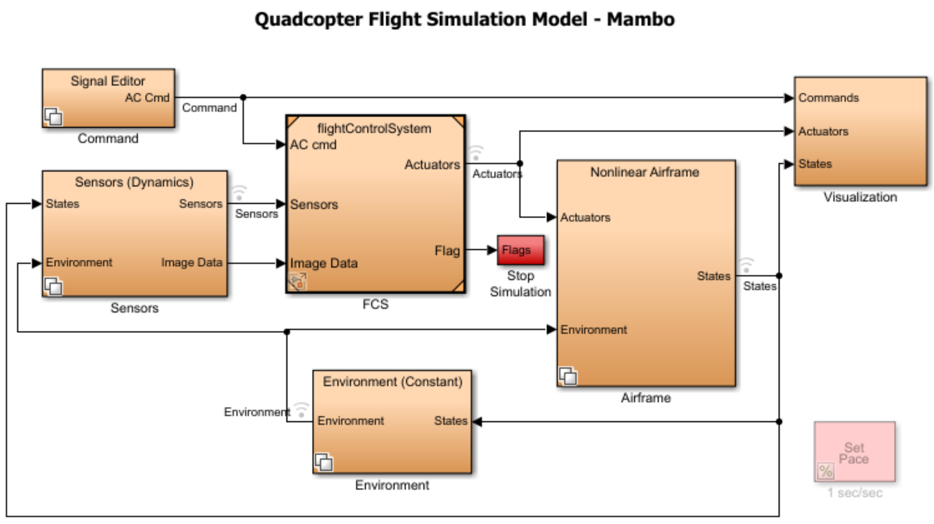

Figure 2. Flight Simulation Model in Simulink

The sensor block simulates the noise and dynamics of the four sensors that are on our mini drone. The drone has a pressure sensor and ultrasound sensor to measure altitude, an IMU to measure attitude relative to gravity and a camera that senses the horizontal motion by using image processing (Babu et al., 2017). The sensor readings are passed on to the flight control system. But we don't need pressure readings or IMU readings rather we need to know the altitude and attitude of the drone. So, there must be an additional logic that converts all of the measured states coming from the sensors into the control states needed by the controller. We will call this block the state estimator block (Delansnay & Wouwer, 2022).

Flight Control System block generates the error. It consists of a PID controller. This block is auto coded and loaded onto the mini drone. It has three inputs reference commands, sensor reading and image processing data. Reference command is usually moved into the flight code. There are two outputs motor speeds and stop flag. Stop flag is used for safety protection (Gopalakrishnan, 2017).

The output from the FCS block are motor commands (that are obtained by conversion of the altitude and attitude signals by using motor mixing algorithm) and are played through the plant the airframe dynamics. We can choose between linear and non-linear model (Schaub & Junkins, 2003).

We are aware that non-linear models are excellent for simulations, but we do not want any non-linearities in our system for control design or linear analysis. We require a linear model to tune our six PID controllers, even though it is obviously less accurate than the non-linear model (Kadhim & Abdulsadda, 2022).

We currently have a collection of non-linear models wrapped around a collection of flight code that will be put onto the small drone in the main model. Therefore, in addition to linearizing the flight software, all other subsystems must also be linearized in order to have a linear model of the entire system. As a result, we create an entirely new paradigm for control design devoid of non-linearities. Once we have that simplified model, we can utilise Simulink's PID tuner programme to get the updated controller gains (Merheb et al., 2017).

Since the altitude controller is not connected to any other loops, we will just tune it first. We set the reference instructions for roll, pitch, and yaw to zero to ensure that they are totally excluded from the calculation. Remove these subsystems since we are assuming that state estimation and sensor dynamics have no impact on the control architecture (Musa & Basir, 2021). So, in essence, we are going to presumptively know exactly what the drone's true altitude is thanks to our controller. We are left with a linear airframe model, constant environment subsystem, motor mixing method, and our altitude controller with RPY commands set to zero after the non-linearities have been eliminated. To verify our presumptions, we will test the controller on the non-linear model after we have tuned It (Noordin et al., 2022).

