Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 179 (2021) 242–251

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Computer Science and Computational Intelligence 2020

Development of control algorithm for a quadcopter

Adilet Tagay[a], Abylkaiyr Omar[a] and Md. Hazrat Ali[a]*

53 Kabanbay Batyr Ave., Nur-Sultan, Kazkahstan.

Abstract

Nowadays, drones are popular with their multipurpose functioning. They can be applied in different environments, especially those that are harmful and can cause health hazards to the human being. However, drones are expensive, have limitations in the lifting capabilities, difficult in control, and auto-balancing. This paper focuses on deriving a mathematical model of the quadcopter with its characteristic properties to solve the auto-balancing problem. The research determines the mathematical model of the unmanned aerial vehicles (UAV) and then incorporates characteristic values of the constructed model (I_{xx}, τ_{thrust} *and etc.*) to the general model. The derived equation is used in identifying the controlling parameters of the quadcopter. The key focus of this research is to develop a cost-effective, self-stabilizing, and robust control system using affordable components. A gyroscope MPU6050, a transmitter, and a receiver (with at least 4 inputs) were integrated with microcontrollers to develop the system.

Keywords: drone, UAV, quadcopter

1. Introduction

Going back through history, unmanned aerial vehicles first invented by Reginald Denny during the First World War I[4]. At that time, the elements to built UAV were costly. The appearance of small-sized gyroscopes gave new propulsion to the development of the UAVs. All the necessary components and mechanisms in building different types of copters are cheap and available today. Nowadays, the application of the quadcopters is increasing widely, especially for dangerous tasks such as observation of the toxic lands, firefighting, rescue operations, and other activities that can be accomplished without physical intervention. This technology gives a chance to avoid severe damages to human

* Md.Hazrat Ali. Tel.: +77172706145
E-mail address: md.ali@nu.edu.kz

10.1016/j.procs.2021.01.003

life. On the other hand, most accidents take place under acute conditions, where unexpected difficulties may affect the work of the quadcopter. Therefore, the main aim of the research is to develop a robust quadcopter with advanced controllability based on the PID controller.

According to Abdelkhalek et al.[5], the quadcopter stabilization was controlled by an angular acceleration plus PD controller, which led to the PD- A and compared it with the regular PD controller of quadcopters. To show that the quadcopter controlled by PD-A is better controlled, theoretical studies and experiments were carried out using quadcopter with only one degree of freedom. After conducting a comparison on Matlab SIMULINK, it was concluded that PD-A decreases the speed, which is undesirable. But, it also declines overshoot, fluctuation amplitude, and frequency. However, disturbance cannot be handled efficiently by the regular PD rather than PD-A[5]. A modeling, control, stabilization with simulation results of a quadcopter is discussed in a previous research[3].

Navabi and Mirzaei also did an experiment and tested 2 suboptimal control methods to improve the controllability of the quadcopters. There are several controllability methods, such as linearized control, sliding mode control, backstepping[2] control, etc. But, the work mainly focuses on the State-Dependent Ricatti Equation (SDRE) to solve the infinite- time nonlinear controlling problem of the flight of quadcopter. As this method works as a closed system, it requires continuous recording from the output each time step. So, as a result, it was concluded that it provides stability to the system. But, the second, Linear Quadratic Regulator (LQR) method failed the test for robust controllability and stabilization[6].

Besides, a unique variant of the controlling method was offered by Jing-Liang and et al[7]. They first made a system that controls the altitude of the quadcopter aircraft with negligible uncertainty by using the SDRE approach. Then they start investigating the sliding method to get the robust and stable quadrotor. According to Jing-Liang et al. (2015), the sliding control method and SDRE control method characteristics were combined among themselves. As a result, quadcopter with robust controller obtained that guarantees the stabilization of the quadcopter.

Choi and Ahn[8] experimented with another method. They offered a backstepping feedback linearization method to control and stabilize the quadrotor. The theorem is made on the base of the Lyapunov theorem. According to paper, the quadcopter has three sub-controllers, which are responsible for controlling attitude, altitude, and the last is point controller. According to the experimental experiments, the quadcopter was able to take off vertically, hovering, and landing.

