QUADCOPTER

Team Members

Chaithanya
Vybhav Neelisetty
Rupam
Dinesh Chandra
Ashna
Aakash Desai
Siddharth
Akash Hirani
Guru
Vatsal Modi
Stavan
Praharsh
Axit
Karthik
Babulal
Kushal
Pranavarsh

Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and We were extremely privileged to have got this all along with the completion of our project. All that We have done is only due to such supervision and assistance and We would not forget to thank them.

We respect and thank respected Faculty members and Seniors, for providing us an opportunity to do the project work in **DRISHTI** and giving us all support and guidance which made us do the project duly.

We are extremely thankful to our super seniors Mr.**Sanket Ambedkar** ,Mr.**Varun Vidiyala**, Mr.**Ajinkya Zalte**, and Mr.**Bhanu Teja** for providing such a nice support and guidance, although they had a busy schedule managing their own projects.

We owe our deep gratitude to our project guide **B.Bhanu Teja** Sir, who took a keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all Professors and Teaching assistants of **Electronics Engineering Department and Mechanical Engineering Department** which helped us to work on our project work. Also, We would like to extend our sincere esteems to all staff in the laboratory for their timely support.

Team Aero

Abstract

The goal of our groups project is to design and build a Flight Controller for quadcopter. This means going through the process of researching previous models, performing calculations, purchasing individual parts, testing those parts, designing the final product, designing an Arduino based controller, and finally fitting everything together. All this is to be done with the minimum amount of outside help we can manage (i.e. not using predetermined code, not buying a boxed set of materials).First we have prepared the mathematical model of quadcopter to understand the dynamics of quadcopter and to tune parameters of Controller.Then we have designed control system for quadcopter.The data of the body movement is take by several sensors embedded on body mentioned in document.We have also applied advanced filters to get accurate data from sensors.

Contents

Introduction

Control System base of every industrial process which has its application in every field of science being medical instruments in hospital to satellites in the sky or underwater autonomous vehicles. Now, what is control system? Control System controls output based on input provided and can be open or closed-loop depending on the requirement of the user.
Aim: of this project was to design a Flight Controller for QuadCopter. Designing Flight Controller includes analyzing mechanics of Quadcopter and studying its differential equations which satisfactorily describe physical laws acting on it also known as mathematical model. Mathematical model predicts the behaviour of the Quadcopter in accordance with signals from the transmitter. Also, the main aim is to introduce different sensors and communication protocols through which feedback is provided to the control system.

Basics Physics of QuadCopter

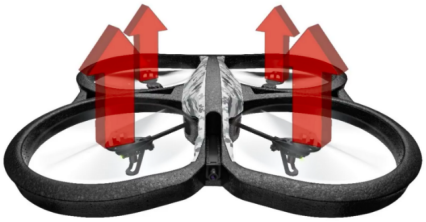
The Propellers act as wings.They generate thrust by rotating at Fast speeds,which pulls the air downwards and keeps the quadcopter in the air.

Physics

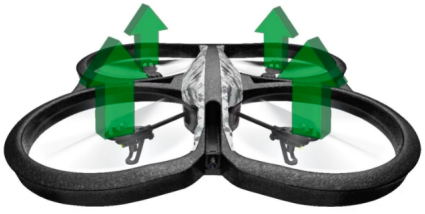
- Weight of quadcopter is cancelled by Thrust produced by propellers.
- Net direction of thrust gives direction of movement of quadcopter.
- conservation of angular momentum is done by rotating one set of opposite propellers in clockwise and another set of opposite propellers in anti-clockwise.

Movements of Quadcopter

1. **Take off**: to Rise above the ground, you need a net upward Force. The Motors generate Thrust that is greater than the Weight, making the quad rise upwards.



2. **Hover**: Hovering in Air is simple. Motors generate Thrust. The Thrust should equal the weight of the System. The two forces cancel and our drone Hovers.



3. **Attitude**:

- *Roll*:

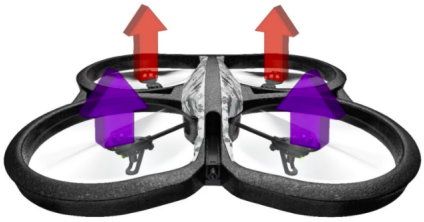
  To Roll towards the Left (Our Left), the Thrust is increased on the 2 motors(X configuration) or a motor(+ configuration) on the Right.We also decrease the Thrust on the 2 motors(X configuration) or a motor(+ configuration) on the Left.

  To Roll towards the Right (Our Right), the Thrust is increased on the 2 motors(X configuration) or a motor(+ configuration) on the Left.We also decrease the Thrust on the 2 motors(X configuration) or a motor(+ configuration) on the Right.

- *Pitch*:

  To Pitch Forwards (Towards us) The Power to the 2 rear motors(X configuration) or a rear motor(+ configuration) motors is increased. This creates a net forward force which causes the Drones nose to Pitch Downward.We also decrease the power to the 2 front motors(X configuration) or a front motor(+ configuration)motors to keep the angular momentum conserved.

  To Pitch Backwards (Away from us) The Power to the 2 rear motors(X configuration) or a rear motor(+ configuration) motors is decreased. This creates a net backward force which causes the Drones nose to Pitch Upward.We also increase the power to the 2 front motors(X configuration) or a front motor(+ configuration)motors to keep the angular momentum conserved.

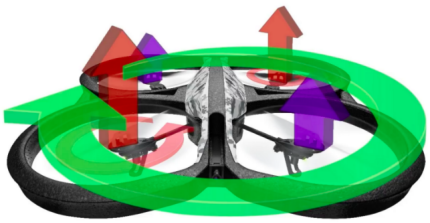


- *Yaw*:

To Yaw Clockwise. We increase the Thrust on the Anti-Clockwise moving Motors.Decrease the Thrust on Clockwise Rotating Motors. To keep the Net upward /downward force zero.There is a resulting Anti-Clockwise Torque.The Quad rotates Clockwise to conserve the Angular Momentum.

To Yaw Anti-Clockwise. We decrease the Thrust on the Anti-Clockwise moving Motors. Increase the Thrust on Clockwise Rotating Motors. To keep the Net upward/downward force zero. There is a resulting Clockwise Torque. The Quad rotates Anti-Clockwise to conserve the Angular Momentum.

There are two kind of propellers.Clockwise and Anti- Clockwise propellers which will produce thrust upward in their respective name directions. We must keep two clockwise propellers opposite arms and anti-clockwise propellers in another opposite arm so the angular momentum produced by one set propellers is countered by another set of propellers and angular momentum is conserved.



Mathematical Model

Importance of Mathematical Model

Mathematical Model of a system is used to express physical systems using mathematical equations. After developing a conceptual model of a physical system it is natural to develop a mathematical model that will allow one to estimate the quantitative behaviour of the system. In our project, we are using the mathematical model to understand the dynamics of quadcopter completely and to tune the control system parameters automatically.

Introduction

1. *Assumptions*:

   - The quadcopter is a rigid body.
   - Quadcopter is symmetrical.
   - Center of mass coincides with the geometric centre.
   - inertia of the motor is considered to be neglected.
   - Earth is assumed to be flat.

2. *Notations*:

   Due to the complexity of a system with 6 degrees of freedom, various methods of notation have been developed and are required in order to sufficiently describe the critical variables.

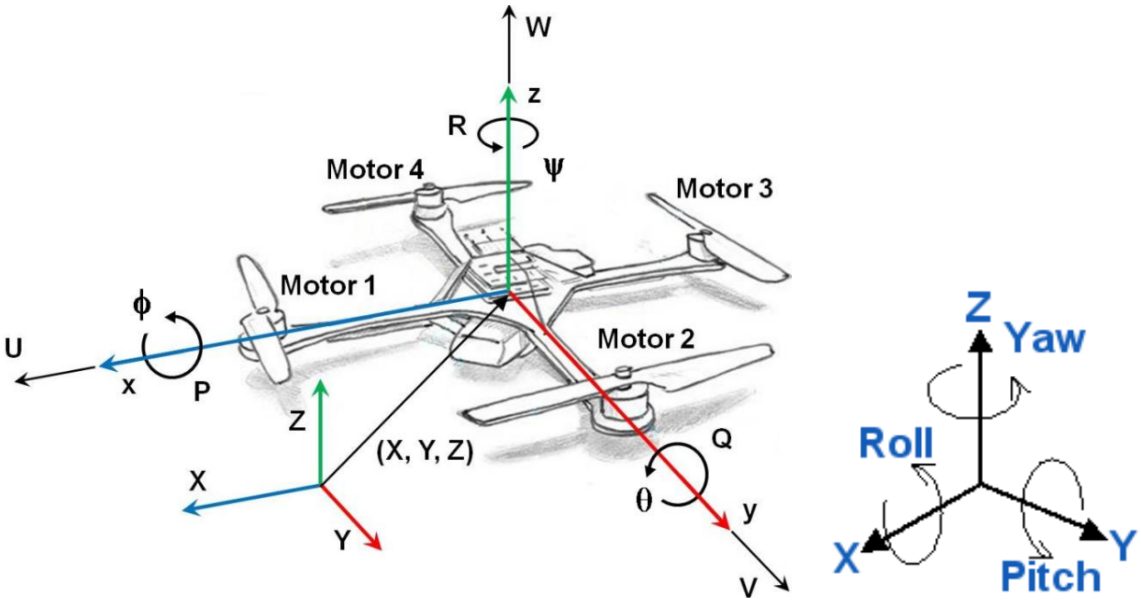
   $$^{y}x^{z}_{h|f}$$

   The base variable x is state variable.$\dot{x}$ is its first derivative.$\ddot{x}$ is second derivative.the left superscript y tells us from which frame the x variable is taken or performed.While the right superscript z tells us from which frame the x variable vector components are represented.h represents from which point the variable value is and f is from which frame.i is inertial frame. b is body frame.CM is Center of Mass.