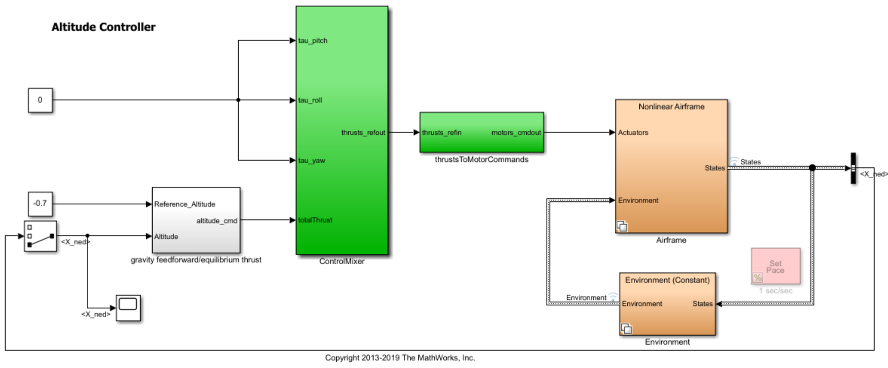

Figure 3. Model for Altitude Controller Tuning

We use a PID block to handle the derivative and apply additional filtering because the lack of sensor data or estimators implies we lack a genuine rate state. Since there are no sensors involved in this model, the derivative of the altitude signal is quite clean, meaning that there is no noise present. The auto-tune is then used to adjust this block (Okasha et al., 2022a).

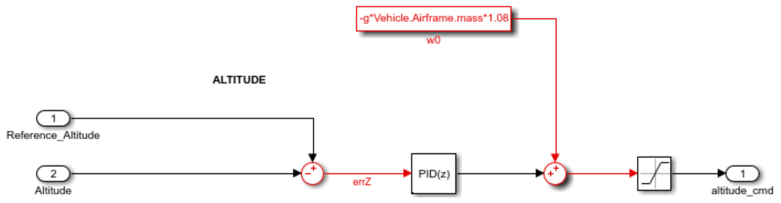

Figure 4. Altitude Controller

We can now see that the control logic for our altitude controller is fairly straightforward. The altitude is measured and contrasted with a reference altitude of 0.7m. The feed-forward gravity term is then added after the mistake has been supplied into the PID controller. To put it simply, this term increases the amount of thrust needed to offset the weight so that the PID controller just needs to increase positive or negative thrust to move up or down (Okasha et al., 2022b).

We will now use the same process to adjust the other PID controllers after we have tuned the altitude controller. The roll pitch and altitude commands for the yaw controller are both set to 0. After yaw, we will tune roll and pitch individually before moving to the outer loop, keeping the inner loop controllers active and maintaining the drone's orientation (Tamayo et al., 2018).

The attitude controller is given as follows. The logic is simple. It serves as the inner loop for the XY-position controller i.e., the output from the position controller serves as the reference for roll and pitch attitudes and is compared with the estimated states.

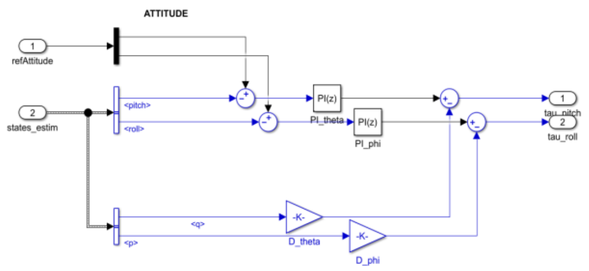

Figure 5. Attitude Controller

Yaw controller is given below. The reference is compared to the estimated yaw state to generate the error term which then passes through a PD controller.

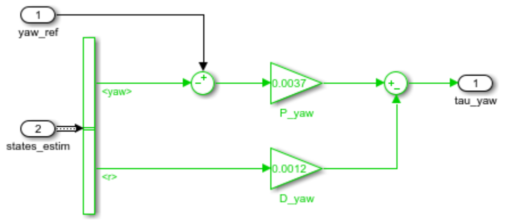

Figure 6. Yaw Controller

The XY position controller is given below. It serves as the outer loop for the roll/pitch controller as told earlier. Rest of the logic is same as used in the linear controller. One thing that was not in the linear controller is that this controller is also taking into consideration the yaw orientation of the drone since it will affect in which direction X and Y axis are and thus the movement of the drone in those directions.

