DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN OF SIMULATOR FOR UNMANNED AERIAL VEHICLE OF QUADROTOR

by
İlhan ATEŞ

November, 2012
İZMİR

DESIGN OF SIMULATOR FOR UNMANNED AERIAL VEHICLE OF QUADROTOR

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Electronics Engineering Program

by
İlhan ATEŞ

November, 2012
İZMİR

M. Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DESIGN OF SIMULATOR FOR UNMANNED AERIAL VEHICLE OF QUADROTOR"** completed by **İLHAN ATEŞ** under supervision of **ASSOC. PROF. DR. MEHMET KUNTALP** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Assoc. Prof. Dr. Mehmet Kuntalp

Supervisor

[illegible]

Assoc. Prof. Dr. Zeki KIRAL

(Jury Member)

[illegible]

Asst. Prof. Dr. Ahmet ÖZKURT

(Jury Member)

[illegible]

Prof. Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis advisor Assoc. Prof. Dr. Mehmet KUNTALP for giving me an opportunity to work on an innovative topic such as unmanned air vehicles. I am very grateful for his advice and suggestions. I gained valuable experience by working on this thesis.

I would also like to thank my colleague who is Mr. Emrah YAVUZ (Arçelik R&D Engineer) and my family for their help and endless support.

İlhan ATEŞ

DESIGN OF SIMULATOR FOR UNMANNED AERIAL VEHICLE OF QUADROTOR

ABSTRACT

This work covers a simulator for unmanned air vehicle called quadrotor. For the simulator design, the main components of a real quadrotor are explained and the modeling of them is implemented in the simulation. The main goal of this work is to argue for an idea that a real quadrotor design could be developed upon a well designed simulator with a little effort.

In first sections, the importance of unmanned air vehicles and the reasons for why the quadrotor is commonly used for educational purpose are explained. After that quadrotor working principles explained in detail and the equations of dynamics are derived.

In an unmanned air vehicle, measurement system, control system and communication system have vital importance. All sensors used in a quadrotor are mentioned and the modeling equations of them are derived. Because sensor outputs have different noise characteristics, an effective filtering operation is applied. Furthermore, a well designed control system is required for a stable flight operation and fulfilling duties commanded from a remote operator.

As a last, a three dimensional quadrotor model and medium are designed to visualize all the work done. Simulator helicopter is controlled from a user interface which has number of functional buttons for operator commands and vehicle's instant state values on it.

Keywords: Unmanned air vehicle (UAV), quadrotor helicopter, UAV simulator, inertial measurement systems, control systems.

QUADROTOR İNSANSIZ HAVA ARACI İÇİN SİMULATOR TASARIMI

ÖZ

Bu çalışma, quadrotor adlı insansız hava aracı için simülatör tasarımını anlatmaktadır. Simulator tasarımı için gerçek sistemin temel bileşenleri anlatılmış ve simülasyon ortamında modellemeleri yapılmıştır. Çalışmanın ana amacı gerçek tasarımların, simulator modeli üzerinde çok az çaba ile geliştirilebileceği tezini savunarak iyi bir simülasyon modeli çıkarmaktır.

İlk bölümde insansız hava araçlarının önemi anlatılmış ve quadrotor tipi helikopterin kullanım yaygınlığından bahsedilmiştir. Daha sonra quadrotor çalışma prensipi detaylı bir şekilde anlatılarak, sistemin dinamik denklemleri çıkarılmıştır.

İnsansız hava araçlarında ölçüm sistemi, kontrol sistemi ve haberleşme sistemi hayati önem taşır. Sistemde kullanılan sensörler tek tek incelenmiş ve modelleme denklemleri çıkarılmıştır. Sensörlerden ölçülen verilerin gürültü içermesinden dolayı, gerçek bir ölçüm değeri almak adına filtreleme işlemi uygulanmıştır. Ayrıca, sistemin verilen görevleri yerine getirebilmesi ve dengeli bir şekilde çalışabilmesi için iyi bir kontrol sistemi uygulanmasının önemi üzerinde durulmuştur.

Son olarak, istenilenleri görsel olarak sonuçlarını görebileceğimiz üç boyutlu bir simülator tasarımı gerçekleştirilmiştir. Helikopter modeli tasarımı, ortam ve görsel bir kullanıcı arayüzü aracığıyla sistemin çalışması test edilmiştir.

Anahtar sözcükler: İnsansız hava aracı(İHA), quadrotor helikopter, İHA simülator, inersiyal ölçüm sistemleri, kontrol sistemleri.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

This thesis focuses on the design of a simulator for unmanned air vehicle(UAV) called quadrotor. The UAVs have gained an increasing interest in last years because of the wide area of applications like surveillance, imaging, dangerous environments, navigation, mapping, fire fighting and exploration in military or commercial applications. The quadrotor is one of the most preferred types of UAV that can apply in many application areas mentioned above. The reason is the very easy construction and control using four rotors against the conventional helicopter that uses one main rotor and one tail rotor.

In a conventional helicopter, the vehicle is controlled using many kinds of combined commands that are transmitted to the rotors using complex mechanical systems that consist of so many moving elements. On the other hand, one of the main advantages of quadrotor is the simplicity of the vehicle in terms of control of overall system. The quadrotor has fixed pitch rotor angles and reduced mechanical complexity. The four rotor arrangement also provides more payload capacity, maneuverability and resistance to disturbances. The payload and power consumption of the quadrotor are increased due to the number of motors; but the payload to power ratio is still advantageous and use of electrical motors reduces the power consumption.

The quadrotor has many advantages over conventional helicopters in terms of control and simple construction, but there are still drawbacks that cause it not to be used widely in many of the application areas. The stabilizing and direction control of the quadrotor is a difficult job because dynamics are not linear. The lower lift force and processing power of the onboard electronics are other limitations for a control system design. This means, in order to implement more complex control mechanism such as take off, landing and path following, more suitable sensors might be provided on the quadrotor.

The quadrotor has four propellers driven by four motors in a cross configuration. While the front and the rear motor rotate counter-clockwise, the left and the right motor rotate clockwise. If the rotors rotate at the same speed and if the gyroscopic effects are omitted, the vehicle just hovers. In fact, the thrust generated by each motor has to be exactly the same to provide a stable hover. The slightest change in thrust on one of the rotors will turn upside down the vehicle on a sudden. By changing the speed of the single motors, the lift force can be adjusted and vertical and horizontal motion can be created.

In the simulator design, all necessary components are modeled including sensors, motors, propellers and microcontroller unit. The simulator only takes motor set values and determines its own attitudes according to dynamic equations and outputs the modeled sensor values instantly. The sensor values and user commands are processed in control system and the vehicle is driven into user demands. A simple user interface is designed to take commands and view quadrotor instant state values on it.

CHAPTER TWO
QUADROTOR PRINCIPLES AND DYNAMICS

2.1 Quadrotor Principles

Quadrotor is controlled by applying differential thrusts. The motors are arranged in pairs of reverse rotating motors so that the torque generated from the first pair of motors is exactly reacted by the torque from the second pair of motors, which are rotating in the other direction. With balanced torques, there is no rotating moment and the vehicle does not rotate about the vertical z-axis. This condition is pictured in Figure 2.1. If the thrust generated by each motor is equal to one-fourth of the vehicle's mass, the quadrotor will continue to hover stable. Altitude can be controlled by increasing or decreasing the thrust from each motor by the same amount so that total thrust changes but net torque on the vehicle is zero.

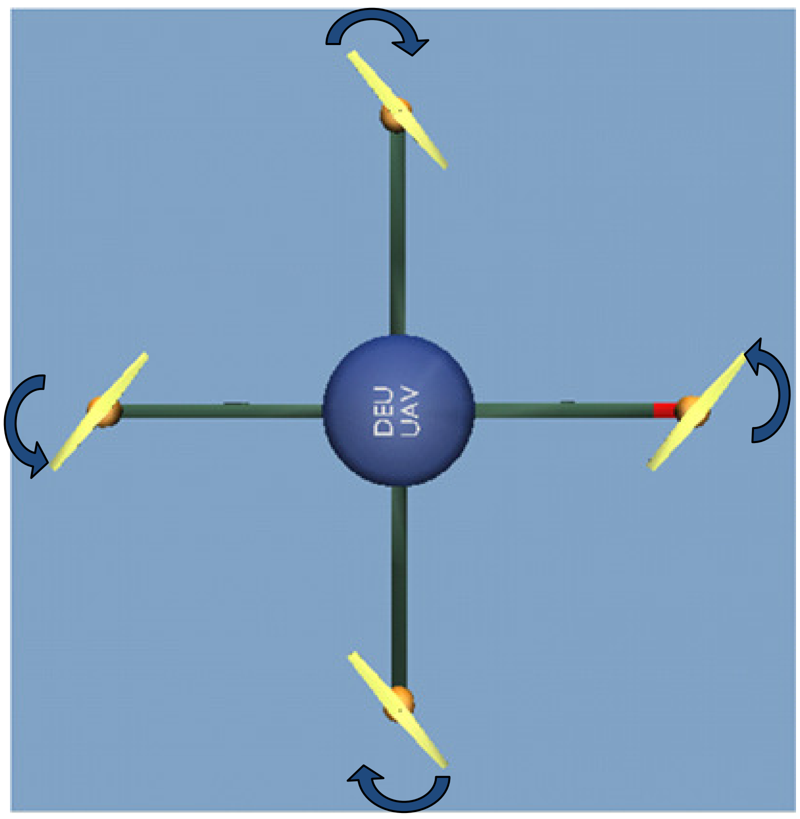


Figure 2.1 Balanced thrust resulting in a hovering platform

Figure 2.2 illustrates the application of differential thrust. The left motor's speed slightly increased and right one decreased same amount. If differential thrust is applied to a pair of opposing rotors, the total thrust does not change. In addition, the net torque about the vertical axis remains zero because the decrease in torque on the slower motor is balanced by the increase in torque on the other motor since they are spinning in the same direction. The difference of thrust between two motors will create a moment in x-axis. In Figure 2.2, the moment is created about the x-axis and the vehicle will roll to the right.

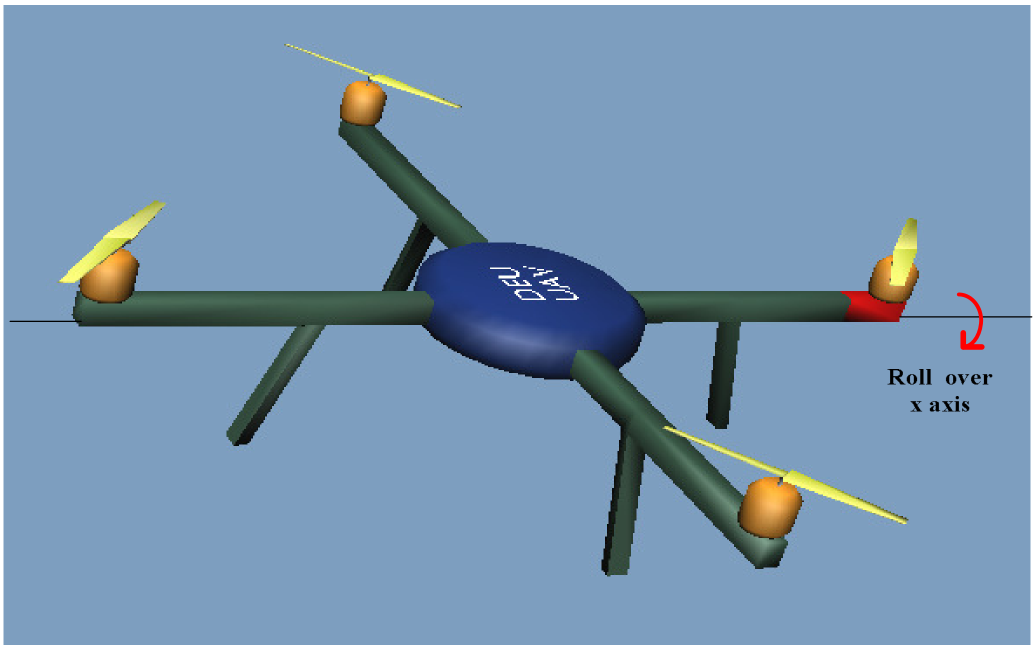


Figure 2.2 Differential thrust resulting in a rolling moment

If the direction of the differential thrust is changed to opposite, the vehicle will roll to the left. Likewise, differential thrust can be applied to the other pair of rotors, this would cause either a pitch up, as shown in Figure 2.3, or a pitch down moment depending on the direction of differential thrust.

When a differential thrust causes the quadrotor to pitch or roll, the total thrust vector is inclined away from the vertical. As a result, part of the lift force is broken into the horizontal and vertical component and this will cause acceleration in inclined direction. For example, in Figure 2.2, the quadrotor is rolling to the right. This tilts the lift force vector to the right and creates an acceleration in right direction. The quadrotor would move towards the right side. When the lift force vector is tilted away from the vertical, there is also a loss of altitude because the amount of thrust on vertical axis is decreased.

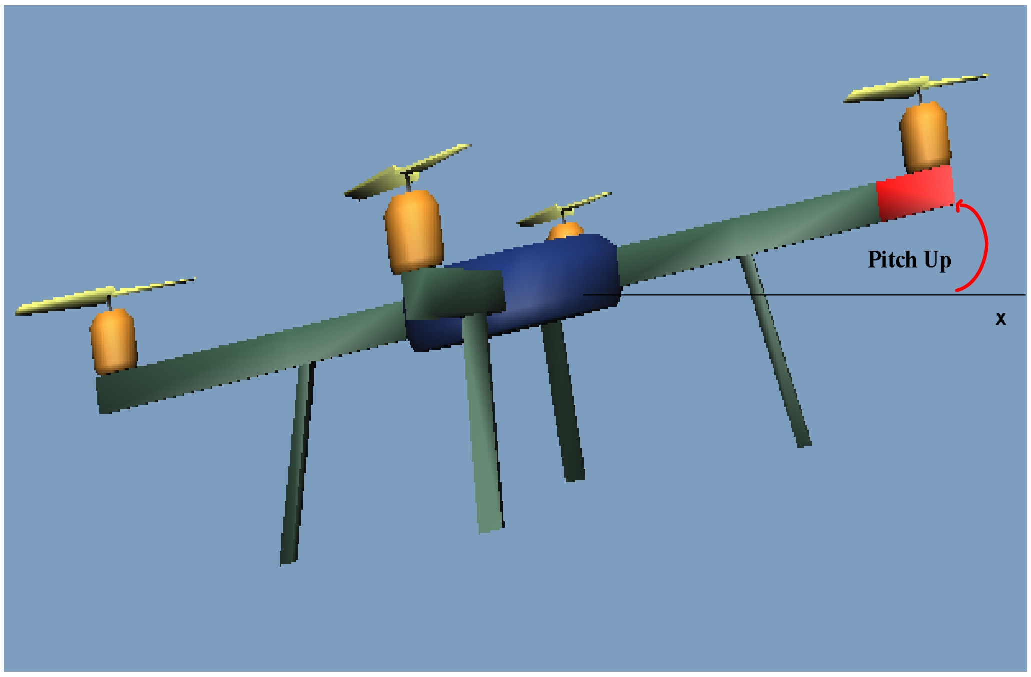


Figure 2.3 Differential thrust resulting in a pitch up moment

In practice, with small angles of rotation, the change is little and can not be noticed by the operator. Drastic changes in the pitch or roll angles, however, would cause the platform to quickly lose altitude.

A change in yaw is managed by applying a differential torque to the two pairs of rotors, as shown in Figure 2.4. Because the total thrust is the same, the altitude does not change, but the unbalanced moments cause the body to rotate about the vertical z-axis.