3. Coordinate system:

   Another important aspect of the math model is the coordinate system that is used.The

chosen model and conventions will become very important as you work through your model, so be sure to keep your decisions in mind and clearly documented. The coordinate system will vary whether you use a plus (+) or X configuration. In this project X configuration is used because if you have a "+" configuration, then the thrust forces are applied at a distance r. If you have an X configuration, then the thrust forces are applied at distance of r*cos($\pi/4$) approximately 0.71*r since the arms are at a 45 degree angle from the axis of rotation. The moment of inertia is the same when you do the math, so the difference is really that you can torque with all four motors, and therefore have $\sqrt{2}$ more available torque to rotate. This means you can get about 41% more rotational acceleration from an X than a "+".



Moment of Inertia Matrix

Moment of Inertia matrix w.r.t body is denoted by J^b.Moment of Inertia w.r.t x is denoted by J_{xx}.Moment of Inertia w.r.t y is denoted by J_{yy}.Moment of Inertia w.r.t z is denoted by J_{zz}.

$$J^b = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}$$

Thrust Coefficient(C_T) and Torque Coefficient(C_Q)

Thrust Coefficient:

The motors' thrust is the driving force behind all quadcopters.The thrust, , provided by a single motor/prop system can be calculated as follows

$$T{=}C_T \rho A_r r^2 \bar{\omega}^2$$

- C_T:thrust coefficient
- ρ :density of air
- A_r:the cross sectional area of the propeller's rotation.
- r :is the radius of the rotor.
- $\bar{\omega}$:the cross is the angular velocity of the rotor.

For simple flight modelling, a lumped parameter approach can be used to simplify the characterization process

$$T{=}c_T \bar{\omega}^2$$

where c_T is the lumped parameter thrust coefficient that pertains to the individual motor/prop system.

Torque Coefficient:

In order to understand the motor effect on yaw, the torque force of the motor/prop system must also be determined and can be done in a similar fashion to that of the thrust tests. The related lumped parameter equation is shown below

$$Q{=}c_Q \bar{\omega}^2$$

where c_Q is the lumped parameter torque coefficient that pertains to the individual motor/prop system. To calculate the c_T and c_Q done two experiments mentioned further.

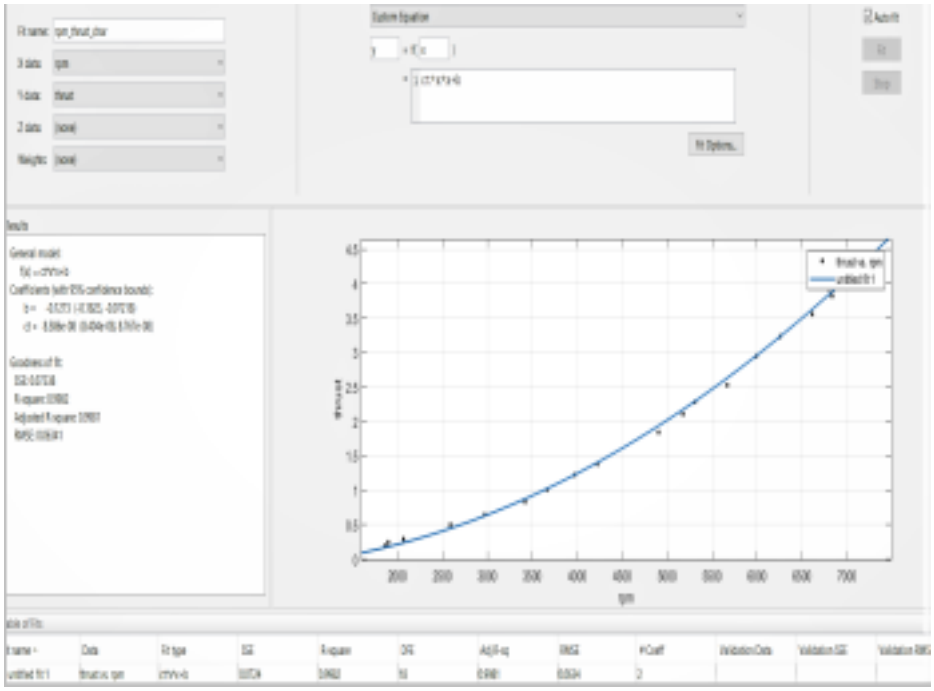
C_T experiment

Principle:Rotating BLDC with the propeller(Desired) Produces a thrust in the downward direction

Requirements:BLDC,Propeller(1045),ESC,LIPO(or)Any Energy Source,Transmitter and Receiver,Weighing Machine,ARDUINO,IR sensor

Procedure: Make the connections of BLDC with ESC,LIPO and Receiver and mount a propeller to it. Now place the BLDC on the Weighing machine such that BLDC should not move and wires should not interfere the Propellers. Fix a white strip on the BLDC and place a Ir sensor nearer to the BLDC (should not touch propellers) and make IR sensor connections with the Arduino.Upload Code(A) into the Arduino and open the serial monitor.

Now by changing the throttle in Transmitter from 1000s to 2000s note down the corresponding RPM from ARDUINO serial monitor and Thrust from Weighing machine Now using these values plot a graph in MATLAB Curve Fitting App and find the thrust coefficient (= 8.586e-08).



this graph was plotted by Curve fitting tool in Matlab Simulink.

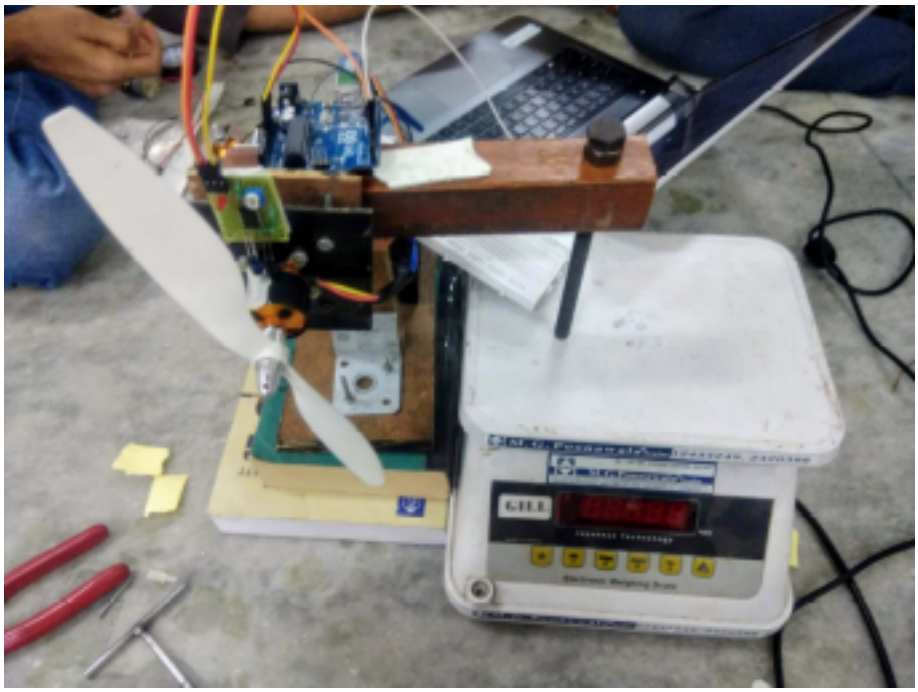
C_Q experiment

Principle: Rotating BLDC with the propeller(Desired) placed on a free end of a rigid support.Produces a Torque due to force applied on free end We know that

$$Torque = Force * perpendiculardistance(r)$$

Requirements: BLDC,Propeller(1045),ESC,LIPO(or)Any Energy Source,Transmitter and Receiver,Weighing Machine,ARDUINO,IR sensor

Procedure: Make the connections of BLDC with ESC,LIPO and Receiver and mount a propeller to it. Now place the BLDC on the wooden frame as shown in the figure() and place the mechanism free ens screw on the Weighing machine such that BLDC should not move and wires should do not interfere with the Propellers. Fix a white strip on the BLDC and place a Ir sensor nearer to the BLDC (should not touch propellers) and make IR sensor connections with the Arduino.Upload Code(A) into the Arduino and open the serial monitor.