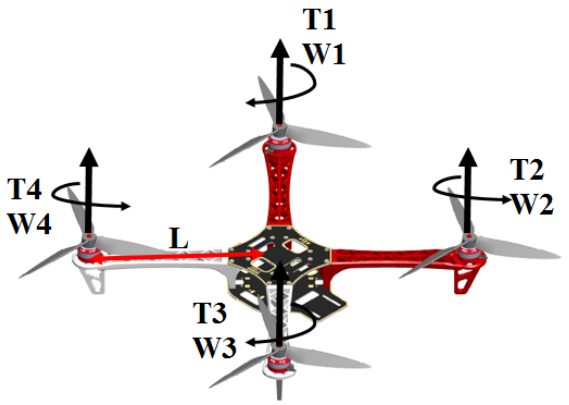
Furthermore, Santos et al[9] proposed the optimal method to the economy the energy of the unmanned aerial vehicle–quadrotor and stabilized it in attitude and position. Optimization of the energy consumption during the hovering and vertical take-off, increase in effective time was reached by applying the LQR (Linear Quadratic Regulator) approach in combination with the nonlinear dynamical model by an exact linearization. Finally, the paper calculated the saved energy on the optimal controller in comparison with the regular PD (proportional, derivative) controller.

Microcontroller plays a vital role as a brain of a system, which receives the data from gyroscope, analyses, and sends the signal to the motors within milliseconds. All the roll, pitch, and yaw, as well as the degrees of freedom, were programmed so that it is changed correspondingly with the tuning of PID coefficients using the integrated formula for PID as below.

$$K_p e + K_d \frac{de}{dt} + K_i \int_0^t e(t)\,dt$$

2. Modeling

The quadcopter can be controlled remotely through the data exchange between the transmitter and receiver.



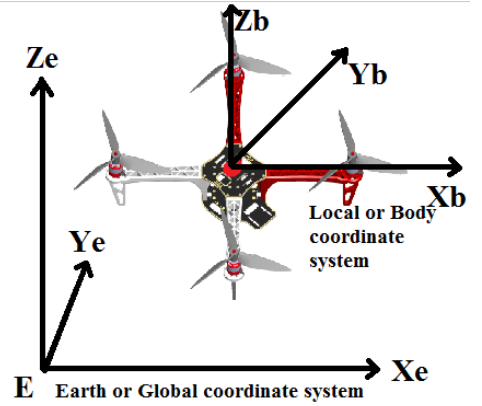


Fig 1. Illustration of the torques Ti and angular velocity directions Wi. Fig 2. Illustration of the Coordinate systems to describe the mathematical model

2.1 Quadcopter Kinematics

Kinematics of any rigid body with six DOF is given by Eqn(1):

$$\dot{\varepsilon} = J\vartheta \tag{1}$$

Here, variables ϑ and $\dot{\varepsilon}$ are generalized velocity and velocity vector, respectively, and both are defined in the body coordinate system. J is a transformation matrix.

$$\varepsilon = [\zeta\ \eta][\zeta\ \ \eta]^T = [X\ Y\ Z\ \phi\ \theta\ \psi]^T \tag{2}$$

$$\vartheta = [\vartheta^B\ \omega^B]^T = [u\ v\ w\ p\ q\ r]^T \tag{3}$$

Generalized rotation and transformation matrix is useful for the transfer of the velocities from a body coordinate system to the global (earth) coordinate system.

$$J = \begin{bmatrix} R & 0_{3\times3} \\ 0_{3\times3} & T \end{bmatrix} \tag{4}$$

Where R is a rotation matrix, and T is an angular transformation matrix

$$R = \begin{bmatrix} cos\psi cos\theta & \cos\psi \sin\theta \sin\phi - \sin\psi \cos\phi & \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi \\ \sin\psi \cos\theta & \sin\psi \sin\theta \sin\phi + \cos\psi \cos\phi & \sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi \\ -\sin\theta & \cos\theta \sin\phi & \cos\theta \cos\phi \end{bmatrix} \tag{5}$$

$$T = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\phi \end{bmatrix} \tag{6}$$

2.2 Quadcopter Dynamics

Newton-Euler method

The mathematical model of the quadcopter dynamics is derived using the Newton-Euler method. This includes specific parameters of the quadcopter, which are total mass (m), mass moment inertia of the body (I). It is presumed that the quadcopter frame is symmetric. Therefore, the inertia matrix is diagonal $I_{xx} = I_{yy}$.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{7}$$