The operations of balanced thrust, differential thrust about the roll axis, differential thrust about the pitch axis and differential torque are the only ways of controlling the quadrotor as shown in Figures 2.1–2.4

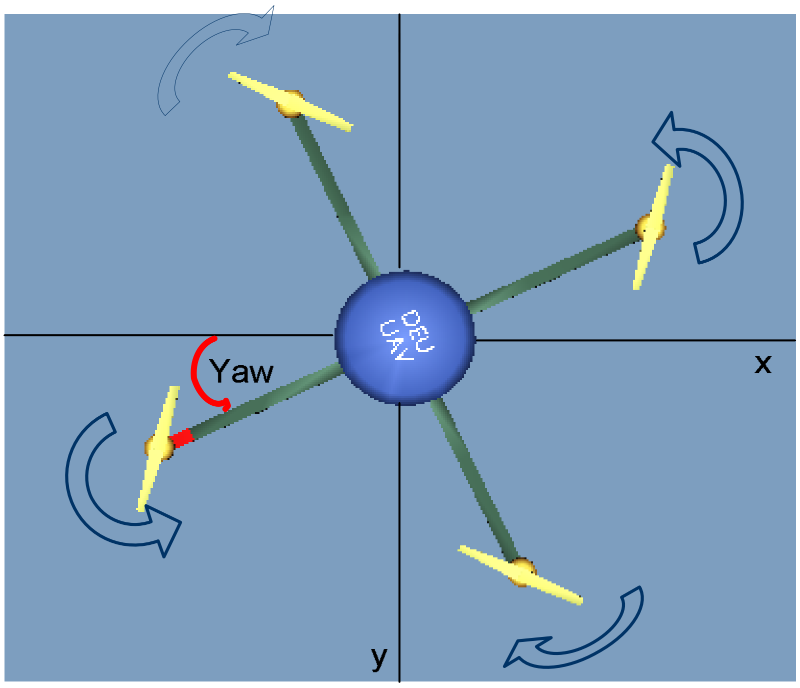


Figure 2.4 Yaw motion with a differential thrust.

2.2 Coordinate Systems

In quadrotor, there are three critical angles as roll, pitch, and yaw. These three important angles of rotation about the center of mass of the vehicle constitute the overall attitude of the system. In order to observe these attitude angles and changes to them as the vehicle experiences a motion, the use of two coordinate systems is required as body(local) frame and global(earth, navigation) frame. The body frame system is attached to the vehicle itself at its center of mass. The global frame system is fixed to the earth and is taken as an inertial coordinate system in order to simplify analysis. The angular difference between these two coordinate systems is adequate to determine the vehicle attitude. In inertial navigation systems, the sensor measurements are taken in body frame. In order to calculate position and velocity, the measurements should be transformed to global frame.

2.2.1 Body Frame

The body frame is a reference frame carried by the tracked object and basis for the onboard inertial sensors. Its origin is at the gravitational center of the object.

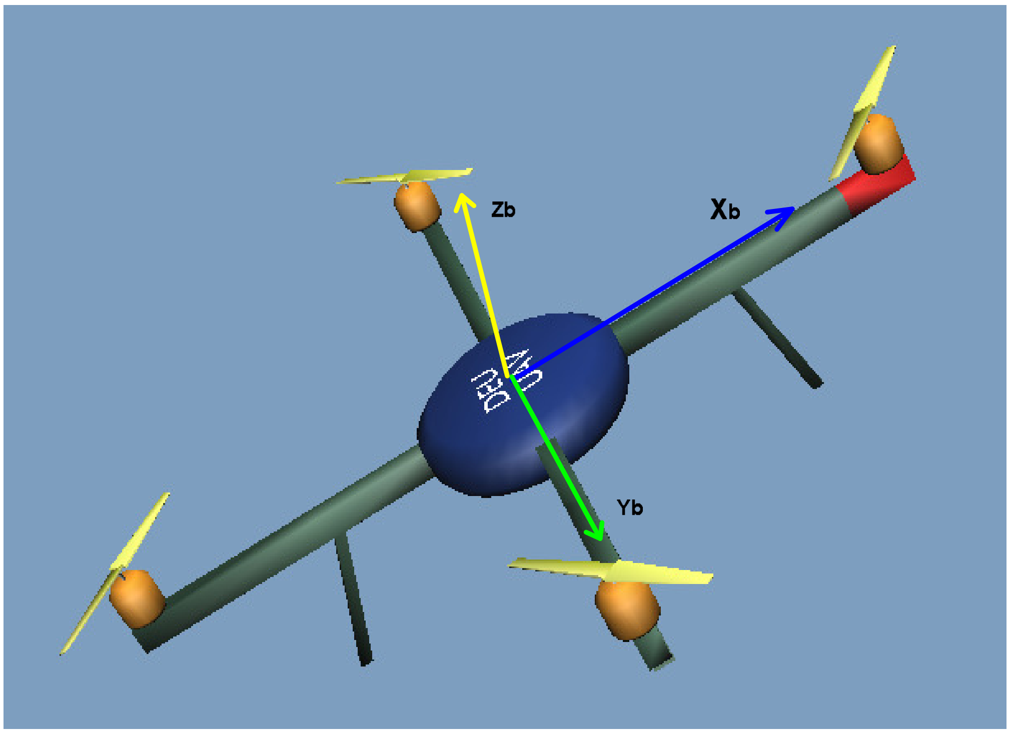


Figure 2.5 Body frame representation

2.2.2 Global Frame (Navigation Frame, Earth Frame)

In the navigation frame, N is represented by the orthogonal vector basis x0; y0; z0 and is attached to the earth. Here x0 points north, y0 points east and z0 points toward the center of earth.

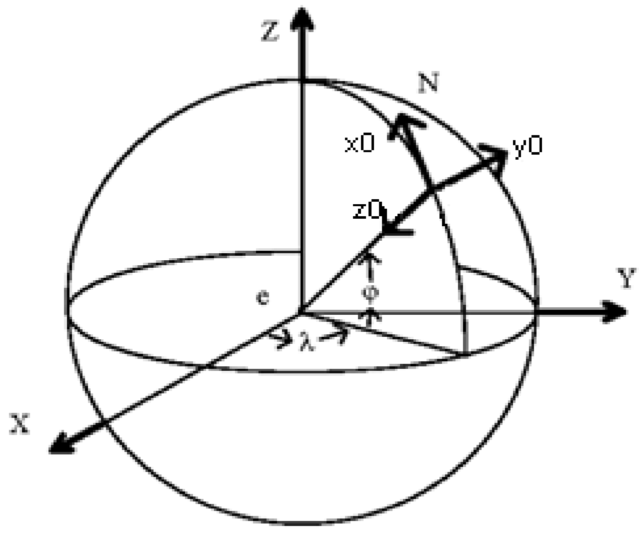


Figure 2.6 Global frame representation

2.2.3 Frame Transformation

The frame (coordinate) system transformation is acquired by using Direction Cosine Matrix. The Euler angle and Quaternion transform method uses the direction cosine matrix to transform. The orientation and movement in the body frame can be transformed to the global frame using transformation matrix.

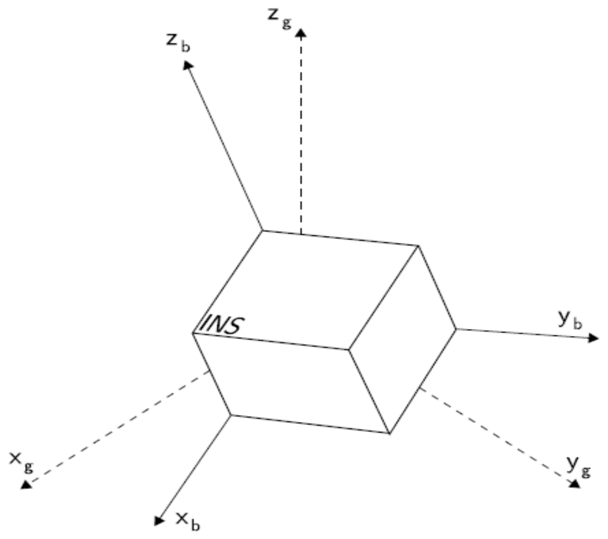


Figure 2.7 Body and global frame

$$P_b^{xyz} = \begin{bmatrix} P^x{}_b \\ P^y{}_b \\ P^z{}_b \end{bmatrix} \tag{2.1}$$

$$P_g^{xyz} = \begin{bmatrix} P^x{}_g \\ P^y{}_g \\ P^z{}_g \end{bmatrix} \tag{2.2}$$

$$P_g^{xyz} = C_b^{g} \cdot P_b^{xyz} \tag{2.3}$$

2.2.3.1 Direction Cosine Matrix

Direction cosine matrix consists of three columns with unit vectors projected along transformed axis from original axis. If the body frame is converted to the global frame, the direction cosine transformation matrix;

$$C_g^{\ b} = \begin{bmatrix} \cos(\theta_{xx'}) & \cos(\theta_{xy'}) & \cos(\theta_{xz'}) \\ \cos(\theta_{yx'}) & \cos(\theta_{yy'}) & \cos(\theta_{yz'}) \\ \cos(\theta_{zx'}) & \cos(\theta_{zy'}) & \cos(\theta_{zz'}) \end{bmatrix} \tag{2.4}$$

Hereθ_{xy} denotes the angle between the body coordinate x axis and the global coordinate y axis.

2.2.3.2 Euler Angle

The Euler angle defines the coordinate transformation using angular rotation. For the transformation from navigation axes to body axes, firstly we rotate about z-axis through an angle ψ, then we rotate through an angle θ about the new y-axis and finally the system is rotated about new x-axis through an angle φ. These are so called yaw, pitch and roll angles.

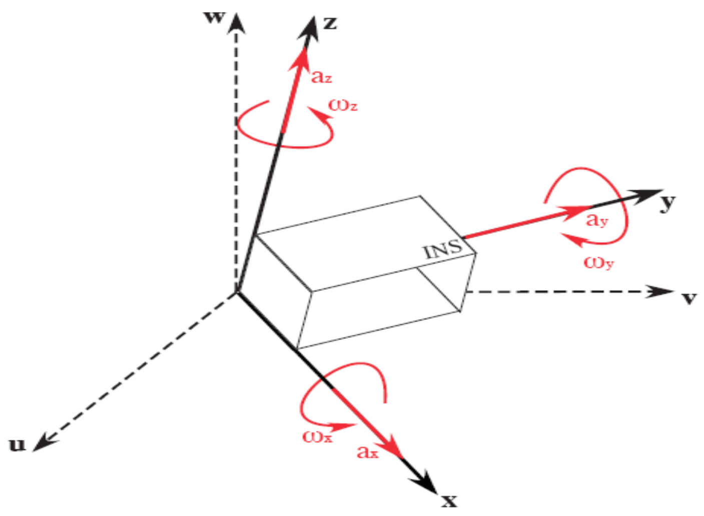


Figure 2.8 Eular angle rotations

The rotation matrices for each axis;

$$C\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.5}$$

$$C\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.6}$$

$$C\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

From these three separate matrices, the overall coordinate transformation matrix between body frame and navigation frame presented with Euler angles is defined as;

$$C_b^{\ n} = C\psi . C_\theta . C_\phi \tag{2.8}$$

2.2.3.3 Quaternion Angles

For two coordinate systems, there is always one invariant axis. Using this invariant axis and rotations around it, it is possible to place one axis to other. The quaternions use an orientation unit vector γ to describe the rotation axis and the angle δ to describe the rotation around it. The angles directing the orientation unit are α, β and γ.

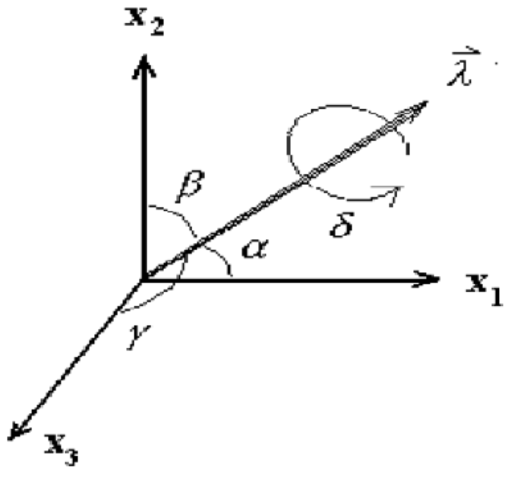


Figure 2.9 Quaternion angle rotation

The coordinate system rotations are put together in four parameters q0, q1, q2 and q4, forming a Quaternion vector E.

$$q_0 = \cos\frac{\delta}{2}, q_1 = \cos\alpha.\sin\frac{\delta}{2}, q_3 = \cos\beta.\sin\frac{\delta}{2}, q_4 = \cos\gamma.\sin\frac{\delta}{2} \tag{2.9}$$

The Direction Cosine Matrix can be expressed in terms of these four parameters as follow.

$$R_q = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.10}$$

2.3 Quadrotor Dynamic States

The roll angle Φ, the pitch angle θ and the yaw angle ψ will be represented in the statevector. Additionally, the angular velocities of these about each axis will be represented as $\dot{\Phi}, \ \dot{\theta}, \ \dot{\psi}$. These six states define the attitude of the vehicle with respect to body frame. Another six states are necessary to represent the attitude of the vehicle with respect to the earth frame. These states include the coordinates of the vehicle within the earth frame along each of its principal axes, shown as X, Y, and Z. Moreover, the velocity of the vehicle in each of these directions is also required and shown as $\dot{X}, \ \dot{Y}, \ \dot{Z}$. Together, these 12 state variables constitute the state vector of the quadrotor platform. This state vector is provided in equation 2.11 below.

$$X = [\ \Phi \ \Theta \ \psi \ \dot{\Phi} \ \dot{\Theta} \ \dot{\psi} \ X \ Y \ Z \ \dot{X} \dot{Y} \dot{Z}] \tag{2.11}$$

2.4 Quadrotor Forces and Moments

In order to design an accurate model of the system, all the forces and moments on the vehicle must be taken into consideration. When these forces and moments are mentioned, some assumptions are made in order to simplify analysis. The vehicle’s movement and attitude is provided by forces and moments created on it. Each of the forces can be broken into an x, y, and z component. The following Newton-Euler equations 2.12 and 2.13 define the total effect of the net forces and moments on the vehicle. The basic dynamic equations for linear and rotational motion:

$$\mathrm{F} = \mathrm{Ma} \tag{2.12}$$

$$\tau = \mathrm{Ia} \tag{2.13}$$

If the two matrix equations for force and torque are combined, the basic dynamic equations 2.14-2.17 for the quadrotor are obtained.

$$\begin{bmatrix} F_X \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \begin{bmatrix} a_X \\ a_y \\ a_z \end{bmatrix} = ml_{3x3} \begin{bmatrix} a_X \\ a_y \\ a_z \end{bmatrix} \tag{2.14}$$

$$\begin{bmatrix} \tau_X \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} l_{xx} & l_{xy} & l_{xz} \\ l_{xy} & l_{yy} & l_{yz} \\ l_{xz} & l_{yz} & l_{zz} \end{bmatrix} \begin{bmatrix} a_X \\ a_y \\ a_z \end{bmatrix} \tag{2.15}$$

$$\begin{bmatrix} ml_{3x3} & 0_{3x3} \\ 0_{3x3} & I_{xyz} \end{bmatrix} \begin{bmatrix} \dot{V}^B \\ \dot{\omega}^B \end{bmatrix} + \begin{bmatrix} \omega^B x\,(mV^B) \\ \omega^B x\,(I_{xyz}\omega^B) \end{bmatrix} = \begin{bmatrix} F^B \\ \tau^B \end{bmatrix} \tag{2.16}$$

$$\begin{bmatrix} ml_{3x3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega\, x\,(mV) \\ \omega\, nx\,(I\omega^B) \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{2.17}$$

The important parameters in designing the control system are the $\dot{\mathrm{V}}$(change ofbody linear velocity) and the $\dot{\omega}$(change of body angular velocity). If the differential is carried out through the parameters that define the various axes and angles available to these velocities, state variables that will determine the attitude of the vehicle to the control system are found.

Figure 2.10 represent how the forces influenced the system based on which coordinate system is referenced. In the left picture of Figure 2.10, global and the body frame are aligned in the z-axis direction, the thrust produced by the rotors is the same for both coordinate system. As the vehicle experiences a roll or pitch angle change as in the right picture of Figure 2.10, the alignment of the body and earth frame disappears. All of the thrust force is still available to the body coordinate system, but only some part of it exists in the z axis of the earth frame. This demonstrates the need of the rotation matrices previously derived and will be useful in describing the vehicle in both systems for attitude calculation and translational motion.