Now by changing the throttle in Transmitter from 1000s to 2000s note down the corresponding RPM from ARDUINO serial monitor and force from Weighing machine.Measure the distance between free end screw and rigid support and note it down as r. And from the product of r and force we get Torque Now using these values plot a graph in MATLAB Curve Fitting App and find the thrust coefficient(c_Q)

Thrust-Torque Matrix

we can create a matrix describing the thrusts and torques on the system like that shown below

$$\begin{bmatrix} \sum T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ -d*c_T & d*c_T & d*c_T & -d*c_T \\ -d*c_T & -d*c_T & d*c_T & d*c_T \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} \bar{\omega_1}^2 \\ \bar{\omega_2}^2 \\ \bar{\omega_3}^2 \\ \bar{\omega_4}^2 \end{bmatrix}$$

We got above matrix by several experiments performing on it.To get Final matrix we should consider Gyroscopic forces.But in this project, it is neglected for simplicity.

$$M^b_{A,T} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}$$

$$F^b_{A,T} = \begin{bmatrix} 0 \\ 0 \\ \sum T \end{bmatrix}$$

State Equation

A State-space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations or difference equations. State variables are variables whose values evolve through time in a way that depends on the values they have at any given time and also depends on the externally imposed values of input variables.Output variables values depend on the values of the state variables.

Angular Velocity State Equation

$$^b\dot{\omega}^b_{b|i} = (J^b)^{-1}(M^b_{A,T} - \Omega^b_{b|i} J^b \omega^b_{b|i}) = \begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix}$$

$$\Omega^b_{b|i} = \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}$$

$$\omega^b_{b|i} = \begin{bmatrix} P \\ Q \\ R \end{bmatrix}$$

- P:angular rate w.r.t x in body frame
- Q:angular rate w.r.t y in body frame
- R:angular rate w.r.t z in body frame
- $^b\dot{\omega}^b_{b|i}$:the angular acceleration across each axis in the body frame with respect to the inertial frame
- $\Omega^b_{b|i}$:cross-product matrix for rotational velocity
- $\omega^b_{b|i}$:the rotational velocity of the quadcopter body within the body frame

Euler Kinematic Equation

Used determine the rate of change of the Euler angles in the inertial frame.According to the aerospace rotation sequence, the rotation of an aircraft is described as a rotation about the z-axis (yaw) then a rotation about the y- axis (pitch) followed by a rotation about the x-axis (roll). Each rotation is made based on a right-handed system and in a single plane.Using these three rotations a composite rotation matrix can be created which can transform the motion of the aircraft from the body frame to a new reference frame. The resulting rotation matrix transforms rotations from the body frame with respect to the inertial frame and can be found using matrix multiplication. Below s, c, and t represent sine, cosine, and tangent functions respectively.

$$C_{b|i} = \begin{bmatrix} c(\theta)c(\phi) & c(\theta)s(\phi) & -s(\theta) \\ (-c(\phi)s(\psi) + s(\phi)s(\theta)c(\psi)) & (c(\phi)c(\psi) + s(\phi)s(\theta)s(\psi)) & s(\phi)c(\theta) \\ (s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi)) & (-s(\phi)c(\psi) + c(\phi)s(\theta)s(\psi)) & c(\phi)c(\theta) \end{bmatrix}$$

$$C_{i|b} = (C_{b|i})^{-}1$$

$$\dot{\Phi} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = C_{i|b} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} = H(\Phi)\omega^b_{b|i} \textbf{ (Euler Kinematic Equation)}$$

Velocity State Equation

$$^b\dot{v}^b_{CM|i} = (\frac{1}{m})F^b_{A,T} + g^b - \Omega^b_{b|i}\omega^b_{CM|i} = \begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix}$$

$$g^b = C_{i|b}g^i$$

$$g^i = 9.8m/s$$

$^b\dot{v}^b_{CM|i}$:the linear acceleration of the center of mass in the body frame with respect to the inertial frame. m:mass of system

Position State Equation

$$^b\dot{P}^b_{CM|i} = C_{i|b}v^b_{CM|i} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix}$$

$v^b_{CM|i}$ is simply the velocity of the quadcopter in the body frame that is rotated into the inertial frame. This state equation allows us to determine the velocity of the quadcopter in the X, Y, and Z directions of the inertial frame.

Linear System

Introduction

A linear system is a mathematical model of a system based on the use of a linear operator. Linear systems typically exhibit features and properties that are much simpler than the nonlinear case. As a mathematical abstraction or idealization, linear systems find important applications in automatic control theory, signal processing, and telecommunications. For example, the propagation medium for wireless communication systems can often be modeled by linear systems.

Definition of a linear system

It satisfies the superposition principle.A system satisfies the superposition principle if the following conditions are satisfied:
1. Homogeneity: Multiplying the input by any constant, multiplies the output by the same constant homogeneity.

$$H\{a \cdot x(t)\} = a \cdot H\{x(t)\}$$

2. Additivity: The response to several inputs applied simultaneously is the sum of individual response to each input applied separately.

$$H\{x(t) + y(t)\} = H\{x(t)\} + H\{y(t)\}$$

The importance of being linear

While there is a good deal of mathematics behind adaptive controllers, it's not particularly hard mathematics. The reason for This is that traditionally most controllers are linear. We can take advantage of this linearity to make the equations relatively easy to manipulate.

The main reason why linear systems are so familiar, is not because they are so ubiquitous (in fact, one author has pointed out that dividing nature into linear and nonlinear systems is like having "non elephant" biology as a special sub-field, and missing the fact that most systems are not linear).

Linearisation of QUADCOPTER Model

The system of equations are nonlinear coupled equations. Therefore, the system of equations was linearized using small assumptions.

The quadrotor is assumed to have very low linear and angular velocities when in motion and assumed not to tilt beyond (-30,30) degrees in pitch and roll. The quadrotor is always flying at near hovering conditions and Coriolis and rotor moment of inertia terms can be neglected.

From our assumption:

$$\cos\theta = \cos\phi = 0$$

$$\sin\theta = \sin\phi = 0$$

Results:

$$a_x = -g + \frac{U1}{m}$$

$$\ddot{\phi} = \frac{U_2}{I_{xx}}$$

$$\ddot{\theta} = \frac{U_3}{I_{yy}}$$

$$\ddot{\psi} = \frac{U_4}{I_{zz}}$$

From this equation, we can say that our equation is linear and SISO

Control System

A control system manages, commands directs, or regulates the behaviour of other devices or systems using control loops. It can range from a single home heating controller using a thermostat controlling a domestic boiler to large Industrial control systems which are used for controlling processes or machines.

Features of a Control System

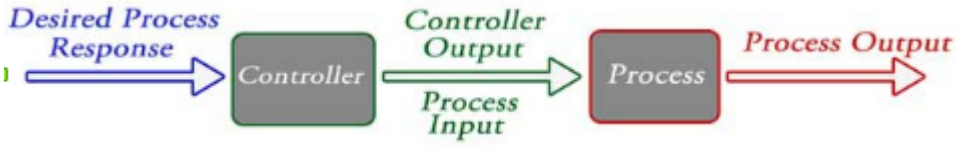
The main feature of a control system is that there should be a clear mathematical relationship between input and output of the system. When the relation between input and output of the system can be represented by a linear proportionality, the system is called a linear control system. Again when the relationship between input and output cannot be represented by single linear proportionality, rather the input and output are related by some non-linear relation, the system is referred to as a non-linear control system.

There are two main types of control system. They are as follow

1. Open loop control system
2. Closed loop control system

Open loop control system

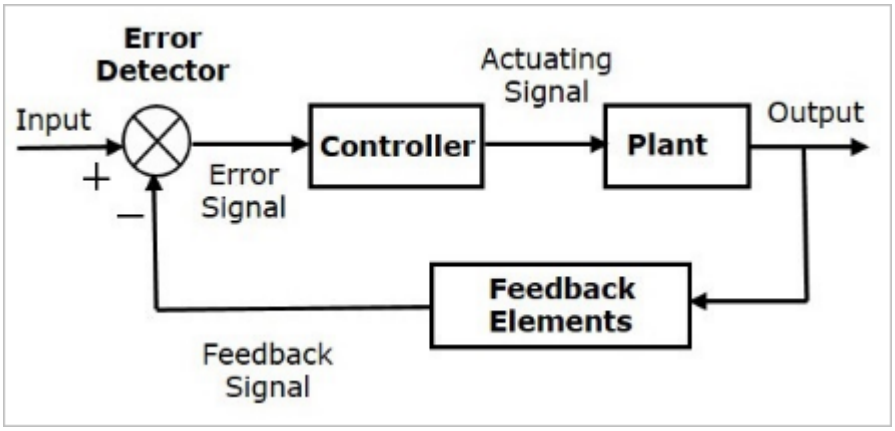
A control system in which the control action is totally independent of output of the system then it is called an open loop control system. A manual control system is also an open loop control system. The figure below shows a control system block diagram of an open loop control system in which process output is totally independent of the controller action.