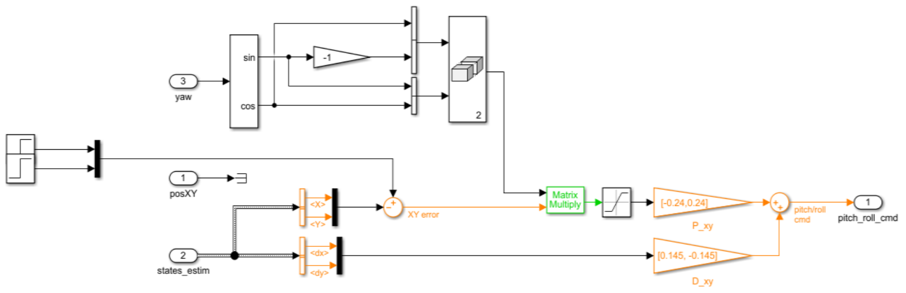

Figure 7. Position Controller

RESULTS AND DISCUSSION

For a bandwidth of 5 rad/s and phase margin of 60 degrees the following response is generated.

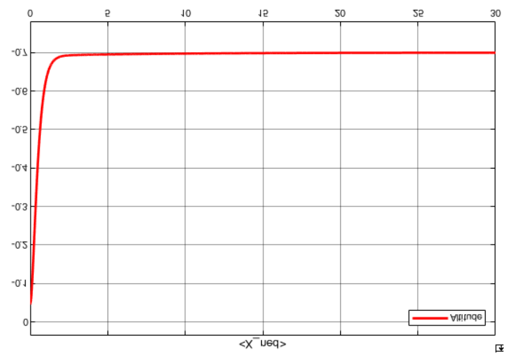

Figure 8. Linear response of tuned altitude controller

We can see that the linear response of the tuned altitude controller shows no overshoot. But we have to test these controller gains on the non-linear model first for verification. The response using the tuned PID on the non-linear model is as follows.

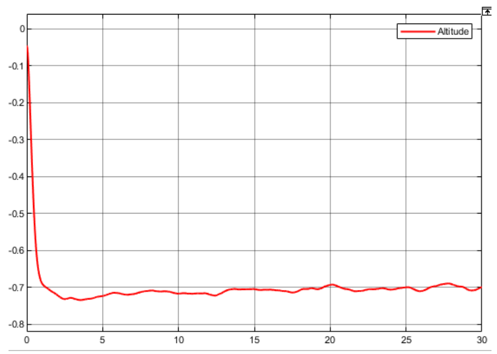

Figure 9. Non-linear response of tuned altitude controller

In a similar way, the other controllers are tuned and simulation is done on a three-dimensional trajectory. The reference that is given to the drone in XYZ space is that the drone has to hover at an altitude of 0.7m. Then at time t=10s a step input of 1m is given in x-direction and then at time t=20s another step input of 1m is given in the y-direction.

The simulation shows that the system hovers slightly above 0.7 meters and there are little deviations from the height it is maintaining when it has to move in x and y directions which makes sense because when a quadrotor moves in lateral or longitudinal axes the thrust of two propellers increase and two propellers decrease. This change in thrust can make the drone lose and gain altitude. The response in x and y directions is shown by the blue plots one respectively. Now, one thing to consider here is that the origin in XY-plane is taken as (57, 95). The simulations show adequate response. Both settles in around 3 seconds.