The quadcopter dynamics is illustrated in Eqn (8):

$$\begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega_B \times (mv_{B)} \\ \omega_B \times (Iv_{B)} \end{bmatrix} = \begin{bmatrix} F^B \\ \tau^B \end{bmatrix} \tag{8}$$

In this way, the generalized force vector is identified:

$$\lambda = [\mathrm{F^B} \quad \tau^B] = [\mathrm{F_x} \ \mathrm{F_y} \ \ \mathrm{F_z} \ \ \tau_x \quad \tau_y \quad \tau_z]^T \tag{9}$$

This force vector consists of three components

Gravitational vector

Gravitational vector force acts on quadcopter's gravity center, and its direction in the earth coordinate system always opposite to the Z_E. However, only specific components of the gravitational vector force act on the body frame when the quadcopter is tilted.

$$g_B = \begin{bmatrix} f_G^B \\ 0_{3\times1} \end{bmatrix} = \begin{bmatrix} R^T \ f_G^E \\ 0_{3\times1} \end{bmatrix} = \begin{bmatrix} mgsin\theta \\ -mgcos\theta sin\varphi \\ -mgcos\theta cos\varphi \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{10}$$

Where $\theta \ and \ \varphi$ are roll and pitch angles. f_G^B and f_G^E forces with respect to the body and earth coordinate system, respectively[5].

Movement vector

Thrust force from propulsors (U_1) and torques around body frame (U_2, $U_3 \ and \ U_4$) are variables in the movement vector. Controlling the quadcopter movement directly influenced these variables. The pilot can change the altitude and attitude of the quadcopter by controlling propellers angular velocity (w_i). Thus, the thrust force is a function of:

$$U_1 = b(w_1^2 + w_2^2 + w_3^2 + w_4^2) \tag{11}$$

Where $b = C_T \rho A r^2$ and

Second variable U_2 is a roll torque around X_B axis and function of the angular velocities of the 2nd and 4th propellers.

$$U_2 = \tau_\varphi = bl(w_4^2 - w_2^2) \tag{12}$$

Where l is the distance between propulsors and center of gravity. Third variable U_3 is a pitch torque around Y_B axis and function of the angular velocities of the 1st and 3rd propellers.

$$U_3 = \tau_\theta = bl(w_3^2 - w_1^2) \tag{13}$$

Last variable U_4 is a yaw torque around Z_B axis and function of the pair angular velocities. For example, by increasing angular velocities of the propellers 1 and 3 negative rotation about Z_B axis is observed.

$$U_4 = \tau_\Psi = d(-w_1^2 + w_2^2 - w_3^2 + w_4^2) \tag{14}$$

Where d is the drag coefficient and $d = C_p \rho A r^3$. Finally, the movement vector is given in Eqn. (15).

$$u_B(w) = E_B w^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b & b & b & b \\ 0 & -bl & 0 & bl \\ -bl & 0 & bl & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{15}$$

2.3 Parameter identification

First, using the simple ruler, an arms-length or distance between the gravity center and rotor measured. Besides, the mass of the total quadcopter is measured using a mass scale. The drag coefficient of the model was calculated by the formula:$d = C_p \rho A r^3$, where ρ is the air density, A is an area of the one propeller, r is the radius of the propeller and C_p is the power factor. Thrust coefficient of the model was derived experimentally. Table 1 shows the parameters of the quadcopter.

Table 1. Specific parameters of the quadcopter model

Parameter	Value
Mass (m)	804g
Arm length (l)	0.215m
Drag coefficient (d)	1.523×10^{-7}
Thrust coefficient (b)	$9.78 \times 10^{-6} N/m^2$
Moment of Inertia (Ixx,Iyy,Izz)	0.0034, 0.0034, 0.005 $kg * m^2$
Rotor Inertia (Jr)	$2.76 \times 10^{-6} kg * m^2$

3. Results and Discussion

The derivation of the mathematical model for the quadcopter is shown in this work. The general movement equation was derived using the Newton-Euler method, and it is illustrated in equation (16) as a state-space model. Here, a new specific movement vector for the constructed model is shown.