Closed Loop Control System

Control system in which the output has an effect on the input quantity in such a manner that the input quantity will adjust itself based on the output generated is called closed loop control system. Open loop control system can be converted in to closed loop control system by providing

feedback. This feedback automatically makes the suitable changes in the output due to external disturbance. In this way closed loop control system is called automatic control system. Figure below shows the block diagram of closed loop control system in which feedback is taken from output and fed in to input.



For continuously modulated control, a feedback controller is used to automatically control a process or operation. The control system compares the value or status of the process variable (PV) being controlled with the desired value or set point (SP), and applies the difference as a control signal to bring the process variable output of the plant to the same value as the set point.

Feedback Systems

In the case of linear feedback systems, a control loop including sensors, control algorithms, and actuators is arranged in an attempt to regulate a variable at a set point (SP). An everyday example is the cruise control on a road vehicle; where external influences such as hills would cause speed changes, and the driver has the ability to alter the desired set speed.

Advantages of Feedback in Control Compared to open-loop control, feedback can be used to

- Reduce the sensitivity of a systems transfer function to parameter changes.
- Reduce steady-state error in response to disturbances.
- Reduce steady-state error in tracking a reference response (speed up the transient response).
- Stabilize an unstable process.

PID Controller

Introduction

A proportional-integral-derivative controller (PID controller) is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates the correction value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable.

Algorithm

The PID controller algorithm involves three separate constant parameters, and is accordingly sometimes called three-term control.

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right)$$

PID Controller Theory

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining Correction as the controller output,Controller manufacturers arrange the Proportional, Integral and Derivative modes into three different controller algorithms or controller structures. These are called Interactive, Non-interactive, and Parallel algorithms. Some controller manufacturers allow you to choose between different controller algorithms as a configuration option in the controller software.

The PID Algorithms are:

- Interactive Algorithm
- Non-Interactive Algorithm
- Parallel Algorithm

Interactive Algorithm

$$CO = K_c\left[E + \frac{1}{T_i}\int Edt\right]\left[1 + T_d\frac{dE}{dt}\right]$$

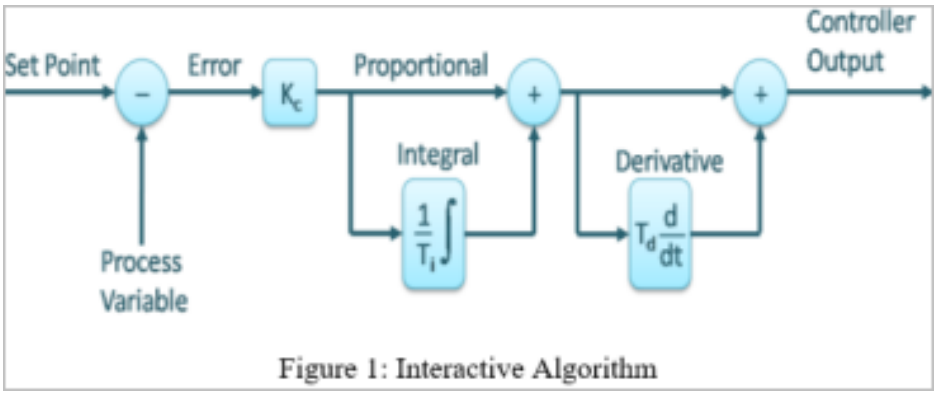


Figure 1: Interactive Algorithm

Non-Interactive Algorithm

$$CO = K_c \left[E + \frac{1}{T_i} \int Edt + T_d \frac{dE}{dt} \right]$$

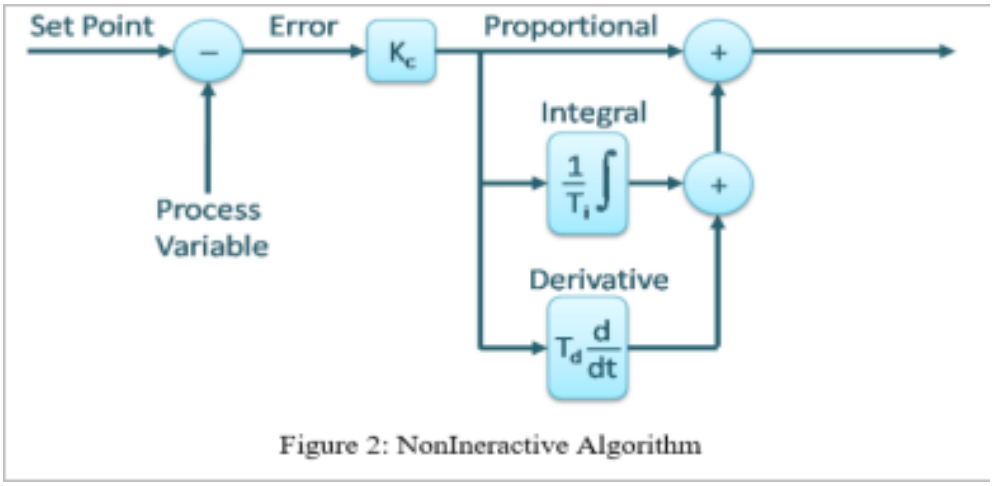


Figure 2: NonIneractive Algorithm

Parallel Algorithm

$$CO = K_p \times E + K_i \times \frac{1}{T_i} \int Edt + K_d \times \frac{dE}{dt}$$

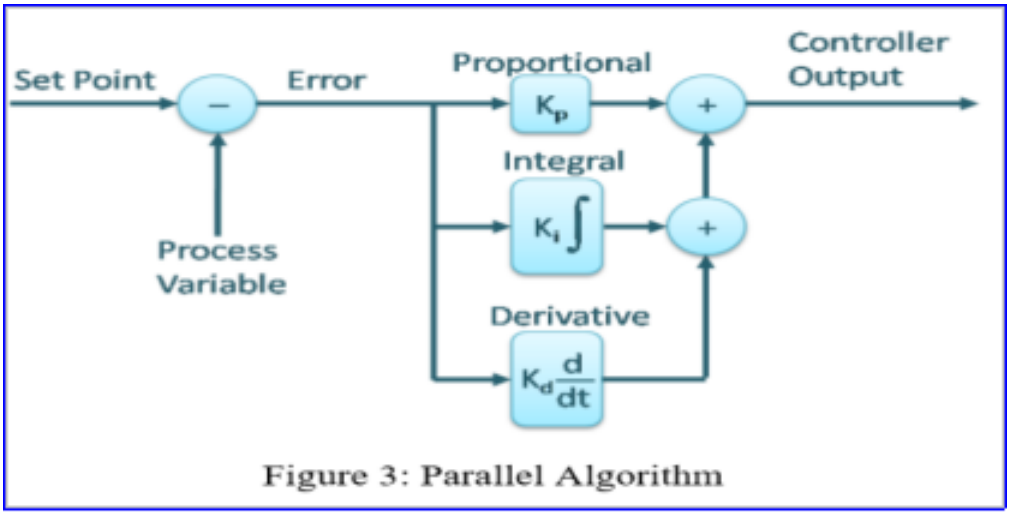


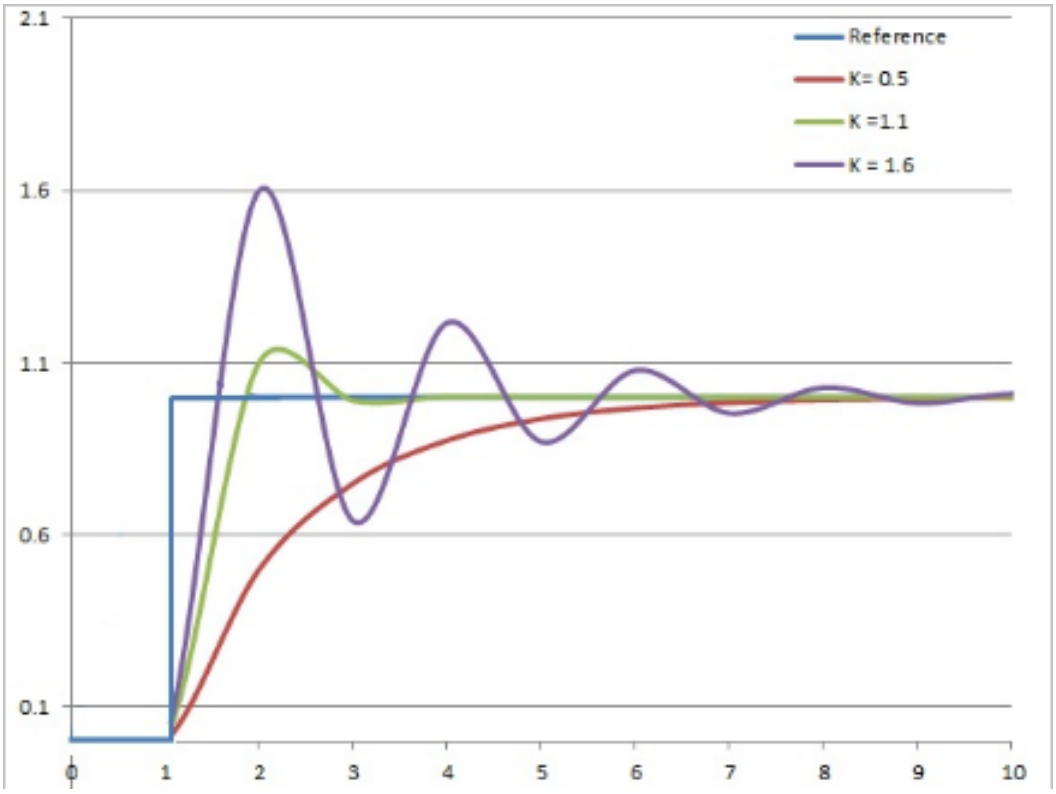
Figure 3: Parallel Algorithm

Proportional Term

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant. The proportional term is given by:

$$Pout = K_p e(t)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change.

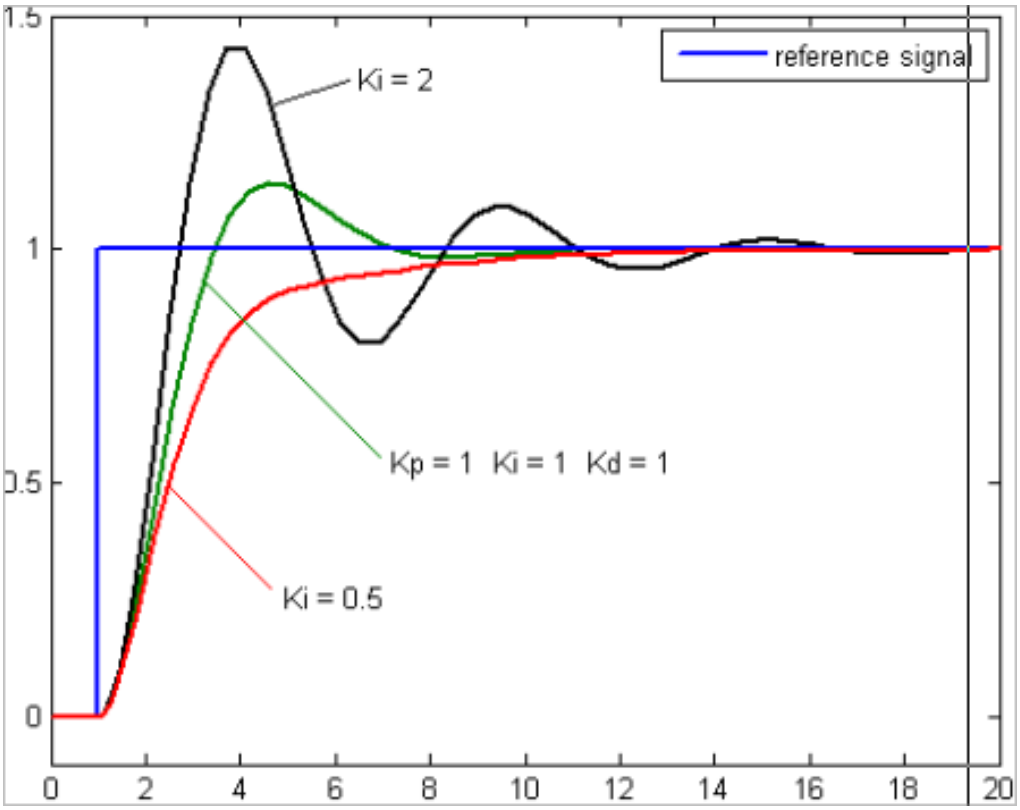


Integral Term

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain Ki and added to the controller output.

$$t_{out} = K_i \int_0^t e(t)dt$$

The integral term accelerates the movement of the process towards set-point and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set-point value.

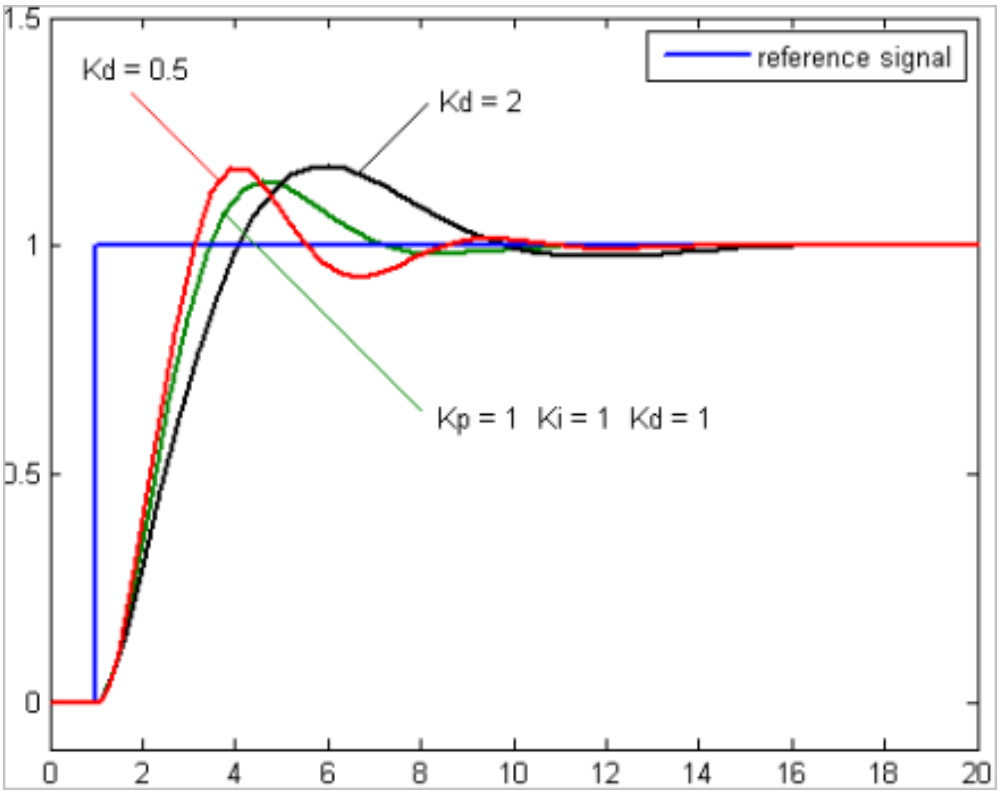


Derivative Term

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain Kd. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, Kd The derivative term is given by

$$D_{out} = K_d \frac{de(t)}{dt}$$

Derivative action predicts system behavior and thus improves settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low pass filtering for the derivative term, to limit the high frequency gain and noise. Derivative action is seldom used in practice though - by one estimate in only 20% of deployed controllers- because of its variable impact on system stability in real-world applications.



Parameter	Risetime	Overshoot	Settling Time	Steady State Error	Stability
K_p	Decrease	Increase	Small Change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor Change	Decrease	Decrease	No Effect	Improve if K_d is small

Tuning

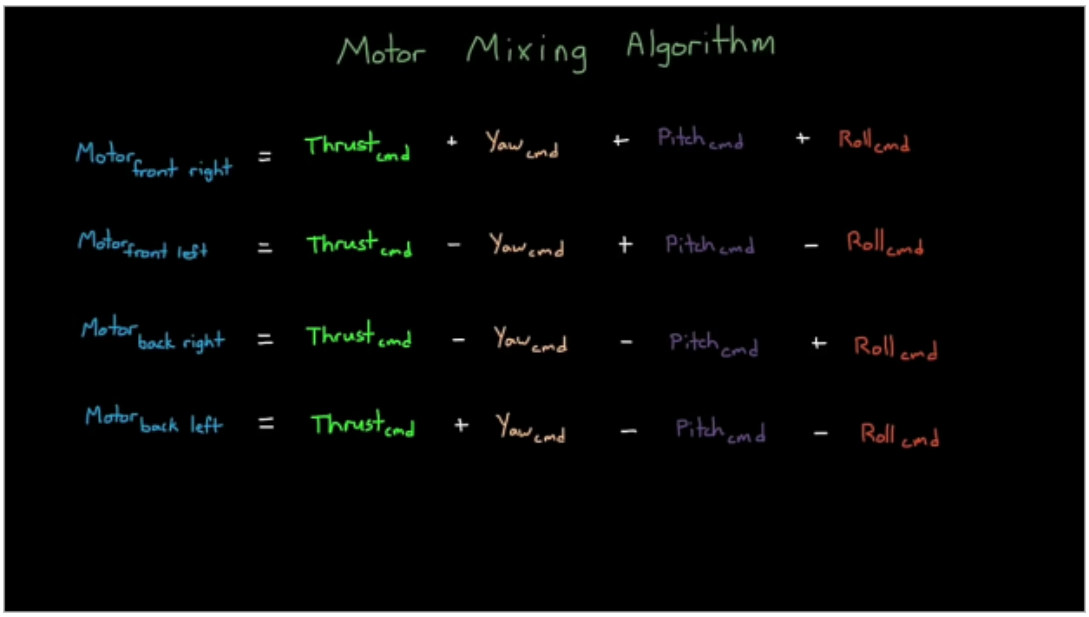
All general methods for control design can be applied to PID control. A number of special methods that are tailor-made for PID control have also been developed, these methods are often called tuning methods. Irrespective of the method used it is essential to always consider the key elements of control, load disturbances, sensor noise, process uncertainty and reference signals.The most well known tuning methods are those developed by Ziegler and Nichols.The methods are based on characterization of process dynamics by a few parameters and simple equations for the controller parameters. It is surprising that the methods are so widely referenced because they give moderately good tuning only in restricted situations. Types in Ziegler and Nichols methods