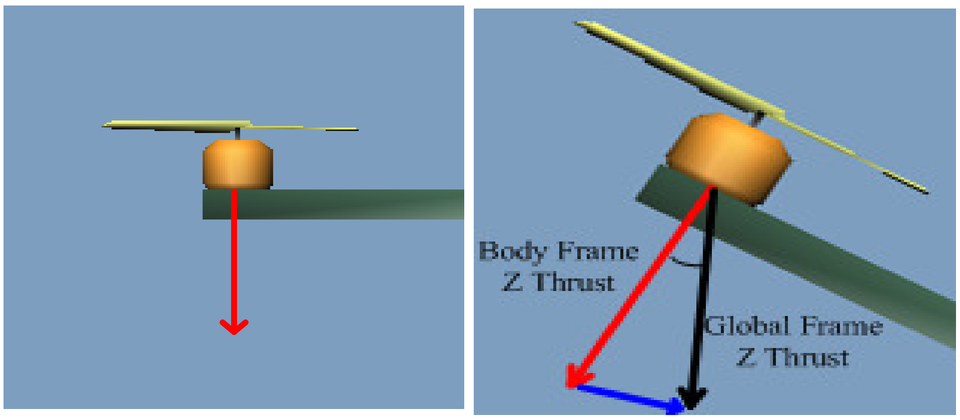


Figure 2.10 No roll/pitch and roll/pitch angle exist

$$Z_{\text{earth}} = Z_{\text{body}} \,.\, \cos\,(\text{angle}) \tag{2.18}$$

From the reference of the body coordinate system, the thrusts produced by the motors are always in the vehicle z-axis direction. The gravity vector is always in the global frame z direction that points the center of the earth. In this case, it is important to use the rotation matrix. Therefore the force of gravity can be defined as in equation 2.19.

$$F_g = mg \begin{bmatrix} -sin\theta \\ cos\theta sin\phi \\ cos\theta cos\phi \end{bmatrix}_{body} \tag{2.19}$$

It is important to note that the force of gravity is observed with respect to the vehicle body coordinate system that attached to the center of mass of the quadrotor platform. Besides gravity, the other forces to be considered are the forces produced by the motors. If these forces are combined with gravitational force, all forces acting on quadrotor platform and acceleration of the vehicle can be found in terms of the body frame.

$$\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = -\frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ F_{\text{thrust}} \end{bmatrix} + g\begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} - \begin{bmatrix} \dot{\theta}\dot{Z} - \dot{\psi}\dot{Y} \\ \dot{X} - \dot{\phi}\dot{Z} \\ \dot{\phi}\dot{Y} - \dot{\theta}\dot{X} \end{bmatrix} \tag{2.20}$$

In last matrix, the rotational and translational velocities are the result of the cross product of the ω and V time derivatives in equation 2.17. The only thrust in the body frame is in the z-direction. To simplify the simulation model, the frictions produced by the air on the rotors will be ignored in the x and y directions.

At this point, another assumption should be noted. On take off and landing, there are important aerodynamic changes because of ground effect. At low altitudes, a decrease in the generated air stream velocity provides greater efficiency from the rotor and so provides more thrust. The ground effect will be omitted when developing the simulation model of the quadrotor platform.

All the moments should be considered to calculate the acceleration rates of the each attitude angles. Each of the three angle accelerations is under the influence of the gyroscopic effect. The gyroscopic effect is the moment generated by the angular velocity of the frame. The following equations show the gyroscopic effects on each angle of rotation.

$$\text{Roll angle gyro effect } = \dot{\theta}\psi(I_y - I_Z) \tag{2.21}$$

$$\text{Pitch angle gyro effect} = \dot{\phi}\psi(I_z - I_x) \tag{2.22}$$

$$\text{Yaw angle gyro effect} = \dot{\theta}\phi(I_x - I_y) \tag{2.23}$$

The equations prove that a change at the velocities of other angles directly influence the acceleration of the target angle. Another moment to discuss is the moment generated by the rotor thrusts. This moment only affects the roll and pitch angles. The equations 2.24 and 2.25 illustrate this moment.

$$\text{Roll angle moment } = \text{ r } (-T_2 + T_4) \tag{2.24}$$

$$\text{Pitch angle moment } = \text{ r } (T_1 - T_3) \tag{2.25}$$

Although the yaw angle is not influenced by the thrust moment, it is still affected by the various rotor thrusts due to unbalance in the counter rotating torques. While the thrusts are balanced, the yaw angle change should be zero, if any external noise or disturbance is neglected. The yaw angle can be controlled with unbalanced thrusts. It eliminates the use of an anti-torque rotor as in conventional helicopter. The equation for thrust unbalance is shown in below.

$$\text{Thrust unbalance } = (T_1 + T_3 - T_2 - T_4) \tag{2.26}$$

The individual moments from the rotor also creates gyroscopic effects. These effects are related with the inertia of rotor, speed of rotor and change of attitudes. The rotor gyroscopic effects are illustrated as below.

$$\text{Roll rotor gyro effect } = J_r \dot{\theta} \Omega_r \tag{2.27}$$

$$\text{Pitch rotor gyro effect } = J_r \dot{\phi} \Omega_r \tag{2.28}$$

The inertial anti-torque moment on the z-axis is similar to the rotor gyroscopic effects for the x and y axis.

$$\text{Inertial anti} - \text{torque effect} = J_r \Omega_r \tag{2.29}$$

The gyroscopic moments have no important effects in the attitude of the vehicle. They are introduced to present a more accurate model, but will not be implemented in the simulation and design of the control system in order to simplify the overall complexity of the platform. All moments together constitute the overall behavior of the principle attitude angles of the vehicle. The equations showing the acceleration of rotation around each axis are given in equations 2.30-2.32 below.

$$\ddot{\phi} = \sum \frac{\tau_x}{I_x} = \left[\dot{\theta}\dot{\psi}\left(I_y - I_z\right) + r(-T_2 + T_4) + J_r \dot{\theta} \Omega_r\right]\left(\frac{1}{I_x}\right) \tag{2.30}$$

$$\ddot{\theta} = \sum \frac{\tau_y}{I_y} = \left[\dot{\phi}\dot{\psi}(I_z - I_x) + r(T_1 - T_3) + J_r \dot{\phi} \Omega_r\right]\left(\frac{1}{I_y}\right) \tag{2.31}$$

$$\ddot{\psi} = \sum \frac{\tau_z}{I_z} = \left[\dot{\theta}\dot{\phi}\left(I_x - I_y\right) + r(T_1 + T_3 - T_2 - T_4) + J_r \Omega_r\right]\left(\frac{1}{I_z}\right) \tag{2.32}$$

CHAPTER THREE
INERTIAL MEASUREMENT

Inertial measurement systems use sensors to track the position and orientation of an object without any external knowledge. Generally, Inertial Measurement Unit (IMU) is used which consist of three axis gyroscope and three axis accelerometer sensors. The gyroscopes measure the angular velocity and accelerometers measure the acceleration. Using these measurements, it is possible to find the position and orientation of a body with a known reference position, velocity and orientation information.

Inertial measurement is used in wide variety of areas such as the navigation of air vehicle, submarines, spacecrafts, missiles and ships. With MEMS (Micro Electro Mechanical Systems) technology this application area has widened greatly.

3.1 Inertial Measurement Sensors

Mostly used inertial measurement sensors are gyroscopes, accelerometers and altimeters. There are also several sensors helping to measurements like ultrasonic sensor, magnetometer, pressure sensor and temperature sensor.

3.1.1 Gyroscopes

Gyroscope is a device which provides angular velocity around operating axis. Using three gyroscopes with orthogonal to each other, it is possible to sense the total orientation of related body. Generally, it makes use of the Coriolis Effect to do it. The coriolis is the effect of the force when a rotating object is deviated from its angular course. The main types of gyroscope are mechanical, optic and MEMS.

3.1.1.1 Mechanical Gyroscopes

Mechanical gyroscope consists of a spinning wheel which can rotate in all three axes with the gimbaled system. When it is subjected to a rotation, the wheel will remain at a constant global orientation and the angles between adjacent gimbals will change. The angle between gimbals is a measure of orientation. In contrast to the other types which measure angular rotation, the mechanical gyroscopes measure the orientation. It contains moving parts, so mechanical friction occurs and the output drifts.

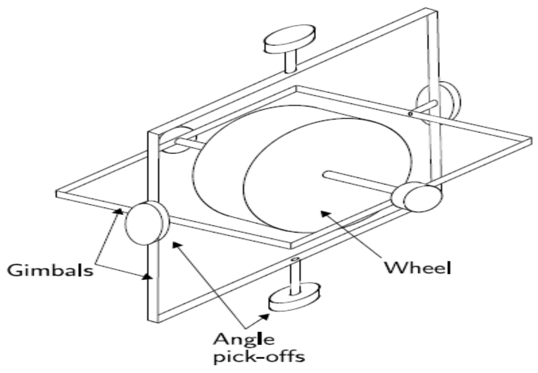


Figure 3.1 Mechanical gyroscope

3.1.1.2 Optical Gyroscopes

Fiber optic gyroscope consists of large coil of fiber optic cable. To measure the angular velocity, two light beams are fired in the opposite direction. The beam traveling in the direction of the rotation will travel in longer path than the other. When the beam travel ends, two beams will combine. The intensity of the combined beam is the angular velocity. The accuracy of the fiber optic gyroscope is directly related with length of path.

3.1.1.3 MEMS Gyroscopes

MEMS gyroscopes make use of the Coriolis Effect which states that in a frame of reference rotating at angular velocity ω, a mass m moving with velocity v experiences a force called Coriolis.

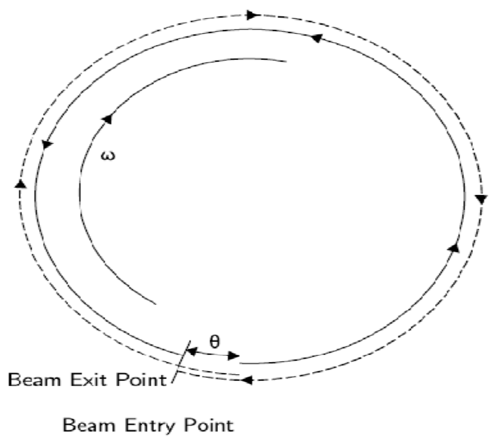


Figure 3.2 Optical gyroscope

$$F_c = -2m(\omega \, x \, v) \tag{3.1}$$

When the mass vibrating along with drive axis is rotated, a vibrating force is produced with sense axis. The produced Coriolis force is directly related with angular velocity.

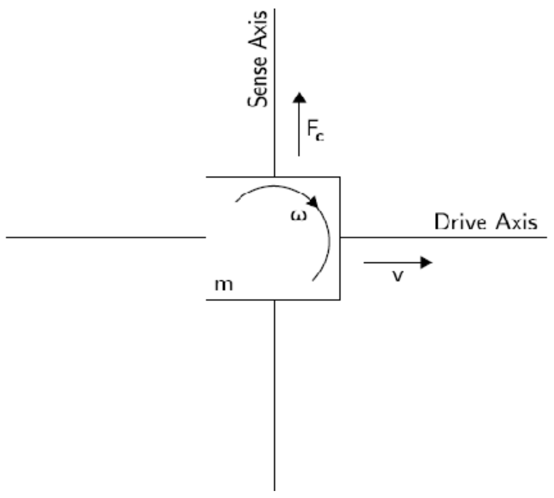


Figure 3.3 The Coriolis force

The MEMS gyroscopes have the advantages of small size, low weight, rigid construction, low power consumption, short start-up time, inexpensive, high reliability and low maintenance. The main disadvantage of the MEMS gyroscope is that it is far less accurate as compare to optical ones.

3.1.2 MEMS Gyro Error Characteristics

3.1.2.1 Constant Bias

The constant bias is the average output from the gyroscope when any rotation is not applied on it. It grows linearly with time and can be calculated by averaging the output of gyroscope when a rotation is not applied.

In unaided systems, the gyroscope bias causes an increasing angular error over time with fatal errors as a result. If the system is integrated with low drift gyroscopes, the real attitude can be estimated.

3.1.2.2 Thermo-Mechanical White Noise

The MEMS gyro readings will be suffered by some thermo-mechanical noise which oscillates at a rate much greater from the sampling period of the readings. As a result, the samples obtained from the sensor are influenced by a white noise, which is simply a sequence of zero-mean uncorrelated random variables.

3.1.2.3 Flicker Noise

The bias of a MEMS gyroscope changes over time because of flicker noise in the electronics and in other components. Flicker noise is noise with a 1/f spectrum, the effects of which are usually observed at low frequencies in electronic components. At high frequencies flicker noise tends to be surpassed by white noise. Bias oscillations due to flicker noise are usually modeled as a random walk.

3.1.2.4 Temperature Effects

Temperature changes in the environment and sensor self heating create a movement in the bias.

3.1.2.5 Gyroscope Misalignment

A misalignment between the gyro axes and the object body axes gives a cross coupling error angle between the measuring axes.

3.1.3 Accelerometers

The accelerometers are used to measure the velocity and position in inertial navigation systems. Three types of accelerometer are available as; mechanical, solid state and MEMS accelerometers.

3.1.3.1 Mechanical Accelerometers

A mechanical accelerometer consists of suspend mass with springs. When acceleration is applied, the mass is displaced along with input axis. This displacement is a measure of acceleration.

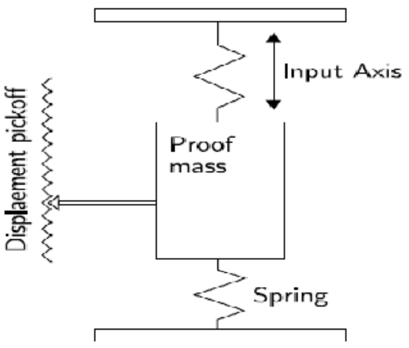


Figure 3.4 Mechanical accelerometer

3.1.3.2 Solid State Accelerometer

Solid-state type accelerometers can be designed with several types as surface acoustic wave, vibratory silicon and quartz devices. Solid state accelerometers are small, reliable and robust. Mostly, the surface acoustic wave (SAW) accelerometers are used. A SAW accelerometer consists of a cantilever beam which is resonated at a particular frequency. A mass is fixed to one end of the beam which can move. The other end is rigidly attached to the case. When acceleration experienced along the input axis, the beam bends. This causes the frequency of the surface acoustic wave to change proportionally to the applied acceleration. The change in frequency is the measure of the acceleration.

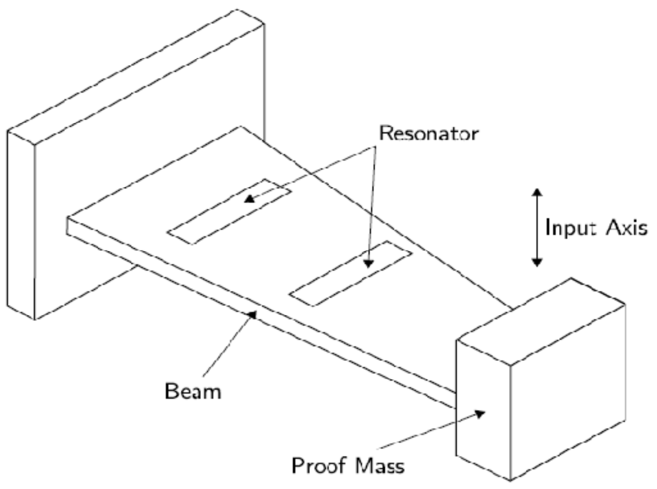


Figure 3.5 Solid state accelerometer

3.1.3.3 MEMS Accelerometer

Micro-machined electromechanical accelerometers use the same principles as mechanical and solid state sensors. There are two main classes of MEMS accelerometer. The first class consists of mechanical accelerometers manufactured using MEMS technique. The second class consists of devices which measure the change in frequency of a vibrating element caused by a change of tension, as in SAW accelerometers.

They are small, light and have low power consumption and short start-up times. Their main disadvantage is that they are not currently as accurate as accelerometers manufactured using traditional techniques but the performance of MEMS devices are improving rapidly.

3.1.3 MEMS Accelerometer Error Characteristics

The main difference between errors coming from accelerometers than gyroscope's is that they are integrated twice in order to track position, where as gyro signals are integrated once to track the orientation.

3.1.4.1 Constant Bias

The accelerometer bias is the difference of its output signal from the true value. A constant bias causes high erroneous position results because the accelerometer output is double integrated. So the position error grows dramatically.

The constant bias is estimated by measuring the long term average of the accelerometer output when it is not applied any acceleration. But this is complicated by gravity, since a component of gravity acting on the accelerometer will appear as a bias.