$$u_B(w) = E_B w^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 9.78 \times 10^{-6} N/m^2 & 9.78 \times 10^{-6} N/m^2 & 9.78 \times 10^{-6} N/m^2 & 9.78 \times 10^{-6} N/m^2 \\ 0 & -2.10 \times 10^{-6} N/m^2 & 0 & 2.10 \times 10^{-6} N/m^2 \\ -2.10 \times 10^{-6} N/m^2 & 0 & 2.10 \times 10^{-6} N/m^2 & 0 \\ -1.523 \times 10^{-7} & 1.523 \times 10^{-7} & -1.523 \times 10^{-7} & 1.523 \times 10^{-7} \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{16}$$

Rotation equations:

$$\ddot{\phi} = 63.235 U_2 - 8.11 \times 10^{-4} \theta \dot{\Omega}_r + \dot{\psi}\dot{\theta} - 1.47 \dot{\theta}\dot{\phi}$$

$$\ddot{\theta} = 63.23 U_3 - 8.11 \times 10^{-4} \phi \dot{\Omega}_r + 1.47 \dot{\phi}\dot{\psi} - 1.47 \dot{\psi}\dot{\phi}$$

$$\ddot{\psi} = 43 U_4 - 0.68 \dot{\theta}\dot{\phi} + 0.68 \dot{\phi}\dot{\theta}$$

Translation equations:

$$\ddot{x} = -\frac{U_1}{0.804} (\sin\phi \sin\psi + \cos\phi \cos\psi \sin\theta)$$

$$\ddot{y} = -\frac{U_1}{0.804} (\cos\phi \sin\psi \sin\theta - \cos\psi \sin\phi)$$

$$\ddot{z} = -\frac{U_1}{0.804}(\cos\phi\cos\theta)$$

3.1 Linear dependency of the gyroscope

Before the gyroscope is programmed, the workability is checked by using the template in the Arduino IDE platform. The results can be seen in Fig. 3.

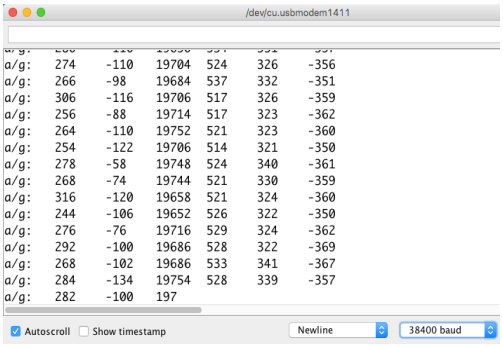


Figure 3. The output of the gyroscope

According to the specifications of the MPU6050 for the maximum 500-degree range, each degree change equal to 65.5 output unit[2].

```
//0.0000611 = 1 / (250Hz / 65.5)
angle_pitch += gyro_pitch * 0.0000611;
angle_roll += gyro_roll * 0.0000611;
```

Figure 4. Conversion of the gyroscope degrees

Also, the address of the gyroscope is found by the I2C scanner template, which is used to retrieve the information from the gyroscope and to balance the quadcopter. First, the second and last rows of Fig. 3 are the outputs of the pitch, roll, and yaw degrees of freedoms, respectively. Thus, if the gyroscope rotates with 10 degrees per second, the output will generate the value equal to 655. This conversion is shown in Fig. 4.

3.2 Hardware components

- Arduino Uno is a microcontroller board used for reading digital/analog input and output values.
- Gyroscope (MPU 6050) identifies the orientation of the body in the 3D space.
- Electronic Speed Controller (ESC) regulates the speed of the motors.
- Lipo battery (2200mAh) is for power supply.
- Transmitter (4 channel) and receiver (Futaba) are used to control quadcopter.
- DC Motors (1000kV) and propeller are used to navigate the quadcopter.

One of the essential parts of the research was the hardware connection of the system. The connection scheme of the system can be seen from **Fig. 5**.