✓The Step Response Method

✓The Frequency Response Method

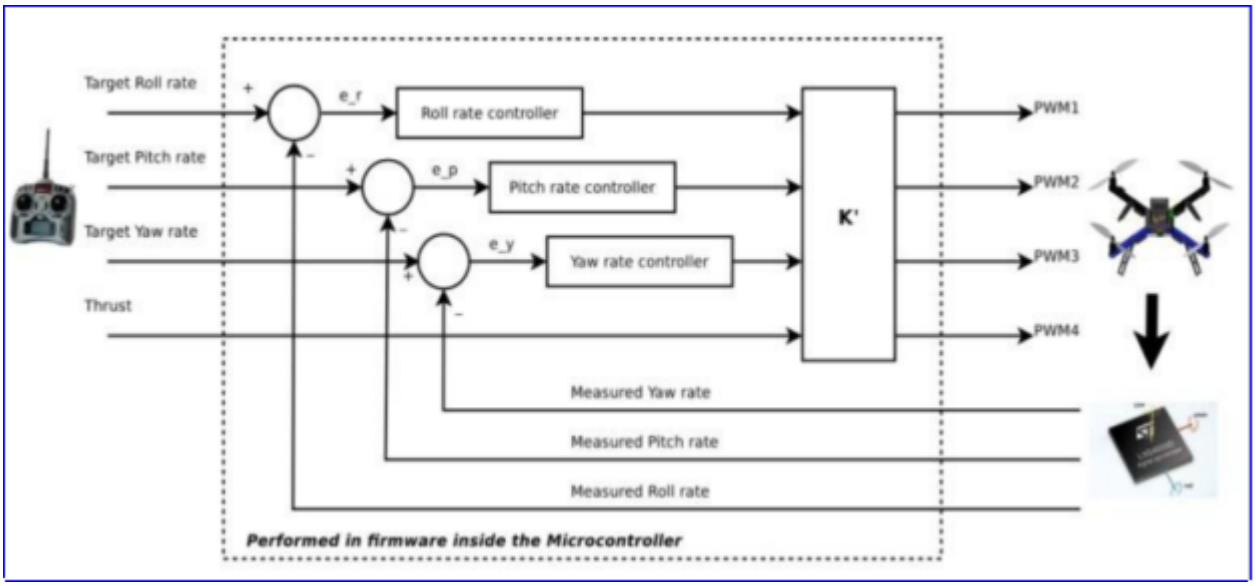
Motor Mixing Algorithm

Here $roll_{cmd}$, $pitch_{cmd}$ and yaw_{cmd} are corrections.
Example: If roll, yaw and pitch is zero than each esc gets the same pwm and quadcopter hover. If only pitch is given in input then according to the algorithm upward force of front two motors increases and back two motors decreases. As a result pitch is achieved.



Control System of the Quadcopter

PWM Signal from receiver are converted to Roll,Pitch,Yaw and Thrust.To find the error we take reference of the IMU and measure the error of corresponding Roll,Pitch,Yaw and feed into the PID controller and from the PID we take output as corrections needed in roll and pitch to maintain the given input angle for receiver.Through Motor Mixing Algorithm PWM for each escs is calculated.


Target Roll rate
Target Pitch rate
Target Yaw rate
Thrust
e_r
e_p
e_y
Roll rate controller
Pitch rate controller
Yaw rate controller
K'
PWM1
PWM2
PWM3
PWM4
Measured Yaw rate
Measured Pitch rate
Measured Roll rate
Performed in firmware inside the Microcontroller

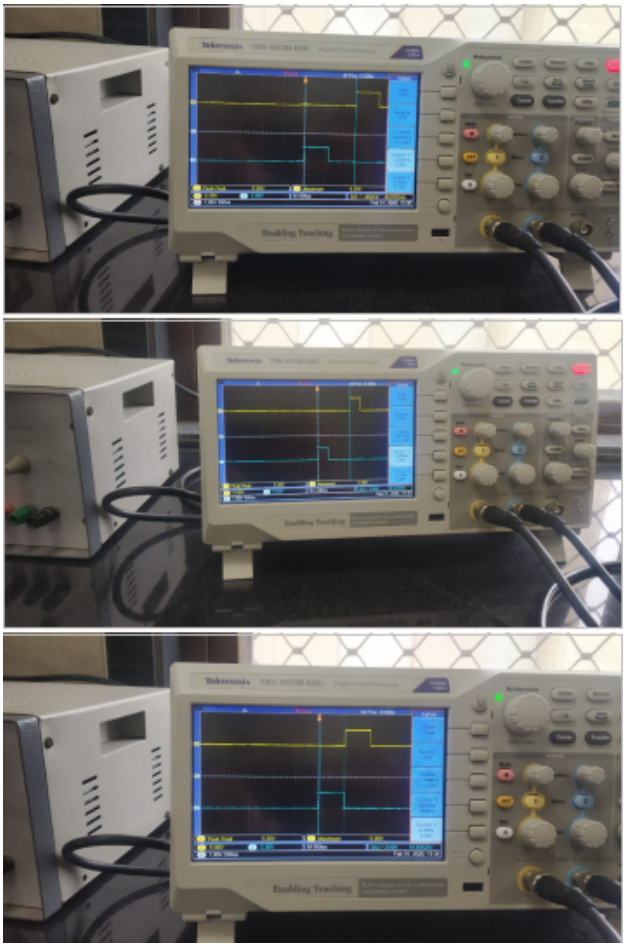
PWM Capturing and Generation

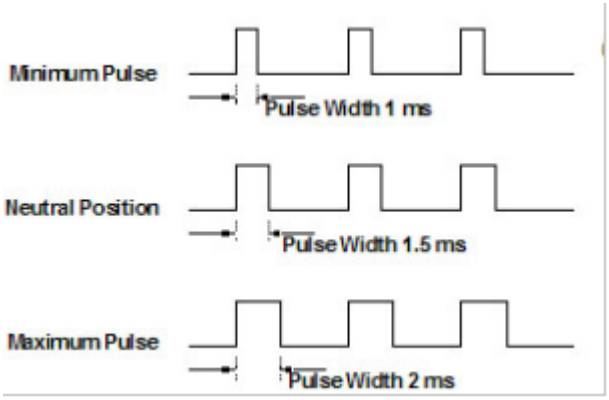
PWM Capturing

The receiver we used to control the ESC gives PWM as a signal. PWM stands for pulse width modulation. Its an analog signal where the length of the pulse specifies the throttle position. The length of the signal pulse normally varies between 1000s and 2000s (micro seconds), with 1000s being the minimum and 2000s the maximum.

Testing of PWM signal

A PWM signal from the receiver was seen with the help of the Oscilloscope.





Logic for capturing PWM signal

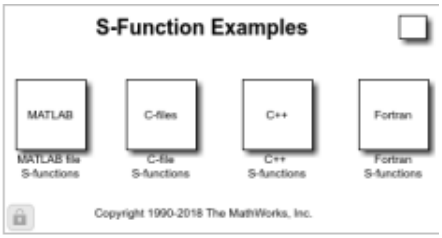
In order to find the PWM values, we need to find the time for which signal remains high. Time is found by taking the difference of rising edge time and falling edge time.

Interrupts facility of microcontroller is used to identify the rising edge and falling edge of pulse.Timer such as micros or millis returns the value of time elapsed. Micros function is used to get the time.