3.1.4.2 Flicker Noise

MEMS accelerometers are subject to flicker noise, which causes the bias to change over time. Such changes are usually modeled as a bias random noise. Using this model, flicker noise creates a second order random noise in velocity whose uncertainty grows proportionally to $t^{\frac{3}{2}}$ and a third order random noise in position which grows proportionally with $t^{\frac{5}{2}}$.

3.1.4.3 Accelerometer Offset

An offset is a displacement of the accelerometers from the body axes which causes an error in the alignment of the system.

3.1.4.4 Temperature Effects

Temperature changes cause oscillations in the bias of the output signal. The relationship between bias and temperature is often highly nonlinear. Any residual bias introduced causes an error in position which grows dramatically with time. If the IMU contains a temperature sensor then it is possible to apply corrections to the output signals in order to compensate for temperature effects.

3.1.4.5 Calibration Errors

Calibration errors appear as bias errors which are only visible as the device is experiencing acceleration.

3.1.4 Altitude Sensors

Ultrasonic sensors can be used for altitude measurement. They use sounds which are at a higher frequency than human ear can hear. They emit a pulse of sound and listen for echoes. By measuring the time it takes for an echo to return from an object, they are able to calculate the distance to that object. Knowing the speed of sound in the air, distance to the target object is given by equation 3.2 below.

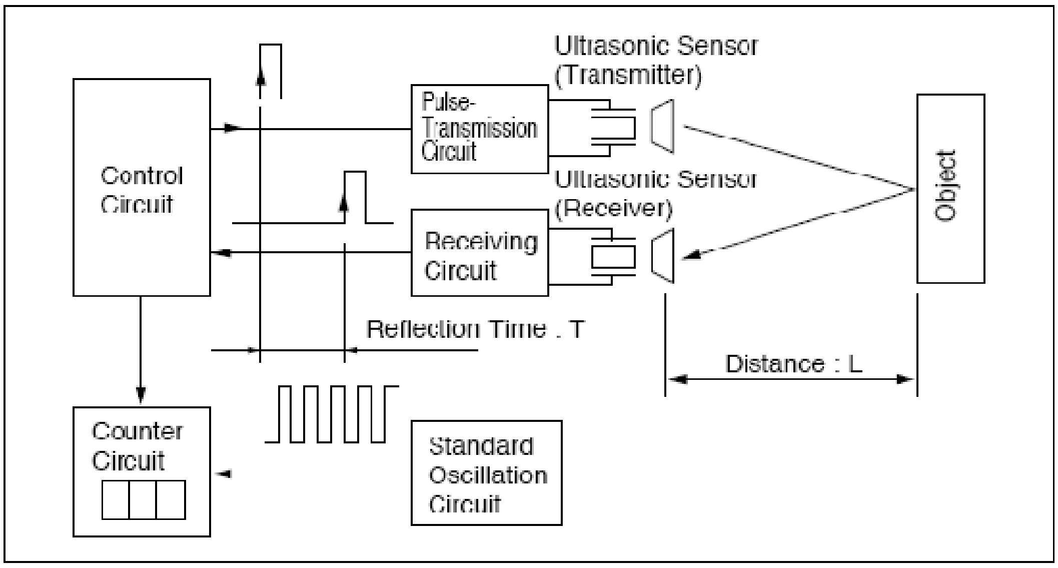

Ultrasonic Sensor (Transmitter)
Pulse-Transmission Circuit
Control Circuit
Ultrasonic Sensor (Receiver)
Object
Receiving Circuit
Reflection Time : T
Distance : L
Counter Circuit
Standard Oscillation Circuit

Figure 3.6 Measuring distance

$$L = C.\frac{T}{2} \tag{3.2}$$

Here, L is distance to object, C is speed of sound and T is total time for echo.

The frequency of sound the sensor uses affects its efficiency. A higher frequency means the wavelength is shorter. This gives a wider angle of detection, but reduces the operation range. The shape of target objects, medium temperature and humidity are also important as using ultrasonic sensors. A typical ultrasonic sensor can be tuned to detect objects at ranges between a few centimeters and 10 meters.

3.2 Inertial Measurement Units

Inertial measurement unit is a cluster of all necessary sensors to be able obtain the state information. Inertial measurement units are classified in two groups according to their operating navigation frame as stable platform and strap-down systems.

3.2.1 Stable Platform Systems (Gimbaled)

In this configuration, the main principle is to keep the inertial sensors aligned with global navigation frame. The stable platform can rotate in three axes with the help of gimbals. The gyroscopes sense the any rotational movement and this is sent to torque motors which rotate gimbals so cancels out any external rotations.

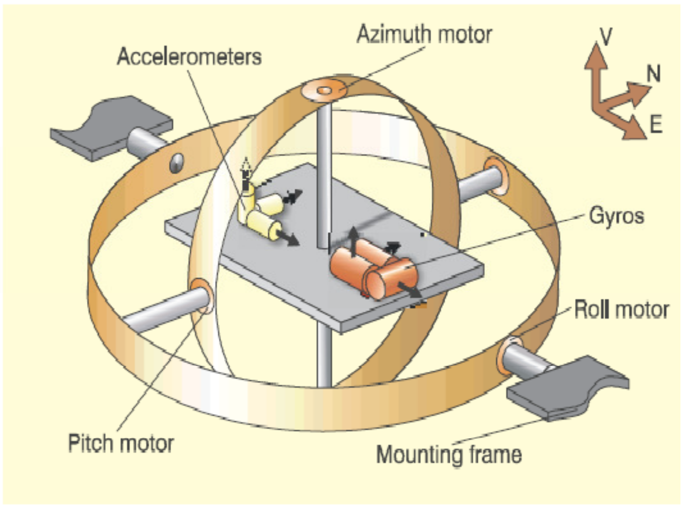


Figure 3.6 Stable platform measurement unit

The angles between gimbals are interpreted as orientation. Accelerometer outputs are referenced with global frame. So the velocity can be calculated directly by integration once after gravity correction. The position is obtained double integration of accelerometer readings. It is noted that accelerometer readings contain the gravitational acceleration.

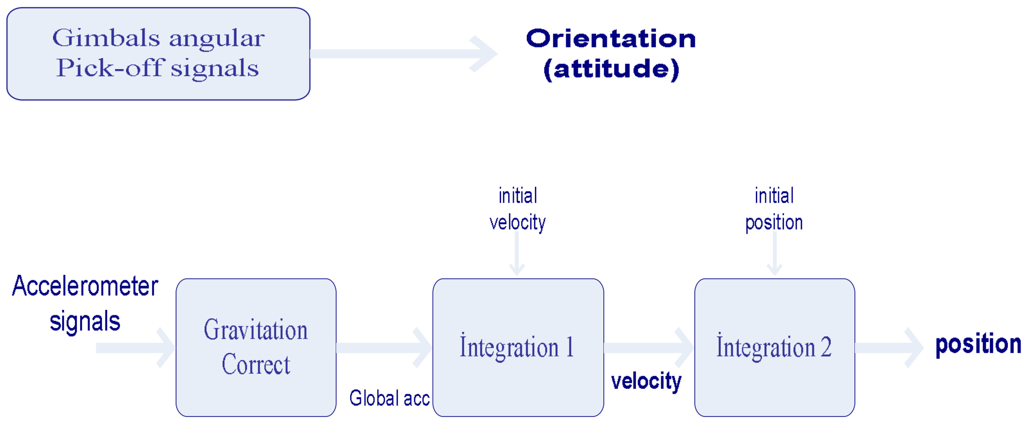


Figure 3.7 Stable platform IMU operations

3.2.2 Strap-down Systems

In strap-down systems, the gyroscopes and accelerometers are placed on the body in a rigid manner. The main difference from stable platform system is that the measurements are referenced with the body frame. The orientation is obtained with just integrating gyroscope readings. The body frame referenced accelerometer readings are transformed to global frame with the help of orientation and integrated twice to get the position.

The strap-down systems are less complex mechanically and smaller than stable platform systems but have computational complexity.

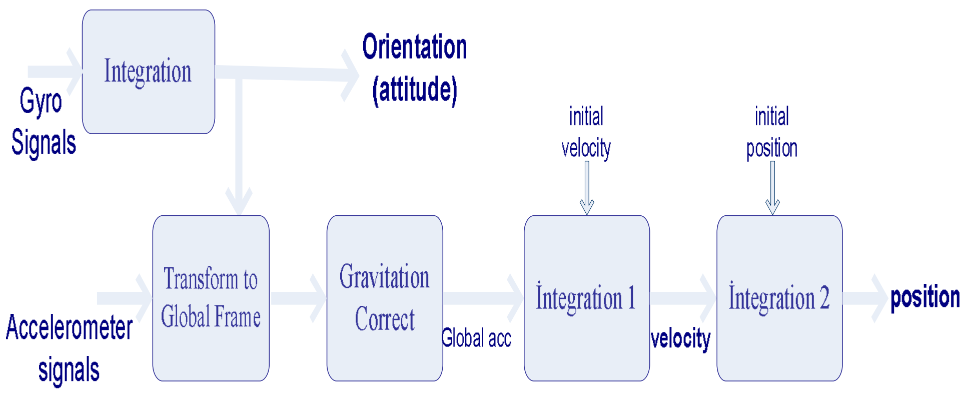


Figure 3.8 Strap-down IMU operations

3.3 Filtering

Raw sensor data is noisy and not usable value. Moreover, each sensor has different noisy characteristics. So sensor raw readings need to be filtered and merged together. There are typically three type of method used for combining sensor data. The first is known as correction. A sensor data is used to correct another sensor data. Second is binding sensor data together. This involves merging different parts of a sensor together and disregarding other parts. Third method is fusion and the best for merging sensor data. In this method, the values from each sensor are combined together in a rated, statistical manner to produce good results. In IMUs, accelerometer which is used to get the position and gyroscope which is used to get attitude of the vehicle plays an important role. Due to the nature of acceleration measurement, the outputs fluctuate not only to changes in the gravitational vector but also to very small accelerations and disturbances like translational acceleration, thrust and altitude changes, wind, etc. The accelerometer reading is very noisy and tends to error. The gyroscopes are used to smooth out these errors. The gyros are, however, not without their own limitations. Although they suffer little from translational noise components due to their rotational measurement system, they tend to suffer heavily from drift and hysteresis. By statistically averaging the values from the sensors together, the attitude of the vehicle can be determined with a decreased noise and error component. Typically, sensor fusion algorithms use a form of Kalman filtering to observe a noisy signal over time in order to produce a signal that is closer to the true value of the measurement. The Kalman filter approach typically uses the time domain principles of the noise with no regard to the signal transfer functions or frequency components. As an alternative, the complementary filter method concerns itself with analysis of the frequency domain with no consideration of the statistical description of the noise signal. Essentially, a complementary filter can be called as a steady-state Kalman filter.

3.3.1 Kalman Filter

Kalman filter is a bunch of mathematical equations that implement a predict-correct type estimator that is optimal in the sense that it minimizes the estimated error under some presumed conditions. The Kalman filter can estimate the variables of a wide range of processes which require the estimate of the states of a linear system. In order to use a Kalman filter to eliminate noise from a signal, the process that measured must be described as a linear system. A linear system is simply a process that can be defined by the following two equations:

$$x_k = A.x_{k-1} + B.u_k + w_{k-1} \tag{3.3}$$

$$z_k = H.x_k + v_k \tag{3.4}$$

In the equations A, B, and H are matrices; k is the time index; x_k is called the state of the system; U is a known input to the system; z_k is the measured output; and w and v are the noise. The variable w is called the process noise, and v is called the measurement noise. Each of these variables is a vector and therefore contains more than one element. The vector x_k contains all of the information about the current state of the system, but $x_k$vector cannot be measured directly. Instead, z_k which is a function of x_k can be measured but z_k is affected by the measurement noise v_k.

The system behavior is known according to the state equation and also measurements of the position, so the best estimate of the state x_k is should be determined. An estimator that gives an accurate estimate of the true state even though that cannot be directly measurable, should be determined. There are two requirements for the estimator; first, the expected value of the estimate should be equal to the expected value of the state, second, the estimator should have the smallest possible error variance.

The Kalman filter satisfies the two requirements under some assumptions about noise. It is assumed that the process and measurement noises have the average value of zero and also assumed that there is no correlation between them.

$$Q = E\,(w_k \,.\, w_k^T) \tag{3.5}$$

$$R = E\,(z_k \,.\, z_k^T) \tag{3.6}$$

The filter contains four main operations; Initialization, Prediction, Observation and Estimation.

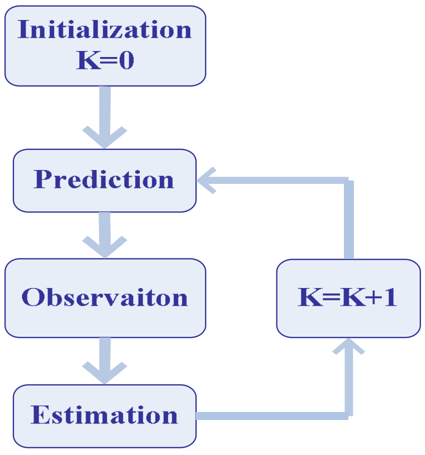


Figure 3.9 Kalman filter operations.

In the initialization step, initial state variables are set. In an inertial navigation system, initial position, velocity and acceleration values are determined.

$$x_k^{\sim} = A.\,x_{k-1} + B.\,u_k \tag{3.7}$$

$$P_k^{\sim} = A.\,P_{k-1} + A^T + Q \tag{3.8}$$

The observation matrix is measured by inertial sensors. The difference between the predicted sensor values $H.\,x_k^{\sim}$ and real sensor values are called as innovation.

$$z_k - H.\,x_k^{\sim} \tag{3.9}$$

The observation is processed in every period and sensor one by one for each. The Kalman gain is calculated for every time step after observation.

$$K_k = P_k^{\sim} . H^T . (HP_k^{\sim} . H^T + R)^{-1} \qquad (3.10)$$

And the new state estimation for new Kalman Gain;

$$x_k = x_{k-1}^{\sim} + K_k (z_k - H.x_k^{\sim}) \qquad (3.11)$$

The first term used to derive the state estimate at time k is just predicted state x variable at k-1 time. This would be the state estimate if a measurement is not existed. In other words, the state estimate would propagate in time just like the state vector in the system model. The second term in the equation is called the correction term and it represents the amount by which to correct the propagated state estimate due to our measurement. The covariance matrix for new state is expressed as;

$$P_k = P_k^{\sim} (1 - K_k H) \qquad (3.12)$$

The first job during the measurement update is to calculate the Kalman gain. The next step is to actually measure the process to obtain and then to generate the next state estimate. The final step is to obtain the next error covariance. After each time and measurement update pair, the process is repeated with the previous 'next estimates' used to predict the new a former estimates.

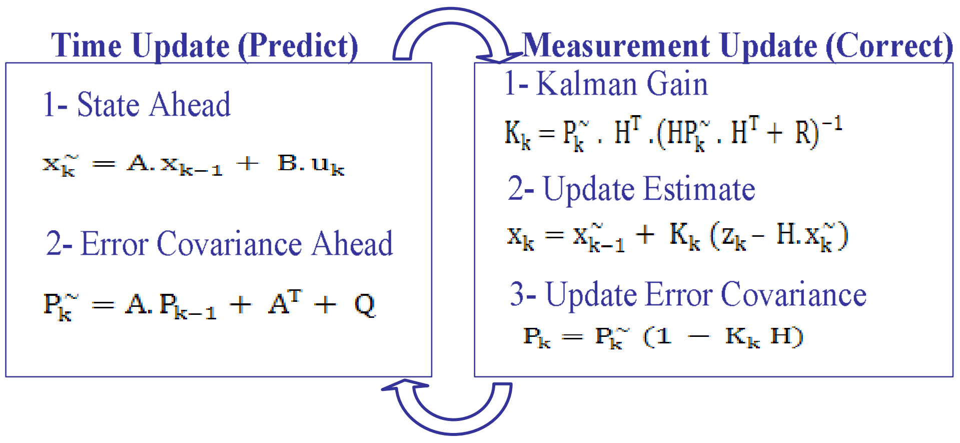

Time Update (Predict)
1- State Ahead
$x_k^{\sim} = A.x_{k-1} + B.u_k$
2- Error Covariance Ahead
$P_k^{\sim} = A.P_{k-1} + A^T + Q$
Measurement Update (Correct)
1- Kalman Gain
$K_k = P_k^{\sim} . H^T .(HP_k^{\sim} . H^T + R)^{-1}$
2- Update Estimate
$x_k = x_{k-1}^{\sim} + K_k (z_k - H.x_k^{\sim})$
3- Update Error Covariance
$P_k = P_k^{\sim} (1 - K_k H)$

Figure 3.10 Kalman filter loop

CHAPTER FOUR
CONTROL SYSTEM

Applications that require control of the system output when reference input value changed, need to a control algorithm. A control system is commonly used in applications like motor control, control of temperature, pressure, flow rate, speed, force or other variables.