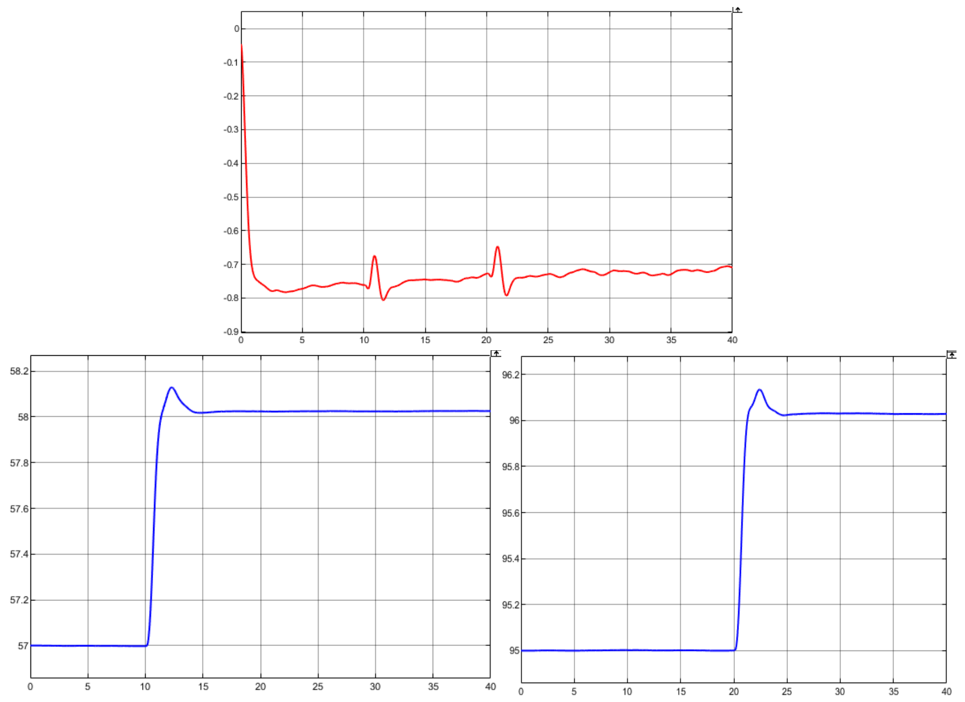

Figure 10. Non-linear x,y and z responses of tuned PID controller

The change in pitch and roll attitudes to produce the required lateral and longitudinal motions can be seen in the following figures blue and red respectively. The quadrotor pitches negatively to produce a positive motion in X-direction and after it has tracked the reference it has to pitch in the opposite direction to bring it back to level. Similar response is produced by the rolling motion except the direction as positive roll produces motion in positive y-direction. The quadrotor does not yaw throughout its trajectory.

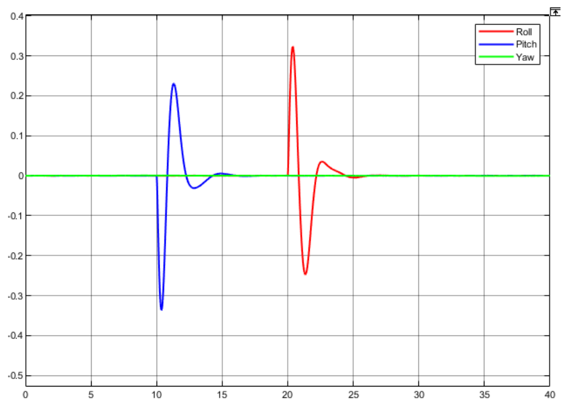

Figure 11. Roll, pitch, and yaw angle

The response for translation in x and y axis, and the velocities are shown below. At 10 seconds, the velocity in x-direction increases as the drone moves in x- direction which then decreases and becomes zero when the drone reaches 1m in x direction. Similarly, the velocity of y is increased as the drone moves in y-direction which then decreases and becomes zero when drone reaches 1m in y-direction. The response in XY-plane is almost same as predicted by the simulations.

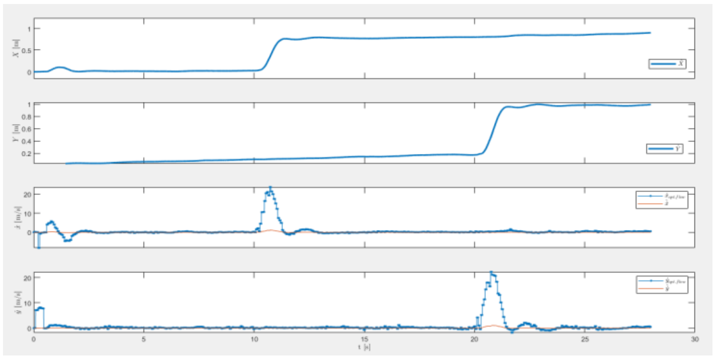

Figure 12. Translation in x, y axis their rates and velocity

The altitude response for all the sensors is given below. The response of pressure and sonar sensor is very noisy. But Kalman filter is able to make adequate prediction of state. The estimate of altitude from Kalman filter is not noisy. The Kalman filter follows the response closely.

.