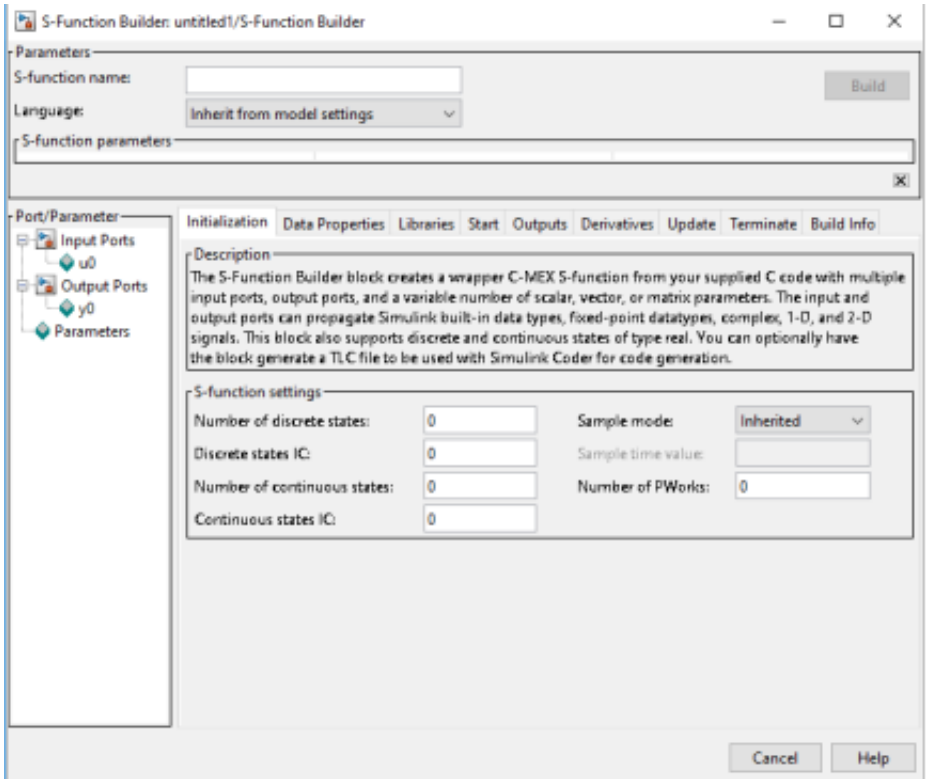
There is no block of interrupts in simulink support package for arduino so after understanding the above logic ,code is written in c language. To run this c code in simulink there is something called S- function in simulink.

S function Builder

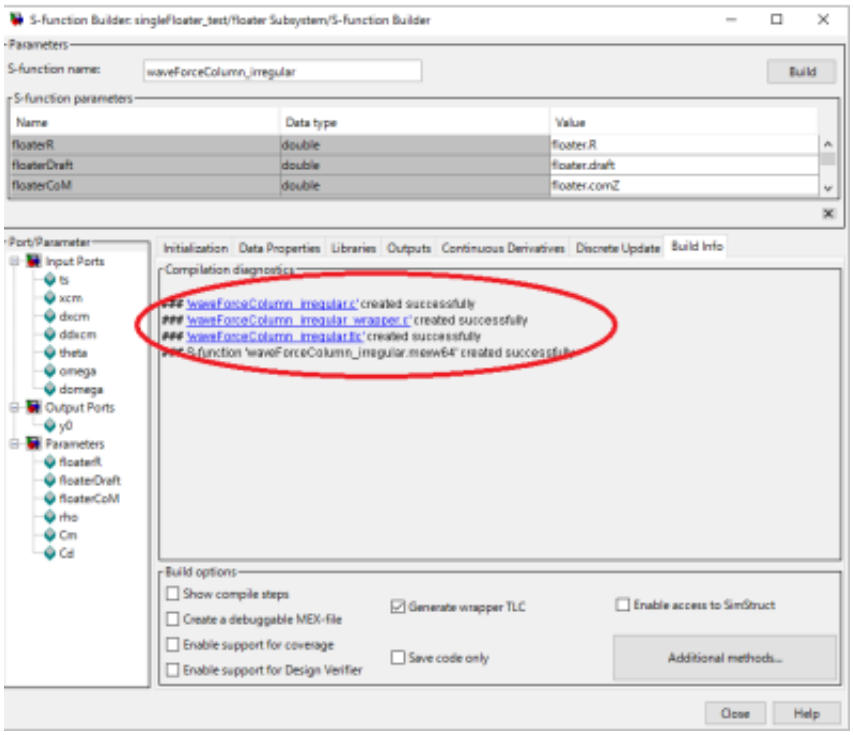
S function is a place where you can generate your own code in c or c++ language. Its main function is to give all the freedom to the programmer to write its own code in his term.



To use S function builder, first of all, you must download the C compiler for the Matlab to generate and build code in C so then it can convert the mex file according to your hardware. S function code is saved as a .mex file.



First block is **initialization** where you have to write a discrete function as 1. Then comes **data properties** where you can define inputs and outputs and their data type and parameters. Next one is **libraries** where you define all your libraries you are going to use in the code. Coming next is **START** where you write your code which you want to use only once like for initialization of i2c service or enabling pins to high, it's the same as the void setup in arduino. Next blocks are **output** and update where you can write all the other stuff. For instance, you can write print code in the output block and calculations stuff in the update block. Be aware that firstly the output block will be executed then the update block. After doing all this stuff you have to just build and it is necessary to write the name of the function you can look at the top part of the S function builder block. After building this up 3 files will be generated.



And in the file which has the same name as of the S function with the wrapper written with it. You will find that its extension is .c which you have to convert to .cpp for it to work on arduino then double click that file (simply open it up). You have to write **extern C** before every function whose name is written with wrapper. Using this S function builder C code of PWM capturing is written and its block is created in simulink.

PWM Generation

To generate PWM signals Servo Block is used in Matlab Simulink. The 1000μs to 2000μ is mapped to 45deg to 142deg.

MPU6050 (Gyroscope + Accelerometer + Temperature) Sensor Module

Introduction

MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small packages. Also, it has additional feature of on-chip Temperature sensor. It has I2C bus interface to communicate with the microcontrollers.

It has Auxiliary I2C bus to communicate with other sensor devices like 3-axis Magnetometer, Pressure sensor etc. If 3-axis Magnetometer is connected to auxiliary I2C bus, then MPU6050 can provide complete 9-axis Motion Fusion output.

PURPOSE: We have used MPU to provide feedback of state of Quadcopter (in terms of rotation with respect to axis) to Control System.

MPU6050 sensors

3-Axis Gyroscope

The MPU6050 consist of 3-axis Gyroscope with Micro Electro Mechanical System (MEMS).It is used to detect rotational velocity along the X, Y, Z axes as shown in below figure.

3-Axis Accelerometer

The MPU6050 consist 3-axis Accelerometer with Micro Electro Mechanical (MEMs) technology. It used to detect angle of tilt or inclination along the X, Y and Z axes as shown in below figure.

DMP (Digital Motion Processor)

The embedded Digital Motion Processor (DMP) is used to compute motion processing algorithms. It takes data from gyroscope, accelerometer and additional 3rd party sensor such as a magnetometer and processes the data. It provides motion data like roll, pitch, yaw angles, landscape and portrait sense etc. It minimizes the processes of host in computing motion data. The resulting data can be read from DMP registers.

On-chip Temperature Sensor

On-chip temperature sensor output is digitized using ADC. The reading from temperature sensor can be read from sensor data register.

$$Temperatureindegrees C = ((temperaturesensordata)/340 + 36.53)/c.$$

Pin Connections

- **SCL**: Serial Clock pin. Connect this pin to microcontrollers SCL pin.
- **SDA**: Serial Data pin. Connect this pin to microcontrollers SDA pin.
- **GND**: Ground pin. Connect this pin to ground connection.
- **VCC**: Power supply pin. Connect this pin to +5V DC supply.

I2C(Inter-Integrated Circuit

I2C Interface

I2C is a two-wire interface comprised of the signals serial data (SDA) and serial clock (SCL). In general, the lines are open-drain and bi-directional. In a generalized I2C interface implementation, attached devices can be a master or a slave. The master device puts the slave address on the bus, and the slave device with the matching address acknowledges the master. The MPU-60X0 always operates as a slave device when communicating to the system processor (microcontroller in our case), which thus acts as the master.SDA and SCL lines typically need pull-up resistors to VDD.The maximum bus speed is 400 kHz.

The slave address of the MPU-60X0 is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin AD0. This allows two or more MPU-60X0s to be connected to the same I2C bus.**Logic level on pin AD0 of MPU60X0 which is to be connected to I2C bus should be zero (AD0 should be connected to ground)**.

I2C Communication

- The first step in a TWI transmission is to transmit a START condition. This is done by setting the TWSTA bit of the TWCR register to 1. The start condition will start transmitting data once the TWINT bit of TWCR register is cleared i.e. set to 1 (TWINT Flag is set to 1 when data is ready for transmission and no ongoing transmission is there)

- When the START condition has been transmitted, the TWINT flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.

- Once the start condition has been successfully transmitted (i.e. wait till TWINT flag sets to one and check status of TWSR), the address of the slave is sent.After that, the TWINT flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent.