Different control algorithms have been used up to now, but the PID algorithm has become common method because of its simplicity and good performance. The PID algorithm can be applied to control any measurable variable, if this variable can be controlled by adjusting some other process variables.

4.1 PID Controller

In Figure 4.1 a diagram of a system with a PID controller is shown. The PID controller compares the measured sensor value of y with a reference value (y0). The difference is called as error (e) which is evaluated to compute a new process input (u) to system. This calculated input will try to approach the measured sensor value to the desired reference value. The alternative method to a closed loop control such as the PID controller is an open loop controller. Open loop control which has no feedback is in many cases not adequate, and is generally not preferred. By adding feedback from the system output, performance can be improved greatly.

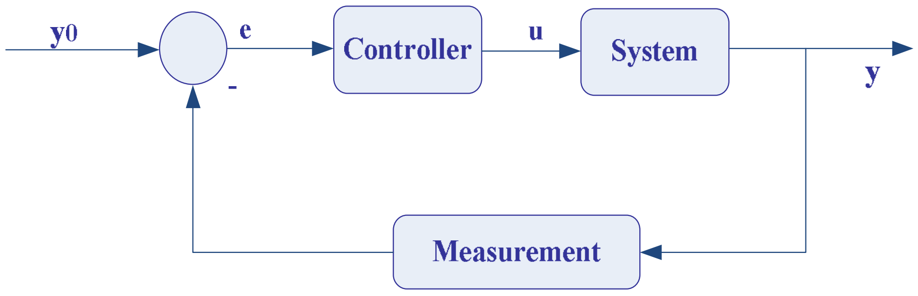


Figure 4.1 Simple PID block diagram

On contrary to simple control algorithms, the PID controller is to be able to adjust the process inputs according to the old and rate of change of the signal. This provides a more accurate and stable control design. Firstly, the system output state is measured by sensors. And then this value is subtracted from desired reference to generate the error value. The error will be evaluated in three ways, to manage the present with the proportional term, recover from the past using the integral term and to predict the future with the derivative term.

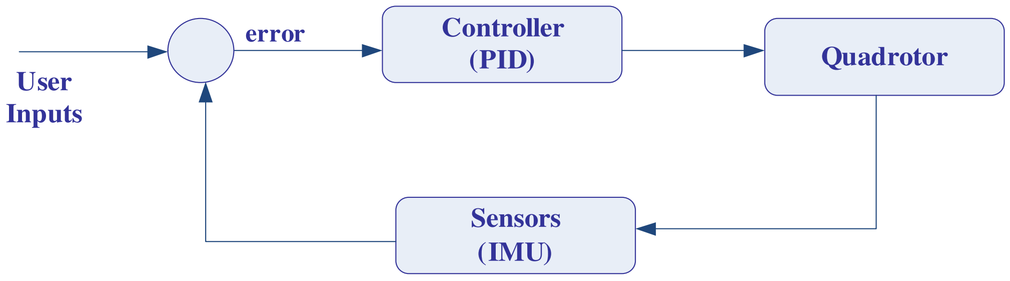


Figure 4.2 Control system for quadrotor

4.1.1 Proportional Term

The proportional term provides a system control input proportional with the error. if just a proportional control is used, there exists a stationary error except when the reference equals to measured state.

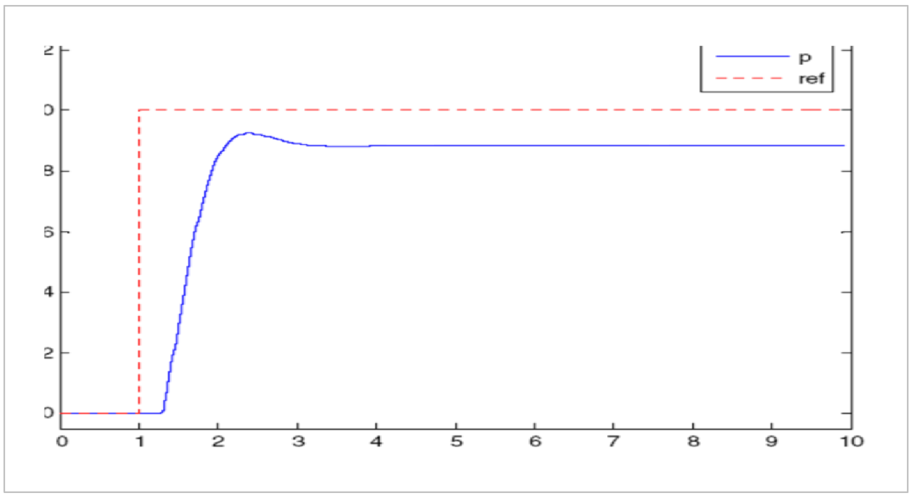


Figure 4.3 Step response of P term.

In figure 4.3, the stationary error in the system process value appears after a change in the desired value. Using a too large P term gives an unstable system.

4.1.2 Integral Term

The integral term provides an addition from the sum of the old errors to the system control input. The summation will continue until the measured system process equals the desired reference value and this provides no stationary error when the reference is stable. Normally integral term is not used alone because of slow response and oscillatory output. The most common use of I term is together with the P term, called a PI controller. The figure 4.4 shows the step response to I and PI controller. As seen the PI controller response have no stationary error and I controller response is very slow.

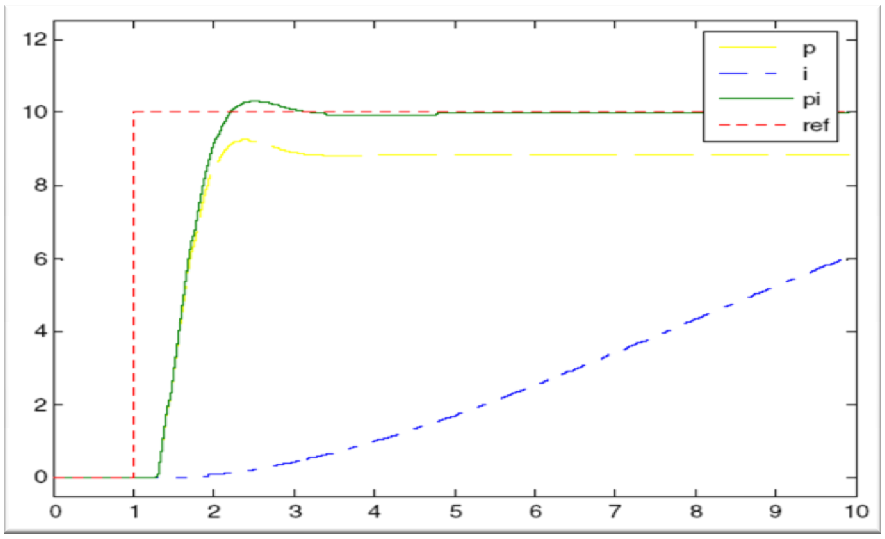


Figure 4.4 Step response of I term

4.1.3 Derivative Term

The derivative term provides an addition from the rate of change in the error to the system control input. An instant change in the error will give an addition to the system control input. This improves the response to a sudden change in the system state or reference value. The D term is typically used with the P or PI as a PD or PID controller. A large D term usually gives an unstable system. The figure 4.5 shows D and PD controller responses. The response of the PD controller gives a faster rising system process value than the P controller.

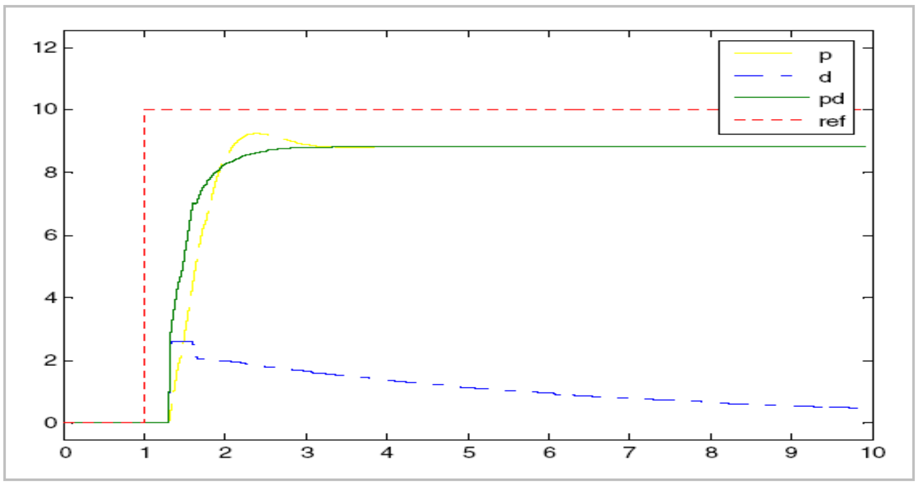


Figure 4.5 Derivative Term

Using all the terms together, as a PID controller usually gives the best performance. The figure 4.6 shows the P, PI, and PID controllers. PI improves the P by removing the stationary error and the PID improves the PI by faster response and no overshoot.

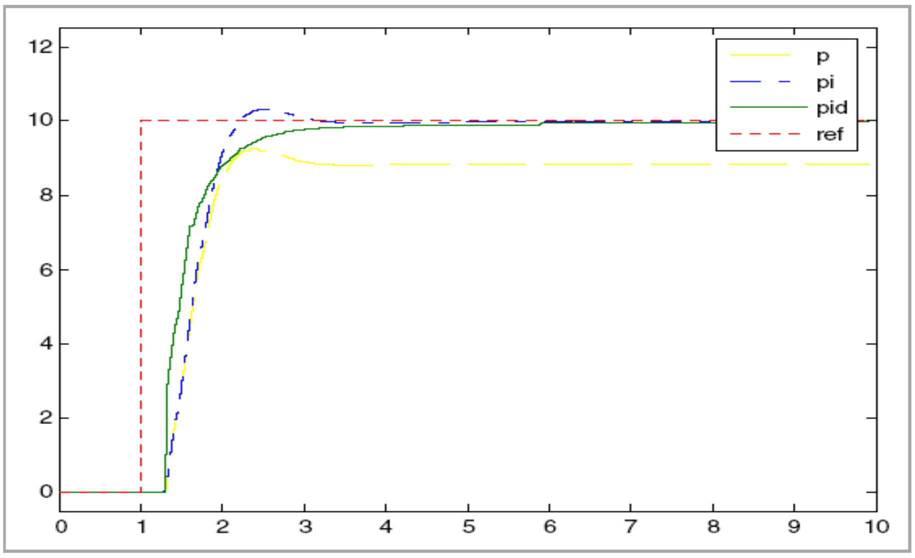


Figure 4.6 PID controller step response

4.1.4 Discrete PID Control Algorithm

The transfer function of the system in S-domain

$$\frac{u}{e}(s) = H(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \qquad (4.1)$$

Here Tp, Ti, and Td denote the time constants of the proportional, integral, and derivative terms respectively. In time domain version of the transfer function;

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\sigma) d\sigma + T_d \frac{de(t)}{dt} \right) \tag{4.2}$$

Approximating the integral and the derivative terms to get the discrete form;

$$\int_0^t e(\sigma) d\sigma = T \sum_{k=0}^{n} e(k) \frac{de(t)}{dt} \approx \frac{e(n)-e(n-1)}{T} \qquad t = nT \tag{4.3}$$

Where n is the discrete step at time t. Finally Discrete PID equation;

$$u(n) = K_p\, e(n) + K_i \sum_{k=0}^{n} e(k) + K_d (e(n) - e(n-1)) \tag{4.4}$$

4.2 Controller Tuning

After the PID controller successfully implemented, the next step is to tune the controller gains to appropriate values. There are several tuning methods in order to find the P, I and D gains appropriately to provide a desired response. These are manual tuning, Ziegler-Nichols, software tuning and Cohen-Coon tuning. In the actual implementation of the platform, disturbances rejection is much more important than getting the reference value exactly every time. The Ziegler-Nichols tuning method is good at disturbance rejection but a little slow to follow the reference input.

Ziegler-Nichols method is quite simple. First, the proportional constant is set a small value. As Kp is increased slowly, the response of controlled variable is observed. The Kp is increased until the response becomes oscillatory continuously. This is known as the ultimate gain (Ku) and the period of the oscillation as Pu. The control law settings are then obtained from the following table.

4.3 Model Simplifications

In model based control, it is common to take the full simulation model and simplify it such that the control is applied to a specific behavior. This makes the initial design of the system less intensive and allows adjusting the system for the most significant effects before adding minor effects that may increase the complexity of the controller. Equations that show the rotational equations of motion for the full simulation model had been developed in Chapter 2.

Table 4.1 PID tuning table

	K_P	K_I	K_D
P	$\frac{K_U}{2}$		
PI	$\frac{K_U}{2.2}$	$\frac{P_U}{1.2}$	
PID	$\frac{K_U}{1.7}$	$\frac{P_U}{2}$	$\frac{P_U}{8}$

In full simulation model, the rotor gyroscopic effects created by the rotational motion of the propellers are not important comparing to the moments created by the rotor thrusts. For this reason, they are eliminated in both the simulation and the controller design. To limit the design over which the controller is valid thus reducing the complexity of the controller while evaluating the most important characteristics of the vehicle. When gyroscopic effects are eliminated;

$$\ddot{\phi} = \sum \frac{\tau_x}{I_{xx}} = \left[\dot{\theta}\dot{\psi}\left(I_{yy} - I_{zz}\right) + r(-T_2 + T_4)\right]\left(\frac{1}{I_{xx}}\right) \tag{4.8}$$

$$\ddot{\psi} = \sum \frac{\tau_z}{I_{zz}} = \left[\dot{\theta}\dot{\phi}\left(I_{xx} - I_{yy}\right) + (T_1 + T_3 - T_2 - T_4)\right]\left(\frac{1}{I_{zz}}\right) \tag{4.9}$$

$$\ddot{\theta} = \sum \frac{\tau_y}{I_{yy}} = \left[\dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) + r(T_1 - T_3)\right]\left(\frac{1}{I_{yy}}\right) \tag{4.10}$$

A stable hover state will be the desired orientation of the vehicle. This means we are only concerned with the rotations of the vehicle near hover. As a result, we will only consider the rotational movements for the roll, pitch and yaw angles. The angular velocities for roll, pitch, and yaw will be very small. For this reason, we can also omit the angle gyroscopic effects introduced in Chapter 2. These eliminations form the simplified control model used to develop the controller. These simplified equations of motion are provided in equations 4.11 - 4.13.

$$\ddot{\phi} = \sum \frac{\tau_x}{I_{xx}} = [\, r(-T_2 + T_4)]\left(\frac{1}{I_{xx}}\right) \tag{4.11}$$

$$\ddot{\theta} = \sum \frac{\tau_y}{I_{yy}} = [\, r(T_1 - T_3)]\left(\frac{1}{I_{yy}}\right) \tag{4.12}$$

$$\ddot{\psi} = \sum\frac{\tau_z}{I_{zz}} = [(T_1 + T_3 - T_2 - T_4)](\frac{1}{I_{zz}}) \quad (4.13)$$

4.4 Input Declarations

After the simplified model is defined, the next step is to define the input (U)vector that will be used to control the system dynamics from the controller. In the simulation, the input to the system dynamics model was based on the relationship between the pulse-width modulation command send from the controller to the motors and the actual thrust output. This relationship is only linear for a small portion of the thrust curve. It is therefore useful to use a more observable rotational speed instead of PWM input. The thrust produced by motors related to the square of the propeller speed when the flight state is in hover and is not a motion, with a thrust constant factor and drag moment factor considered. Therefore, the inputs selected for the system can be expressed as in equations 4.14 – 4.16. The altitude input is comprised of all the motor inputs as a sum to provide the altitude of the vehicle.

$$U_{roll} = rb(-\Omega_2^2 + \Omega_4^2) \quad (4.14)$$

$$U_{pitch} = rb(\Omega_1^2 - \Omega_3^2) \quad (4.15)$$

$$U_{yaw} = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \quad (4.16)$$

CHAPTER FIVE
SIMULATOR DESIGN

The overall software design can be separated to three parts as simulator, UAV and graphical user interface (GUI). User sends commands with GUI, UAV software takes them and uses as reference values for the control system. Control system evaluates reference with current state and calculates actuator states to drive all system into user demands. Simulator model applies actuator states and outputs sensor values to UAV software. This loop continues forever.