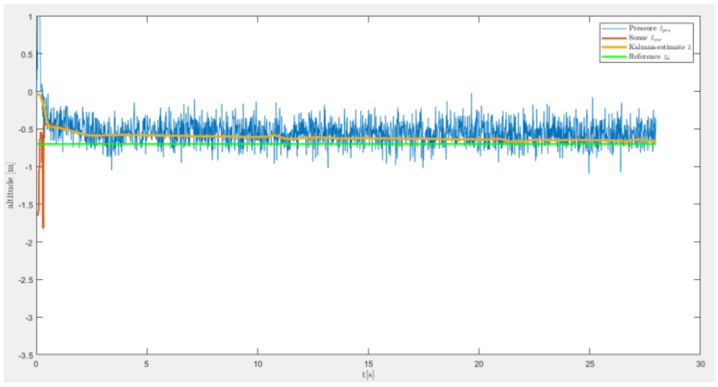

Figure 13. Altitude results

CONCLUSION

In this paper, mathematical model for quadcopter dynamics is obtained by using Newton's Second law. Angular velocity approach is used for input dynamics. The mathematical model for quadrotor dynamics is linearized. Then Simulink support package for parrot mini drone is studied. The non-linear model for parrot mini drone is modified to derive the controllers. Non- linear model is linearized to obtain the gain of PID controller. The verification is done by using the same gains on the non-linear model and testing the simulations in the virtual reality environment and then on actual hardware. Validation is done by comparing the results of simulation and actual flight data.

REFERENCES

Babu, V. M., Das, K., & Kumar, S. (2017). Designing of self tuning PID controller for AR drone quadrotor. *2017 18th International Conference on Advanced Robotics (ICAR)*, 167–172.

Delansnay, G., & Wouwer, A. Vande. (2022). Implementation and Tests of an INDI Control Strategy applied to the Parrot Mambo Minidrone. *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 745–752.

Gopalakrishnan, E. (2017). Quadcopter flight mechanics model and control algorithms. *Czech Technical University*, *69*.

Kadhim, E. H., & Abdulsadda, A. T. (2022). Mini Drone Linear and Nonlinear Controller System Design and Analyzing. *Journal of Robotics and Control (JRC)*, *3*(2), 212–218.

Merheb, A.-R., Noura, H., & Bateman, F. (2017). Emergency control of AR drone quadrotor UAV suffering a total loss of one rotor. *IEEE/ASME Transactions on Mechatronics*, *22*(2), 961–971.

Musa, S. F. P. D., & Basir, K. H. (2021). Smart farming: towards a sustainable agri-food system. *British Food Journal*.

Noordin, A., Mohd Basri, M. A., & Mohamed, Z. (2022). Position and Attitude Tracking of MAV Quadrotor Using SMC-Based Adaptive PID Controller. *Drones*, *6*(9), 263.

Okasha, M., Kralev, J., & Islam, M. (2022a). Design and Experimental Comparison of PID, LQR and MPC Stabilizing Controllers for Parrot Mambo Mini-Drone. *Aerospace*, *9*(6), 298.

Okasha, M., Kralev, J. K., & Islam, M. (2022b). Investigation and realisation of PID and LQR control methods in Parrot Mambo minidrone. *International Journal of Modelling, Identification and Control*, *40*(3), 249–259.

Pérez, I. C., Flores-Araiza, D., Fortoul-Díaz, J. A., Maximo, R., & Gonzalez-Hernandez, H. G. (2014). Identification and PID control for a quadrocopter. *2014 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, 77–82.

Prayitno, A., Indrawati, V., & Trusulaw, I. I. (2018). Fuzzy gain scheduling PID control for position of the AR. Drone. *International Journal of Electrical and Computer Engineering (IJECE)*, *8*(4), 1939–1946.

Schaub, H., & Junkins, J. L. (2003). *Analytical mechanics of space systems*. Aiaa.

Tamayo, A. J. M., Ríos, C. A. V., Zannatha, J. M. I., & Soto, S. M. O. (2018). Quadrotor input-output linearization and cascade control. *IFAC-PapersOnLine*, *51*(13), 437–442.

Zhang, X., Li, X., Wang, K., & Lu, Y. (2014). A survey of modelling and identification of quadrotor robot. *Abstract and Applied Analysis*, *2014*.

Zouaoui, S., Mohamed, E., & Kouider, B. (2019). Easy tracking of UAV using PID controller. *Periodica Polytechnica Transportation Engineering*, *47*(3), 171–177.