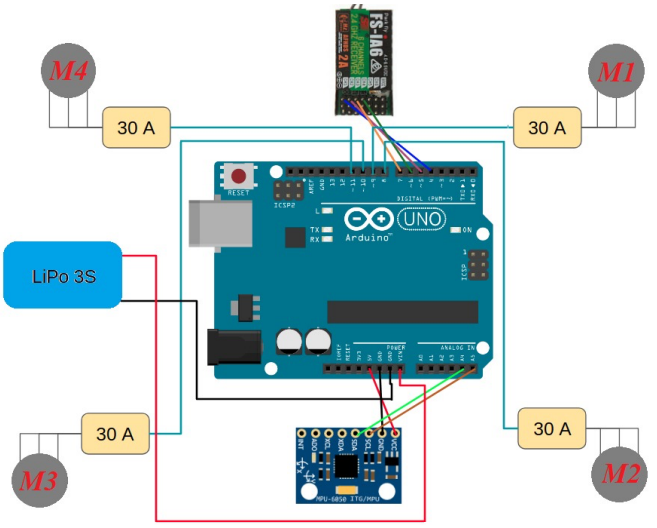


Figure 5. Schematic of hardware connection

According to the scheme, all the parts of the system, including the receiver, transmitter, 4 ESCs, and motors were connected to the controller using conventional jumpers. The development process included testing motors with ESC and Arduino board. Then the corresponding code was implemented using Arduino Software (IDE), and PID values were adjusted. Figure 6 shows the hardware components of the developed quadcopter. After connecting all the necessary parts of the quadcopter, the final weight of the quadcopter is approximately 800 grams.

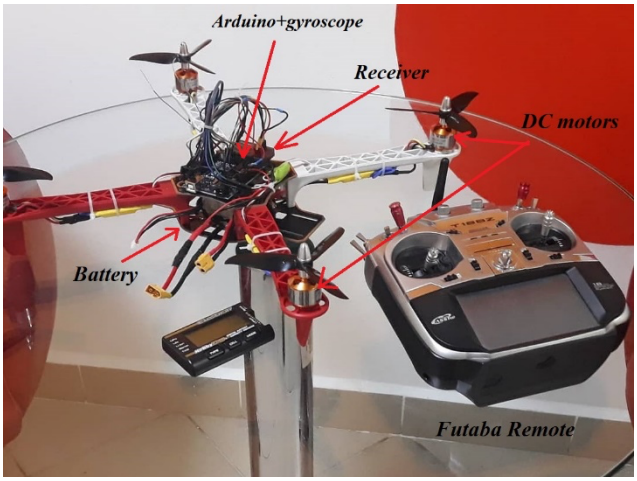


Figure 6. Hardware components of the quadcopter

3.3 Auto-balancing of the quadcopter

The controlling part of the quadcopter can be divided into three steps: setup, ESC calibration, and flight control. The first step is the “Setup” process. In this stage, a lot of variables inserted and defined by the user. Firstly, the program searches for the gyroscope, as it is the main component that would allow getting stabilization of the quadcopter. In this case[1], MPU6050 gyroscope was used. Reviewing the characteristics of the Arduino Uno, the Arduino Uno with chip ATMEGA 328 was used to program the gyroscope. Then, the receiver with at least 4 inputs were connected to the 8,9,10, 11 digital outputs, and connected to the remote controller. The program asks the user to bring all the sticks on remote to the middle point. Then, requests to bring sticks to the maximum position one- by- one

and saves the data (roll, pitch, and yaw).

Further, the user is asked to lift the left part of the quadcopter to assign the left and right sides of the quadcopter. Then, the process is repeated. The changes in the values of the gyroscope are saved to distinguish the sides. Figure 7 below indicates the degrees of freedom of the quadrotor.

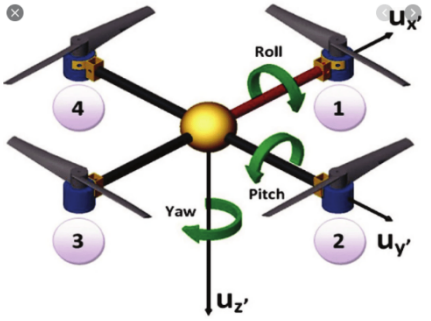


Figure 7. Degrees of Freedom of the quadcopter

The codes above show the directions of the quadcopter, which contain elements such as throttle, pid_output_pitch, pid_output_roll, and pid_output_yaw. Yaw indicates the direction of the props. If negative (-) then, it is CCW, otherwise it is CW. However, the PID_OUTPUT_ROLL shows the locations of the motors. A positive sign (+) indicates the right side, while the negative sign (-) shows the left side of the quadcopter.