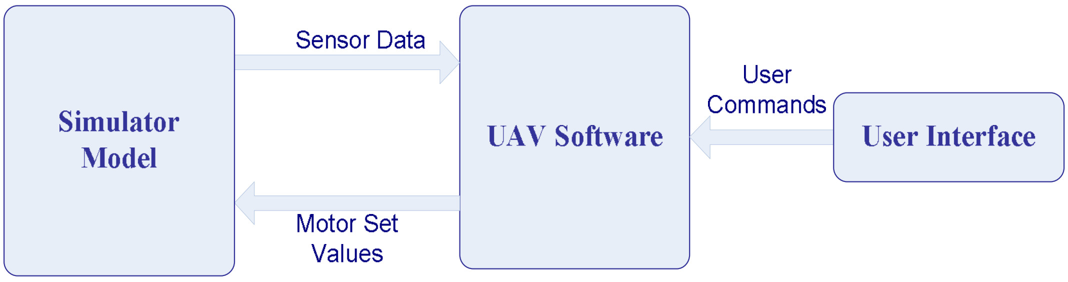


Figure 5.1 Diagram of working principle of overall system

The simulator contains graphic model, dynamic design and MCU modeling. Simulator is separated from UAV software with an abstraction layer. UAV software consists of main UAV middleware core and application layer. User interface is totally different software that consists of common controls for user to send commands to UAV system.

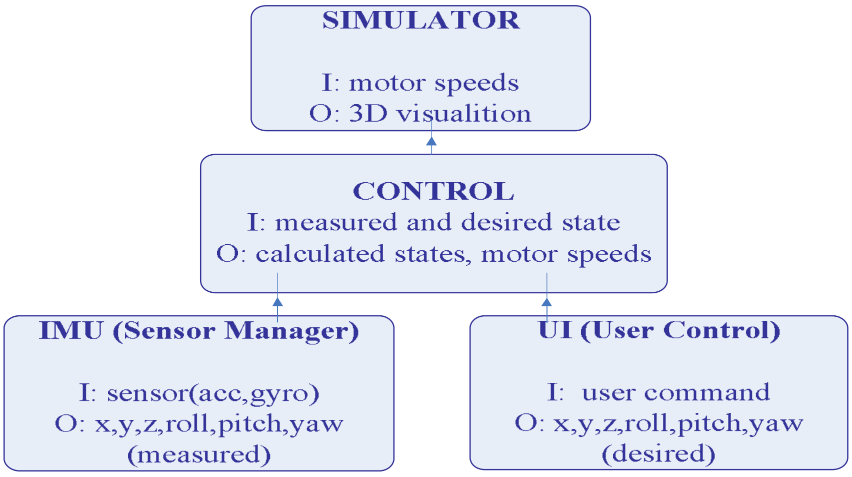


Figure 5.2 Overall system input-ouput model

5.1 Software Layers

Quadrotor overall software is divided to layers in hierarchic manner. Layers also divided to sub-blocks as modules. Below Figure 5.3 illustrates common software layers and blocks. The layered configuration provides abstraction between software tasks. All work is broken down to parts so that the design is simplified. The main advantage of layered design is that the code reusability is increased. The layer and modules can be used other software projects or can be discarded easily.

The quadrotor software consists of layers as application, middleware, driver abstraction and simulator. With driver abstraction layer, the same middleware and application layer can be used for both simulator and real UAV platform.

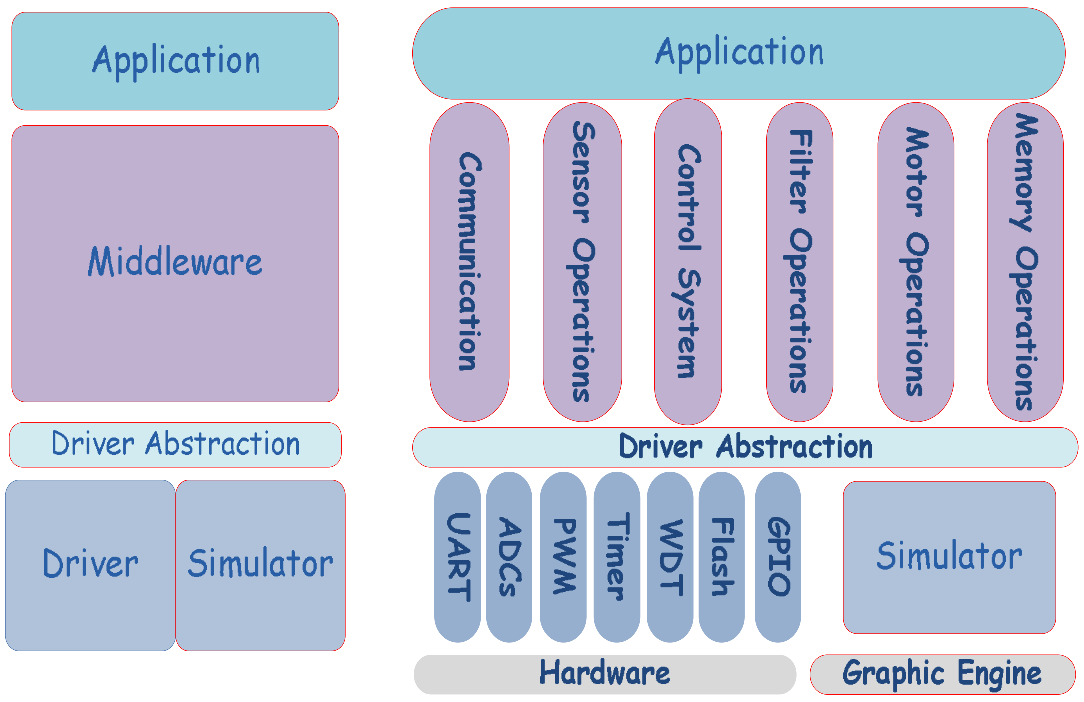


Figure 5.3 General software layer diagram

5.1.1 Simulator Layer

Simulator layer provides visual output of the project such that quadrotor 3D model movements and medium it located on. This layer also includes sensor modeling, dynamic calculations and MCU modeling.

3D model operations are fulfilled using OpenGL\GLUT library. Firstly, the model and terrain are loaded. After initialization system goes into idle state and waits for user command.

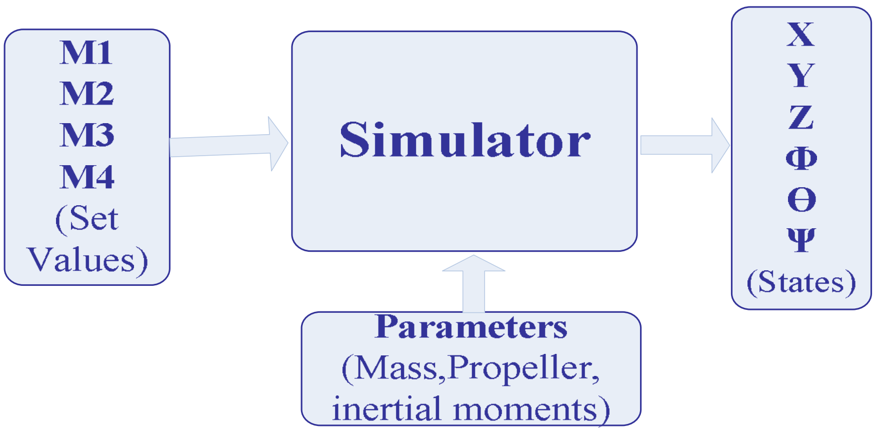


Figure 5.4 Simulator software input output model

All operation done in simulator layer is shown in Figure 5.5. Simulator system just takes motor set values that calculated in control system and outputs sensor values calculated in sensor modeling operation. The rotor speeds are calculated from motor set values using brushless motor model developed. The rotor thrust values are calculated using propeller model which uses rotor speeds and propeller's physical parameters. After that, the quadrotor dynamic equations are processed to obtain the attitude of the vehicle which consists of six degree of freedom x, y, z positions and roll, pitch, yaw angles on the earth frame. Calculated system states applied to quadrotor 3D model instantly. Finally, sensor outputs are calculated with sensor modeling operation that will be explained in next sections.

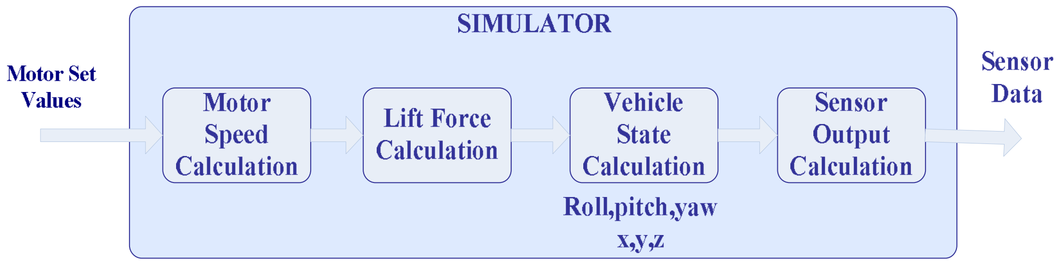


Figure 5.5 Simulator software operations

5.1.2 Driver Abstraction Layer

The driver abstraction layer isolates the platform based operations from main quadrotor operations. Thus it is used to pass the code into simulator instead real quadrotor platform.

5.1.3 Middleware Layer

Middleware layer is designed to fulfill the generic UAV operations. All the necessary modules of UAV system needs should be implemented in this layer. A communication system between operator and vehicle, sensor measurement and filtering to obtain the vehicle instant attitude and a powerful control system for managing vehicle movement are common in all UAV platforms. The middleware software modules are shown in Figure 5.6 below.

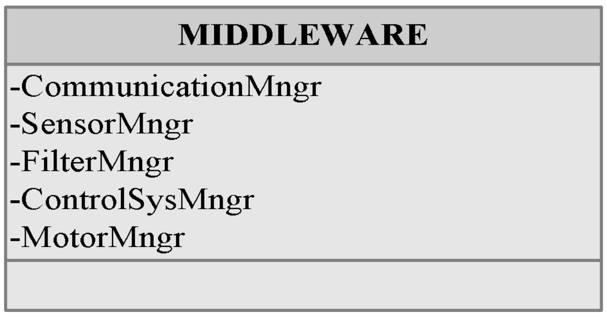


Figure 5.6 Middleware layer

5.1.3.1 Communication Operations

The communication system creates connection between operator and vehicle. The operator sends commands using user interface and vehicle receives them and processes. This module mainly captures the packets sent from user interface and sends packets to user interface. The captured packets are parsed and interpreted as user desired states for the control system. Additionally, it shares quadrotor states, sensor values with GUI. The communication module functions and variables are shown in Figure 5.7 below.

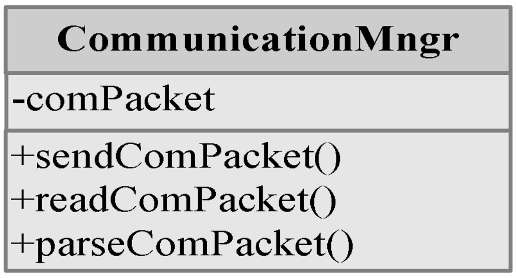
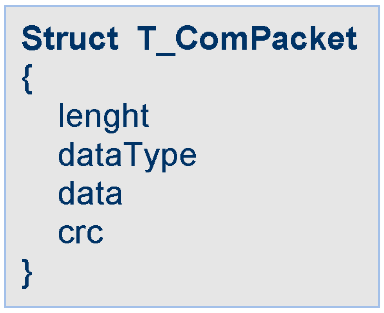


Figure 5.7 Communication module elements

The variable 'comPacket' stores instant communication packet in the form of 'T_ComPacket' struct. The function 'sendComPacket()' sends packets to channel. The function 'readComPacket()' captures packets from channel and 'parseComPacket()' function parses packets captured. 'T_ComPacket' data structure consists of communication protocol packet content as packet length, type (command or data), data and 16-bit crc for data integrity check. The structure is shown in Figure 5.8 below.



```
Struct  T_ComPacket
{
   lenght
   dataType
   data
   crc
}
```

Figure 5.8 package structure

Serial communication (RS232) communication is used between simulator and user interface. Simulator uses Windows WIN32 API library for serial communication.

5.1.3.2 User Command Operations

User commands are interpreted after parsing communication packets and user desired states are determined. All UAV systems have generic user commands as take off, rotate to all directions, hover and landing. All these commands are implemented in simulator and user interface side to control quadrotor.

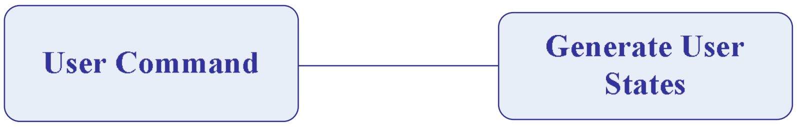


Figure 5.9 User command operations

Module functions are listed in Figure 5.10 below.

UserControlMngr
-userState

+userCmdTakeOff()
+userCmdTurnRight()
+userCmdTurnLeft()
+userCmdLanding()
+userCmdHover()
+interpretUserCommand()

Figure 5.10 User command

The module variable 'userStates' consist of user desired states in the form of 'T_UserState' structure. The commands are interpreted and desired roll, pitch, yaw and altitude values calculated for control system reference input. 'T_UserState' structure is used for control system as reference or desired value. It consists of user's desired roll, pitch, yaw and altitude values.

```
Struct T_UserState
{
  userROLL
  userPITCH
  userIYAW
  userALTITUDE
}
```

Figure 5.11 User state structure

5.1.3.3 Sensor Operations

Sensor operation manager gets raw sensor data as voltages from ADC unit and convert them to actual readings. The filter manager applies filters to get reliable data and produces measured sensor states for control system as current device attitude. Here moving average and Kalman filters are processed.

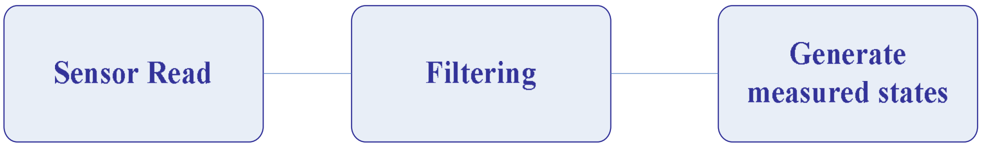


Figure 5.12 Sensor operations

Module functions are listed in Figure 5.13

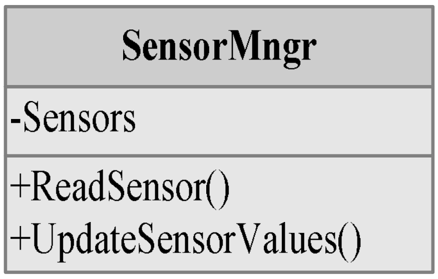
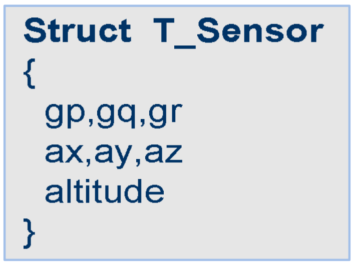


Figure 5.13 Sensor processes

The module variable ‘Sensors’ stores all sensor raw data in the form of ‘T_Sensor’ structure. The function ‘ReadSensor()’ reads raw sensor data and ‘UpdateSensorValues()’ function updates all sensor readings. ‘T_Sensor’ structure consists of elements for all necessary sensor types. The three axis gyro, three axis accelerometer and altitude sensor raw readings are stored for filter operations.



```
Struct  T_Sensor
{
  gp,gq,gr
  ax,ay,az
  altitude
}
```

Figure 5.14 Sensor data

5.1.3.4 Filter Operations

Sensor raw readings contain lots of noise and should be filtered. Two types of filtering are applied as moving average and Kalman filter. Filter module functions are listed in Figure 5.15 below.

```
FilterMngr
-SensorState
+kalmanInit()
+movingAverageFilter()
+ProcessFilter()
```

Figure 5.15 Filter process functions

The function ‘kalmanInit()’ initializes Kalman states. The functions ‘movingAverageFilter()’ and ‘ProcessFilter()’ apply Kalman and moving average filters respectively. ‘T_SensorState’ data structure is used in control system as an actual system state. Firstly, sensor outputs are filtered and interpreted as roll, pitch, yaw and altitude of measurement system.

```
Struct  T_SensorState
{
  sensorROLL
  sensorPITCH
  sensorYAW
  sensorALTITUDE
}
```

Figure 5.16 Filtered sensor data

5.1.3.5 Control System

Control system takes desired user states as reference and filtered sensor states as input, processes PID control algorithm and generates output states.

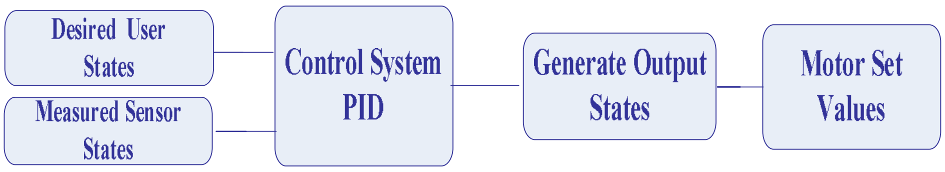
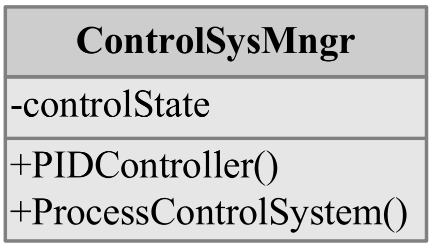


Figure 5.17 Control system operations

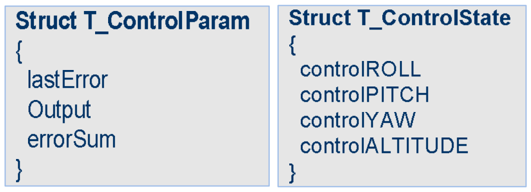
Module functions are listed in Figure 5.18 below.



```
ControlSysMngr
-controlState
+PIDController()
+ProcessControlSystem()
```

Figure 5.18 Control system functions

The module variable 'controlState' stores the output of PID controller. The 'PIDController()' function applies PID operations and 'ProcessControlSytem()' function calls it for all states. The control system outputs are used for actuator system as an input. Here, control roll, pitch, yaw and altitude values are calculated.



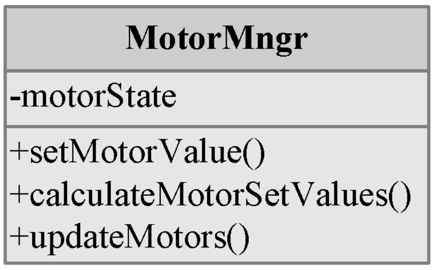
```
Struct T_ControlParam
{
  lastError
  Output
  errorSum
}
```

```
Struct T_ControlState
{
  controlROLL
  controlPITCH
  controlYAW
  controlALTITUDE
}
```

Figure 5.19 Control system data structures

5.1.3.6 Motor Operations

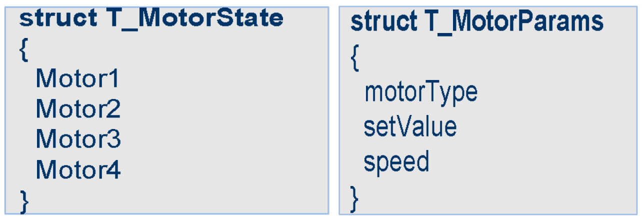
Motor software module takes control system outputs and calculates motor set values. Finally, it sends new calculated values to PWM module. Module functions are listed Figure 5.20 below.



```
MotorMngr
-motorState
+setMotorValue()
+calculateMotorSetValues()
+updateMotors()
```

Figure 5.20 Motor process functions

The module variable 'motorState' stores motor instant parameter as motor location, speed and set values for each motor.



```
struct T_MotorState
{
  Motor1
  Motor2
  Motor3
  Motor4
}
```

```
struct T_MotorParams
{
  motorType
  setValue
  speed
}
```

Figure 5.21 Motor system data types

5.1.4 Application Layer

Using generic UAV middleware functions, quadrotor application is created. Application layer simply calls middleware functions to operate the quadrotor in desired manner.

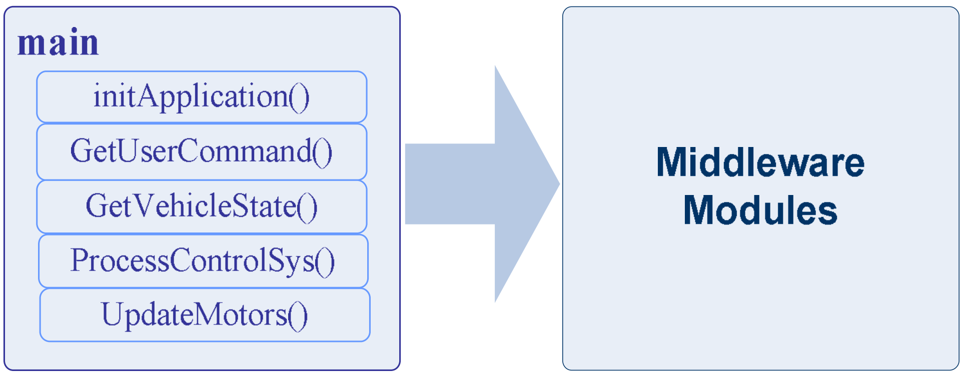


Figure 5.22 Application layer

Firstly, The function ‘InitApplication()’ initiates middleware modules. The function ‘GetUserCommand()’ takes incoming user commands if available. The function ‘GetVehicleState()’ gets instant vehicle states. Here, sensor read and filtering operations are processed. ‘ProcessControlSys()’ processes control system algorithms to drive overall system into operator demands. Finally, actuator system is updated.

5.2 Simulator 3D Visualization

The quadrotor3D model is created with Blender 3D tool as wavefront object format. The model imported to Opengl medium using Opengl mathematics (GLM) library. Basically, it reads a wavefront object file and material file and knows how to draw it on the screen exactly the way it looked in Blender. It provides also texturing and locating the model on the screen everywhere.

After importing quadrotor model, it can be moved anywhere on the screen and rotated with any angle representing roll, pitch and yaw movements using OpenGL functions. Opengl part takes the model states that roll, pitch, yaw, x, y, z in earth frame and positions it on the screen. These states are calculated from motor speeds. The motor speeds are calculated from motor set values which are the control system's output.

5.2.1 OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games.

OpenGL hides complexities of interfacing with different 3D accelerators by providing a simple, uniform interface. In its basic operation, OpenGL accepts primitives such as points, lines and polygons, and converts them into pixels via a graphics pipeline known as the OpenGL state machine. Most OpenGL commands either issue primitives to the graphics pipeline, or configure how the pipeline processes these primitives.

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. This contrasts with descriptive APIs, where a programmer only needs to describe a scene and can let the library manage the details of rendering it. OpenGL's low-level design requires programmers to have a good knowledge of the graphics hardware, but also gives a certain amount of freedom to implement algorithms.

OpenGL is a hardware and system independent interface. An OpenGL application will work on every platform, as long as there is an installed implementation. But, for a good performance, the computer that simulator runs on should have a minimum system requirements as 2GB RAM memory, 2GHz processor speed and 256MB display card memory.

5.2.2 3D Design of Model

Blender 3D design tool is used for drawing quadrotor model. Four arms, one central hub, four motors and propellers are drawn separately and brought together. The model saved as wavefront object format which uses two file as object (.obj) and material file (.mtl). Object file stores description of the surface of a 3D object, composed of triangles or higher degree polygons and material file is an auxiliary file containing definitions of materials like color and textures.

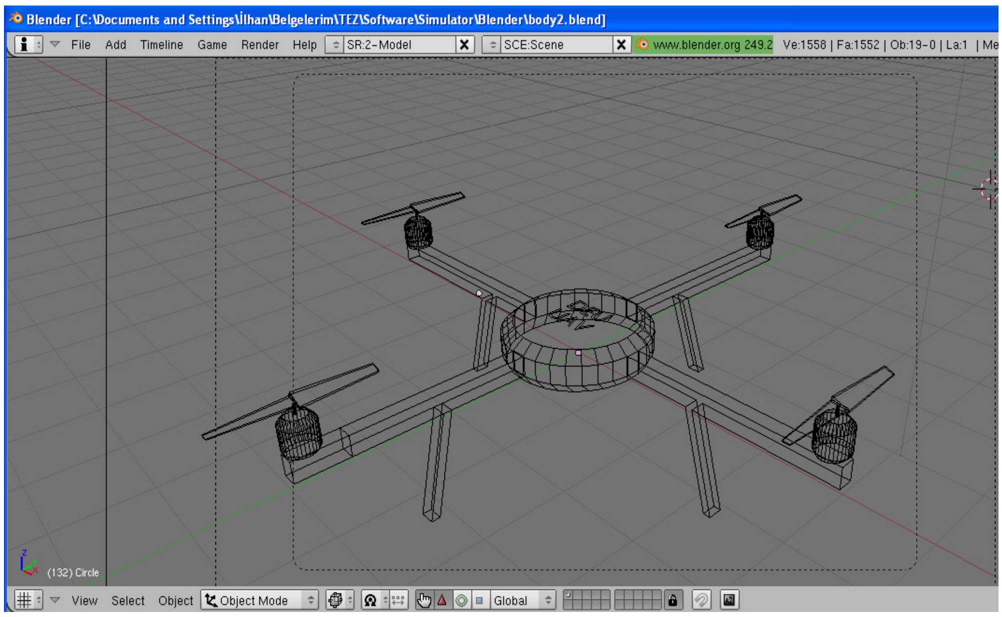

Blender [C:\Documents and Settings\İlhan\Belgelerim\TEZ\Software\Simulator\Blender\body2.blend]
File Add Timeline Game Render Help
SR:2-Model
SCE:Scene
www.blender.org 249.2
Ve:1558 | Fa:1552 | Ob:19-0 | La:1 | Me
(132) Circle
View Select Object
Object Mode
Global

Figure 5.23 3D Simulator model as skeleton.

When working with Blender tool, first 3D model is drawn as skeleton as shown in Figure 5.23. The central hub, arms, motors and legs are created as one object and propeller created as different object because Opengl rotates propellers besides overall quadrotor. Then it is coated with desired colors as shown Figure 5.24. One of the arms is marked as red near motor to represent the front of vehicle.

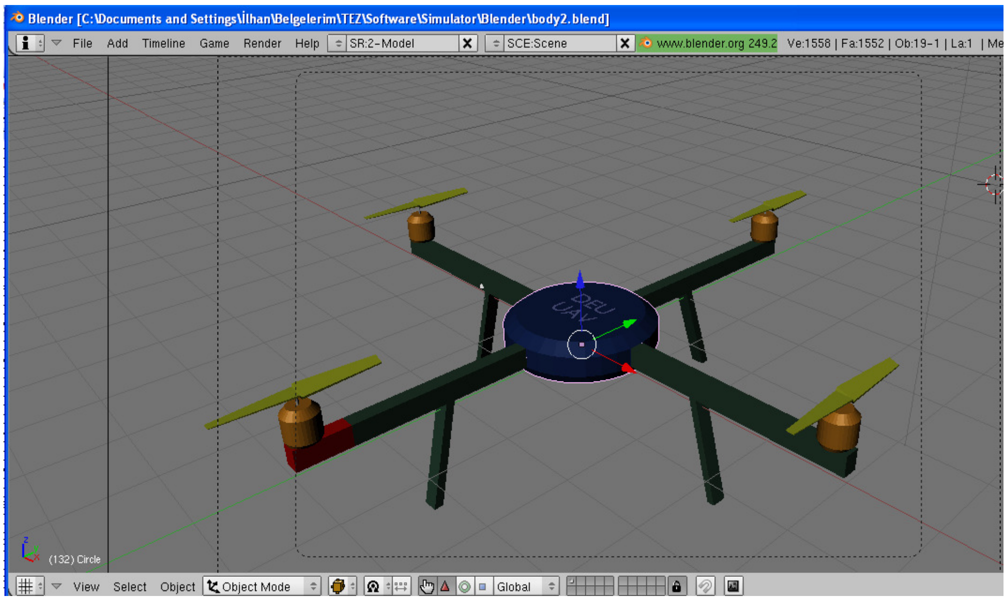

Blender [C:\Documents and Settings\İlhan\Belgelerim\TEZ\Software\Simulator\Blender\body2.blend]
File Add Timeline Game Render Help
SR:2-Model
SCE:Scene
www.blender.org 249.2
Ve:1558 | Fa:1552 | Ob:19-1 | La:1 | Me
DEU UAV
(132) Circle
View Select Object
Object Mode
Global

Figure 5.24 3D Simulator model.

And finally it is previewed as below.

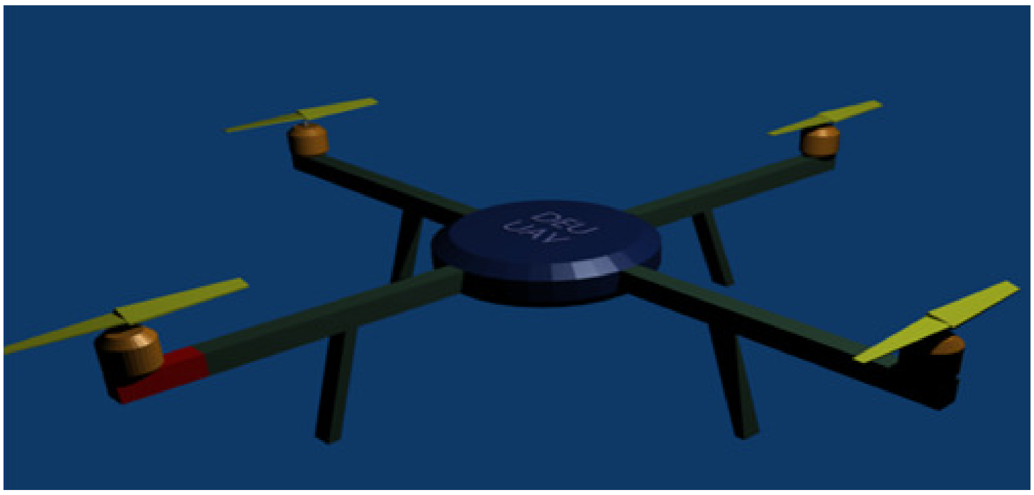


Figure 5.25 3D Simulator model final preview

After completion of the model, it should be exported in an appropriate format. Blender provides lots of format. Wavefront object format is used for the simulator.

Terrains helps to figure out the 3D mediums especially when there exist moving objects. In simulator medium, floor is covered with full green grasses, air with real sky and mountains are placed far behind to increase the visual perception.

5.3 Sensor Modeling

The simulator model just takes motor set values and outputs raw sensor outputs as voltages. Sensors are modeled according to the simulator‘s instant attitude. Simulated IMU consist of 3 axes gyroscope, 3 axes accelerometer and altimeter. This sensor combination allows the platform to track accelerations along the three principal axes of translation (x, y and z) via the accelerometer and rotational speed around two of the principal axes of rotation (in this case, roll and pitch) to give us a total of 6 DOF available.

5.3.1 Accelerometer Model

For each 3 three axis, accelerometer outputs are modeled from simulator dynamic accelerations. ADXL335 accelerometer is used for modeling as base. The output of modeled accelerometer is between 0 to 5 volts. The model equation is shown below.

Output =Offset + Sensivity * sin(Acceleration) + Noise (5.1)

The Offset is measured voltage when no acceleration exist. The sensivity is g value for each volt. The noise is modeled as white noise. White noise is a random signal with a flat power spectral density. In other words, the signal contains equal power within a fixed bandwidth at any center frequency.

5.3.2 Gyroscope Model

The simulator rotational movements are modeled as three axes gyroscopes. IDG500 type gyroscope is used for modeling. The model equation is shown below.

Output = Gyro Zero Bias + Sensivity * Rate + Noise (5.2)

Here, zero bias value comes from no rotation gyro readings. The sensivity is measure of degree for each voltage. The noise model is Gaussian white noise.