Moreover, the PID_OUTPUT_PITCH indicates the sides, whether it is the front or rear side. Negative (-) is the front side, and positive is the frontal area. In this case, the 1st and 4th motors are frontal. Further, the position of the stick values are converted into the pitch, yaw, roll, and throttle that is received from the gyroscope MPU6050. Figure 8 shows the conversion algorithm of the signal.

```
receiver_input_channel_1 = convert_receiver_channel(1);
receiver_input_channel_2 = convert_receiver_channel(2);
receiver_input_channel_3 = convert_receiver_channel(3);
receiver_input_channel_4 = convert_receiver_channel(4);
```

Figure 8. Conversion of the received signals

Furthermore, the stabilization of the system is indicated at the beginning of the program. If the function 'Auto_level' in the code is true, then the quadcopter automatically corrects itself according to the obtained feedback from the gyroscope and level it to initial values measured at the "setup" stage (Fig. 9).

```
float pid_p_gain_roll = 1.3;
float pid_i_gain_roll = 0.04;
float pid_d_gain_roll = 18.0;
int pid_max_roll = 400;

float pid_p_gain_pitch = pid_p_gain_roll;
float pid_i_gain_pitch = pid_i_gain_roll;
float pid_d_gain_pitch = pid_d_gain_roll;
int pid_max_pitch = pid_max_roll;

float pid_p_gain_yaw = 4.0;
float pid_i_gain_yaw = 0.02;
float pid_d_gain_yaw = 0.0;
int pid_max_yaw = 400;

boolean auto_level = true;
```

Figure 9. PID values of each roll, pitch, and yaw

As the motor produces some vibrations, this should be compensated during the angle corrections. Also, to avoid any extra movements, the deadband of 16 microseconds is added to the system. Thus, the system does not react to the change in 8 microseconds in the center of the stick. Where 1500 is the central position of the sticks. Also, the auto-

leveling of the system can be switched off. In such a case, the angle corrections of the pitch and roll DOFs are equated to zero. Further, to stabilize the system, the PID must be calculated as the values are indicated initially (Fig. 9).
Calculation of PID parameters:
The proportional coefficient is equal to:

$$P_{out} = (gyro - receiver) * P_{gain}$$

As can be seen from the formula, if the gyro reading is equal to 10 while the receiver equal to 0 and P gain =1, then the proportional output is equivalent to 10.
Integral coefficient:

$$I_{out} = I_{out} + (gyro - receiver) * I_{gain}$$

The Integral coefficient is the sum of the current and previous readings. So, if the difference of the gyro and receiver equal to 10 while the I gain is 1, then an Integral output is equivalent to 10. But next time, the output increases by 10.
Derivative coefficient:

$$D_{out} = \left(gyro - receiver - gyro_{prev} + receiver_{prev}\right) * D_{gain}$$

D output is the subtraction of the previous readings from the current readings multiplied by the D gain. This output resembles a pulse if we plot the graph of the results. If the current and previous differences of the gyroscope and receiver equal to 10, then the result is equivalent to 0. Overall, the PID output is the summation of all the terms shown above. Finally, additional stability can be achieved by tuning the coefficients of the P, D, and I.

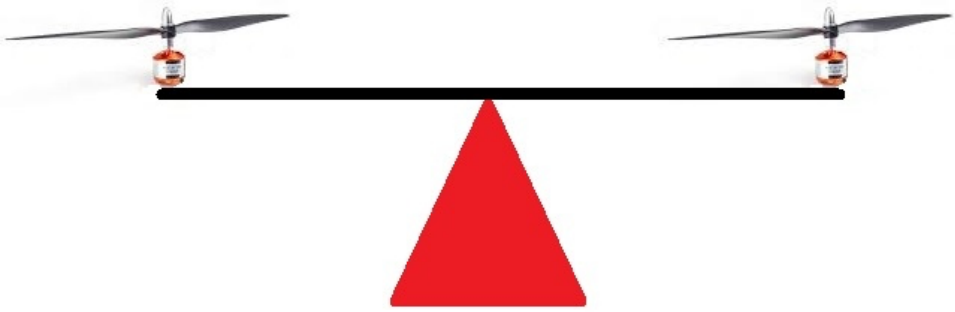


Figure 10. PID testing

Before programming the quadcopter, the influence of the P, I, and D parameters are investigated using two motors with equal props fixed at the same distance from the center position (Fig. 10). All the coefficients are equivalent to 1 at the beginning. Then, the P coefficient is increased by 2 and 3. The fluctuation in the center of the rotation is determined. Further, the I coefficient of the PID was studied. The result is the same; it swings about the center of rotation. The fluctuation is slower than in the previous case as the I parameter value increases each time. The D coefficient is increased while the readings of the P and I coefficients remained 1. So, if one motor is tilted upward, then it would continue this action until the second motor goes to the ground and stays in this position until it is balanced. The same thing happens if the second motor tilted up. It can be concluded that the D coefficient does not reduce angular speed in this situation.