5.3.3 Altitude Sensor Model

Ultrasonic sensor is chosen for altitude measurement. Murata MA40S4R/S model ultrasonic sensor is taken as base. The sensor is modeled with following equation.

$$\text{Output} = \text{Offset} + \text{Sensivity} * \text{Altitude} \tag{5.3}$$

5.4 Component Modeling

Quadrotor main components are also modeled in simulation as brushless motor, propellers and microcontroller (MCU). The MCU model includes all peripherals needed in simulation.

5.4.1 Brushless Motor Model

Brushless motors provide higher payload rates with high energy density lithium polymer batteries. Brushless motors replace the field windings with permanent magnets located on the rotor and move the armature windings to the stator. In this manner, the need for brushed mechanical commutation is eliminated, reducing noise and electromagnetic interference due to arcing. Without the friction of the brushes against the rotor, there is also some gain in efficiency. In addition, maintenance is greatly reduced as there are no brushes to replace. On the other hand, a disadvantage of the brushless motor is the need to perform the commutation electronically in a separate unit.

A complicating factor with the use of brushless motors is the introduction of an electronic speed controller (ESC), a device external to the motor which electronically performs the commutation achieved mechanically in brushed motors. The ESC converts the battery pack DC voltage to a three phase alternating signal which is synchronized to the rotation of the rotor and applied to the armature windings. The motor speed is then proportional to the root-mean-square (RMS) value of the armature voltage and is set by the ESC in response to a pulse width modulated control signal. The relationship between the control signal and the voltage level is not necessarily linear and must be confirmed experimentally. For example, the ESC units used for this quadrotor could be programmed to provide either a linear power response or a linear speed response.

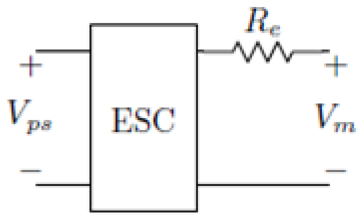


Figure 5.26 ESC model

Direct measurement of the three phase voltage or current can be made with a true RMS meter with sufficient bandwidth and current limits only if it is known whether the armature windings are connected in a wye or a delta configuration. An alternate approach is to model the ESC as shown in Figure 5.26. Accounting for the power consumed by the electronics, Pe, the power available to the motor can be determined from the power supply voltage and current as:

$$P_m = V_{ps} I_{ps} - P_e \tag{5.4}$$

5.4.2 Propeller Model

A propeller accelerates incoming air particles, throwing them towards down in a quadrotor, and it feels a force called thrust on itself in opposite direction. The propeller adds a velocity Δv to incoming air particle that has velocity v. The first half of this acceleration takes place in front of the propeller, and the second half behind the propeller. According to conservation of mass, the mass of air passing through the stream tube must be constant. This increased velocity results in a confinement of the stream tube passing through the propeller.

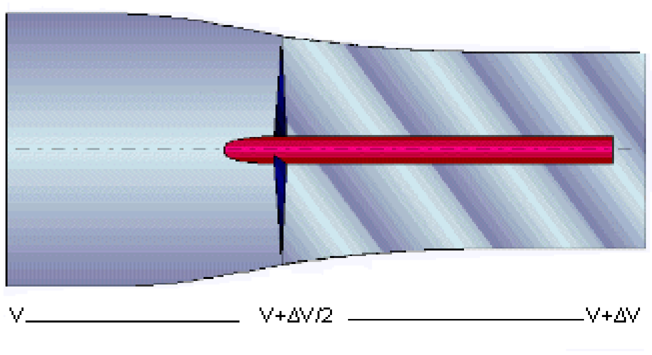


Figure 5.27 Propeller air stream tube

The thrust of a propeller depends on the volume of air accelerated per time unit, on the amount of the acceleration, and on the density of the medium. Based on momentum considerations, it can be expressed by the following formula:

$$T = \frac{\pi}{4}.D^2.\left(v + \frac{\Delta v}{2}\right).\rho.\Delta v \qquad (5.5)$$

Here, T, thrust [N]; D propeller diameter [m]; v, velocity of incoming flow [m/s]; Δv, additional velocity, acceleration by propeller [m/s]; ρ, density of fluid [kg/m³]. *(air: ρ=1.225 kg/m³)*

In the formula, thrust T increases when the diameter D increases or when the density ρ of the medium increases. If the additional velocity Δv is increased, the thrust also increases. For a propeller of a fixed diameter, working in a certain medium at a certain speed, thrust depends on the velocity increase Δv only.

5.5 MCU Modeling

Microcontroller is modeled with necessary peripherals such as UART, ADC, timer, PWM and interrupt modules.

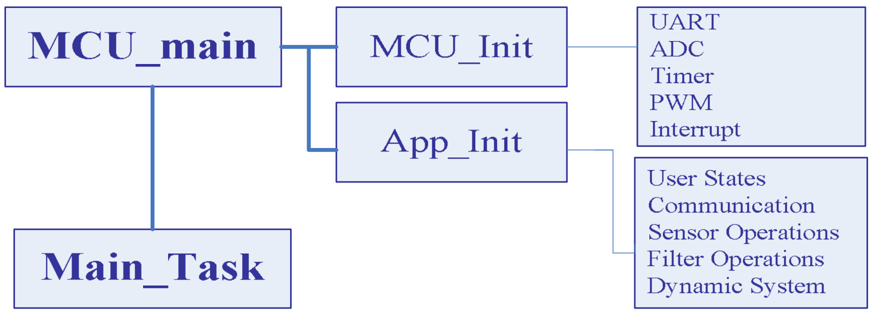


Figure 5.29 MCU model operations

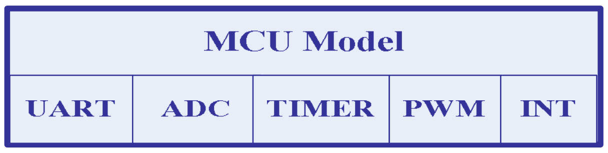


Figure 5.28 MCU model units

5.5.1 UART Model

UART is modeled with serial port device using WIN32 API. Using the API, a serial communication port can be opened with 'CreateFile()' function. And using 'WriteFile()' and 'ReadFile()' APIs, serial data send and receive operations are handled respectively.

5.5.2 ADC Model

ADC is modeled as six channels for each sensor and simply returns modeled sensor values. 10-bit digital value corresponds to analog sensor output in interval between 0 and 5 V.

5.5.3 Timer Model

Timer module is simulated with OpenGL/GLUT timer functions. OpenGL/GLUT provides time dependent function callbacks.

5.5.4 PWM Model

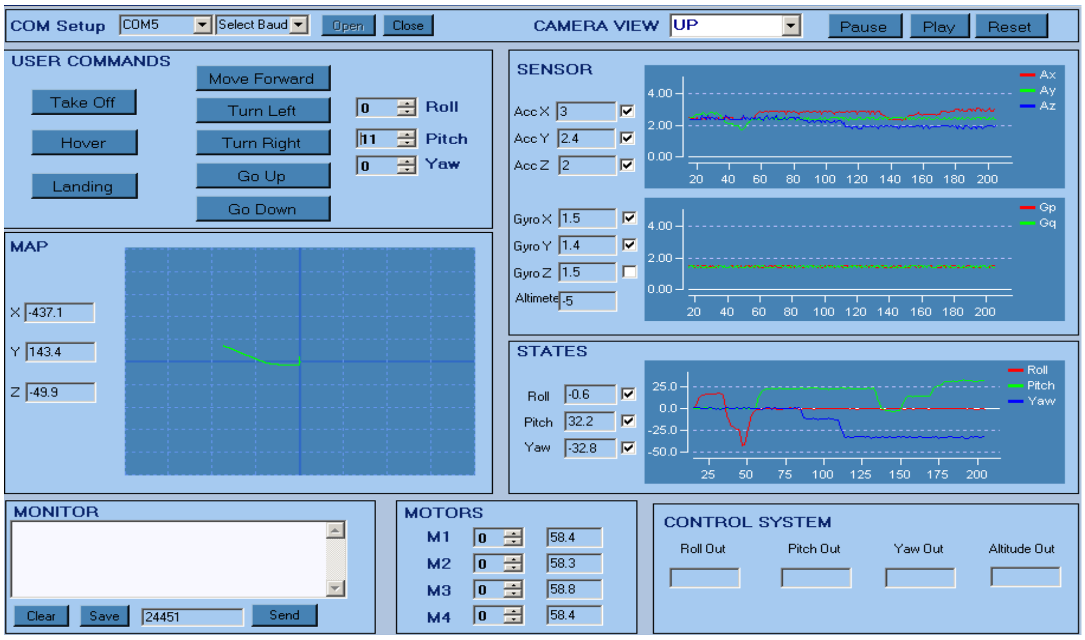
PWM Module just passes motor set values to simulator unit from control system.

5.5.5 Interrupt Model

Timer interrupts are created using timer module and simulator states are updated in specific time intervals. For example, sensor readings and user interface updates are handled with timer based interrupts.

5.6 Graphical User Interface

Graphical user interface is developed under Microsoft Visual Studio 2005 with Visual C++.Net programming language. It allows users to control the quadrotor from remote computer.



COM Setup: COM5, Select Baud
CAMERA VIEW: UP

USER COMMANDS
Roll: 0
Pitch: 11
Yaw: 0

MAP
X: -437.1
Y: 143.4
Z: -49.9

SENSOR
Acc X: 3 [✓]
Acc Y: 2.4 [✓]
Acc Z: 2 [✓]
Gyro X: 1.5 [✓]
Gyro Y: 1.4 [✓]
Gyro Z: 1.5 [ ]
Altimeter: -5

STATES
Roll: -0.6 [✓]
Pitch: 32.2 [✓]
Yaw: -32.8 [✓]

MONITOR: 24451

MOTORS
M1: 0, 58.4
M2: 0, 58.3
M3: 0, 58.8
M4: 0, 58.4

CONTROL SYSTEM
Roll Out
Pitch Out
Yaw Out
Altitude Out

Figure 5.30 User interface design

Firstly, user adjusts serial port settings and opens port. Sensor readings are received and plotted. Moreover, the vehicle six degrees of freedom states are observed in interface as roll, pitch, yaw and x, y, z. Basic commands are included such as take-off, hover, landing, turn left, turn right, go up and go down.

Graphic components are not part of Visual Studio. Third-party PlotLab's Scope Control component is used to visualize the coming sensor and state values. The Scope Control is a very fast multichannel plot component that can plot the data in logarithmic or linear modes.

5.7 Communication Protocol

Simulator software and user interface communicates continuously. User interface sends user commands and simulator sends its instant dynamic states. The communication medium is standard RS232. A special communication protocol is designed between user interface and simulator software. The protocol packet structure is shown in Figure 5.31 below.

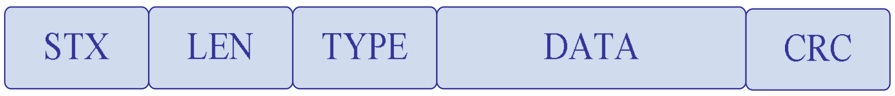


Figure 5.31 Protocol structure

STX: start of transmission

LEN : length of data from TYPE to CRC

TYPE : packet type as command or data

DATA: data to transmit

CRC: cyclic redundancy check for data integrity

CHAPTER SIX
CONCLUSION

The overall objective of this thesis was to design of a generic simulator for quadrotor UAV from point of view of electronic engineering. The focus was to be on the system that fulfills generic UAV operations and a simulator using them to visualize all system capabilities in a 3D medium. The derivation of the modeling equations, the implementation of a simulation model and controller design provided the understanding of the physical characteristics and the behavior of the quadrotor. The platform consists of several subsystems as the dynamics, sensors, motors, control system and communication system which have been examined deeply to understand how they are related in a real platform.

The overall simulation model is simplified to focus into main characteristics of quadrotor vehicle. The simulation model worked sufficiently for stable hover flight, but the air frictions neglected would need to be included for the model to be even more realistic. Moreover, the gyroscopic effects ignored in the development of the equations of motion could be added in future versions of the simulator. In addition, all components which were considered as an efficient exactly could be modeled more accurately. For example, the motors do not have 100% efficiency, because they do not output based on input instantly. This delay could be simulated with a first-order model of the motor. This model would be created from a gain and a pole that is related to the timing constant for the motor. The more complicated sensor modeling could also be designed to increase the accuracy of the simulation.

The platform gained great advancement after the simplified steady-state Kalman filter was implemented to combine data from the gyroscopes and accelerometers. The filter was able to handle the long term drift of the roll and pitch angles of the gyroscopes and removed the noise of the accelerometer. But the long term drift of the yaw angle is not considered. Inclusion of a magnetometer sensor to provide route information would allow the yaw drift to be corrected. While the filter implemented on the vehicle provided a true estimation of the sensor values, this can be improved even more. By observing the noise, the weight that each sensor system plays in the overall could be adjusted in real time. Moreover, an ultrasonic sensor alone is not adequate for altitude measurement beyond a certain limit and should be merged with a pressure sensor to estimate real altitude. Additionally, integration of a GPS model would be important to target path following and true autonomous flight. Environmental effects such as collisions, wind, temperature and raining models could be added to an advanced simulation in order to fully test the platform in a variety of conditions. Lastly, addition of power consumption model that simulates battery status and sound effects for motors could make simulator more realistic.

There are alternative platforms for simulator designs. For example, using Matlab Simulink modeling tool, overall simulator model could be designed with built-in system blocks and the outputs could be visualized with Opengl support.

A simulator for quadrotor UAV has been researched, specified, designed, implemented and tested for this project. This simulator is capable of modeling not only the physics of such a vehicle, but also the architecture of the electronic devices onboard. The software is run on the simulated microcontroller in order to test its suitability and can be transferred to the real platforms with little modification.

REFERENCES

Angeletti G. & Valente, J. R. & Iocchi, D. (2008). *Autonomous Indoor Hovering with a Quadrotor*. Sapienza University of Rome, Italy, B.Sc.

Beard, R. W. (2008). *Quadrotor Dynamics and Control.* Brigham Young University, USA, B.Sc.

Bouabdallah, S. (2007). *The Design and Control of Quadrotors with Application to Autonomous Flying*. Swiss Federal Institute of Technology, Ph.D.

Bresciani, T. (2008). *Modelling, Identification and Control of a Quadrotor Helicopter.* Lund University, Sweeden, M.Sc.

Bror, T. & Valdemar, K. (2003). *Structure of an Inertial Navigation System.* Darmstadt University of Applied Sciences, Germany, B.Sc.

Chronister, J. (2009). *Blender Basics* (3rd ed.). Retrieved May 13, 2009, from http://www.blender.org/education-help/tutorials/

Kemp, C. (2006). *Visual Control of a Miniature Quad-Rotor Helicopter*. University of Cambridge, England, Ph.D.

Latorre, E. S. (2007). *Propulsion system optimization for an unmanned lightweight quadrotor*. Universitat Politècnica de Catalunya, Spain, M.Sc.

Mitov, B. (2010). PLotLab Quick Start. Retrieved April 29, 2010, from http://www.mitov.com/manuals/QuickStart.pdf

Nice, E. B. (2004). *Desıgn of A Four Rotor Hovering Vehicle*. The Faculty of the Graduate School of Cornell University, USA, M.Sc.

Pedersen, A. H. (2006). *Test and Modeling of Four Rotor Helicopter.* Technical University of Denmark, M.Sc.

Pusa, J. (2009). *Strap down inertial navigation system aiding with non holonomic constraints using indirect Kalman filtering.* Department of Mathematics, Tampere University of Technology, Finland, M.Sc.

Rost, R. J. (2006). *OpenGL Shading Language* (2nd ed.) Addison Wesley Professional.

Rönnback, S. (2000*). Development of a INS/GPS navigation loop for an UAV.* Lulea University of Technology, Sweeden, M.Sc.

Sikiric, V. (2008). *Control of Quadrocopter.* The School of Electrical Engineering, Royal Institute of Technology, Sweeden, M.Sc.

Simon, D. (2001). *Kalman Filtering.* Retrieved June, 2001, from http://academic.csuohio.edu/simond/courses/eec644/

Stepaniak, M. (2008). *A Quadrotor Sensor Platform.* Ohio University, USA, M.Sc.

Woodman, O. J. (2007). *An introduction to inertial navigation.* Cambridge University.

Welch, G. (2001). *An Introduction to the Kalman Filter.* University of North Carolina.

Yiting W. (2009). *Development and Implementation of a Control System for a quadrotor UAV.* University of Applied Science Ravensburg-Weingarten, Germany, M.Sc.