To determine the correct coefficients of the PID controller for the quadcopter, it was calibrated several times using the tuning method. The D coefficient was programmed first. In contrast, the P and I coefficients were equal to 0. The value of D increased until the quadcopter became restless. Thus, the final value declined to 25%, and the D coefficient was set. The P coefficient was tuned. The value of P is increased in the small step of 0.1 each time until

the oscillation of the quadcopter is started. This value then divided into 2 (50%) and set as the value of the P coefficient. The I coefficient was chosen in the same way as for P. But, the size of each step was 0.01. Similarly, when the quadcopter started oscillating, the final value was reduced to 50% and set to as the value of the I coefficient. Finally, the P coefficient is increased in small steps until the stability of the quadcopter is reached.

The main problem during the take-off of the quadcopter was the controllability. However, the controllability can be improved by decreasing the length of the connecting wires. The controllability and stabilization of the quadcopter are dependent on PID parameters. So, setting correct values for the P, I, and D would give some good sensitivity to the system. But, tuning the PID requires some time to re-upload the program and some energy to check the behavior of the quadcopter. Moreover, the Arduino Uno plays a vital role in this project. There are several types of Arduino used. However, only the function of the little chip ATMEGA328 was suitable. The main difference of the controller was in the ability to convert and read the degree changes from the MPU6050 gyroscope, which is discussed in a previous section.

4. Conclusions

To sum up, the main objective of the research was to develop the auto-balancing robust quadcopter. By manipulating and tuning the PID coefficients of the roll, pitch, and yaw values, good results of stabilization, and expected maneuvers were reached using the remote controller. However, there is some time-delay to get some proper coefficients of P, I, and D. All the controlling algorithm of the quadcopter was developed on the Arduino Uno platform. Although, the response time of the controller is big, overall, the system shows excellent performance. To further improve the quadcopter's performance and controllability, several recommendations can be made, such as; accurate tuning of the PID gains through more practical experiments, as well as less wiring to keep the structure always balanced. Finally, the vibrations of the motors and noise should be reduced during the test.

References

1. MPU-6050: TDK. (n.d.). Retrieved from https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/.
2. Kotarski, D., Piljek, P., & Matija, K. (2016). Mathematical modelling of multirotor UAV. *International Journal of Theoretical and Applied Mechanics*, *1*, 233.
3. Luukkonen, T. (2011). Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, *22*.
4. History of unmanned aerial vehicles. (2019, December 3). Retrieved from https://en.wikipedia.org/wiki/History_of_unmanned_aerial_vehicles.
5. Abdelkhalek, M. A., El-Demerdash, M. S., El-Tahan, A. A., & Dief, T. N. (2015). Attitude stability of quadcopter using classic control with angular acceleration. *International Journal of Computer Science and Information Technology and Security*, *5*(4), 325-331.
6. Navabi, M., & Mirzaei, H. (2016, October). θ-D based nonlinear tracking control of quadcopter. In *2016 4th International Conference on Robotics and Mechatronics (ICROM)* (pp. 331-336). IEEE.
7. Jing-Liang, S., Chun-Sheng, L., Ke, L., & Hao-Ming, S. (2015, July). Optimal robust control for attitude of quad-rotor aircraft based on sdre. In *2015 34th Chinese Control Conference (CCC)* (pp. 2333-2337). IEEE.
8. Choi, Y. C., & Ahn, H. S. (2014). Nonlinear control of quadrotor for point tracking: Actual implementation and experimental tests. *IEEE/ASME transactions on Mechatronics*, *20*(3), 1179-1192.
9. Omar Santos et al., "Optimized Discrete Control Law for Quadrotor Stabilization: Experimental Results", J Intell Robot Syst (2016) 84:67–81.