- The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted and the slave sends an acknowledgement bit back. After slave address, address of register is sent from which you want to read data (accelerometer, gyroscope etc.). After that check whether acknowledgement bit is received, if yes proceed ahead and if not either stop communication and start again or transit repeated start condition and send slave address again.

- After this we send the SLA+R OR SLA+W depending on whether we want the read or write operation to be done.The master and slave start the transmission of data using the TWDR register.

- When the data packet has been transmitted, the TWINT flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent.The status code will also reflect whether a slave acknowledged the packet or not.

- The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition which is done by setting the TWSTO bit of TWCR register to 1.

- Note that TWINT is NOT set after a STOP condition has been sent.

Some important configurations before starting communication:

- Access power management register (you can search it in data register mapping sheet of MPU6050) so that MPU comes out of sleep mode.
- Select gyroscope and accelerometer outputs using FS_SEL and AFS_SEL registers respectively in Register Mapping Datasheet of MPU6050.
- Similarly by accessing $I2C\_MST\_CTRL$ register having hex address 24 you can set speed of I2C bus

For more information refer MPU datasheet and MPU Register Mapping.

Calculating ROLL and PITCH

Using Accelerometer:

$$acc\_roll = \tan^{-1}\left(\frac{acc\_y[0]}{\sqrt{(acc\_z[0])^2 + (acc\_x[0])^2}}\right) * (180/\pi);$$

$$acc\_pitch = -\tan^{-1}\left(\frac{acc\_x[0]}{\sqrt{(acc\_z[0])^2 + (acc\_y[0])^2}}\right) * (180/\pi);$$

(acc_roll and acc_pitch in degrees)

Using Gyroscope:

$$gyro\_roll = gyro\_x * t_{sample};$$

$$gyro\_pitch = gyro\_y * t_{sample};$$

(t_{sample}:sample time)

Compilimentary Filters

Problem with Accelerometer

As an accelerometer measures all forces that are working on the object, it will also see a lot more than just the gravity vector. Every small force working on the object will disturb our measurement completely. If we are working on an actuated system (like the Quadcopter), then the forces that drive the system will be visible on the sensor as well. The accelerometer data is reliable only on the long term, so a "low pass" filter has to be used.

Problem with Gyroscope

Because of the integration over time, the measurement has the tendency to drift, not returning to zero when the system went back to its original position. The gyroscope data is reliable only on the short term, as it starts to drift on the long term; hence high pass filter has to be used.

Mathematical Equation

$$Transferfunctionof LOWpassfilter(T_{lpf}) = \frac{w}{s+w}$$

$$Transferfunctionof HIGHpassfilter(T_{hpf}) = \frac{s}{s+w}$$

Where $w = \frac{1}{2*\pi*R*C}$,s is used for frequency domain using Laplace transform.

As are signals are discrete these can be converted to discrete domain using

$$S = 1 - inv(z)/T$$

where T is sampling time of code, $inv(z)$ is inverse of z or $1/z$.

$$Roll = acc\_roll * T_{lpf} + gyro\_roll * T_{hpf}$$

.

$$Pitch = acc\_pitch * T_{lpf} + gyro\_pitch * T_{hpf}$$

Problem Faced

- Even using complimentary filter readings from MPU-6050 were not accurate so we tried to implement KALMAN FILTER algorithms in which predictions were made through mathematical model and second measurement was taken through MPU6050 sensor and based on above measurements above actual state is predicted. We are currently working on this algorithm as we are facing some issues regarding it in SIMULINK.
- Another approach was to take 500 around readings during initial state and average it to calculate drift. This amount was subtracted from other readings to obtain accurate readings. Accuracy has been increased to certain extinct. We are looking for other methods to increase our accuracy to almost 100 percent.

Hardware

- Arduino Mega 2560
- RF Receiver/Transmitter (Signal)
- Electronic Speed Controller (ESC)
- BLDC Motors
- LIPO
- Mpu6050

Arduino Mega 2560

Technical specifications

Micro controller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the boot loader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library)

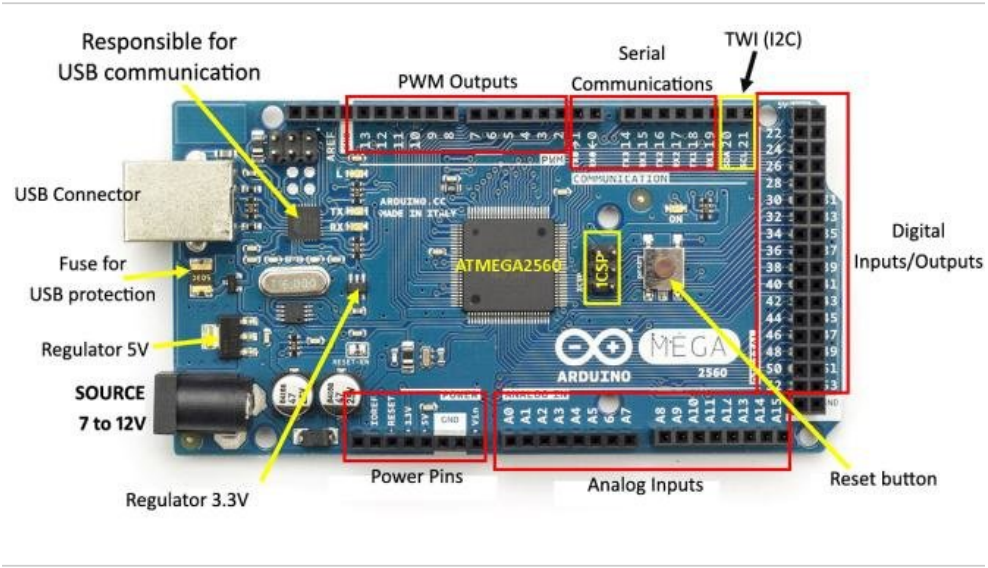
Pins

External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the attachInterrupt() function for details.

PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function

- Pin 4 and 13, controlled by timer0: 16 MHz / 64 / 256 = 976.56 Hz
- Other PWM pins, controlled by timer1-4: 16 MHz / 64 / 510 = 490.20 Hz.

TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the old Duemilanove or Diecimila Arduino boards.



2 FS-i6 Transmitter and Receiver

Specification of Receiver

1. i-Bus Interface:Yes.
2. Data Acquisition Interface:Yes.
3. Weight:392 gm.
4. Power:6V (1.5V AA*4).
5. Antenna Length:26mm * 2 (dual antenna).

6. Transmitting Power: 20dBm.
7. RF Receiver Sensitivity:-105dbm.

Specification of Transmitter

No. of Channels	6.
RF Range (GHz)	2.40 ~ 2.48
Bandwidth (KHz)	500
RF Power (dBm)	≤20
Sensitivity	1024
Low Voltage Warning (V)	≤ 4.2
Power	6V (4 x 1.5AA Not Included)
Length (mm)	174
Width (mm)	190
Height (mm)	40
Weight (gm)	400
Color	Black

Binding Fly-sky receiver

The process of binding your receiver to your radio essentially just pairs them so that the radio and receiver know that they are talking to each other, useful when flying with others. The basic principle for all fly-sky receivers is the same, but some specifics might differ from model to model. Some receivers use a bind button, Others use a bind plug, so take note for your specific receiver.

- Turn your radio on while holding the bind button, once started the screen should read RX binding.
- Now turn on your receiver while holding in the bind button
- Once complete you should see the message Rx Bind OK display for a second.

Electronic Speed Controller (ESC)

An Electronic speed controller is an electronic circuit that acts as the interface between the pilots commands and the individual drone motors there are several types of ESCs in the market ,but

brushless motors require a 3-phase ESC these are easily distinguished by the presence of three soldering pads that are meant to connect to the three motor phases of a brush less motor.

Features

- This 30 amp ESC for BLDC has on board BEC.
- This ESC comes with a male dean T-connector.
- Auto Low BATTERY Slow down at 3.0V/cell Lipo, cut-off at 2.9V/cell lipo.
- APPLICATION: Multirotors, Rc Airplanes, etc.

BLDC(Brushless DC) Motors

Why Do BLDC Motors Turn?

As their name implies, brushless DC motors do not use brushes. With brushed motors, the brushes deliver current through the commutator into the coils on the rotor. So how Does a brushless motor pass current to the rotor coils? It doesntbecause the coils are not located on the

rotor. Instead, the rotor is a permanent magnet; the coils do not rotate, but are instead fixed in place on the stator. Because the coils do not move, there is no need for brushes and a commutator.

With the brushed motor, rotation is achieved by controlling the magnetic fields generated by the coils on the rotor, while the magnetic field generated by the stationary magnets remain fixed. To change the rotation speed, you change the voltage for the coils. With a BLDC motor, it is the permanent magnet that rotates; rotation is achieved by changing the direction of the magnetic fields generated by the surrounding stationary coils. To control the rotation, you adjust the magnitude and direction of the current into these coils.

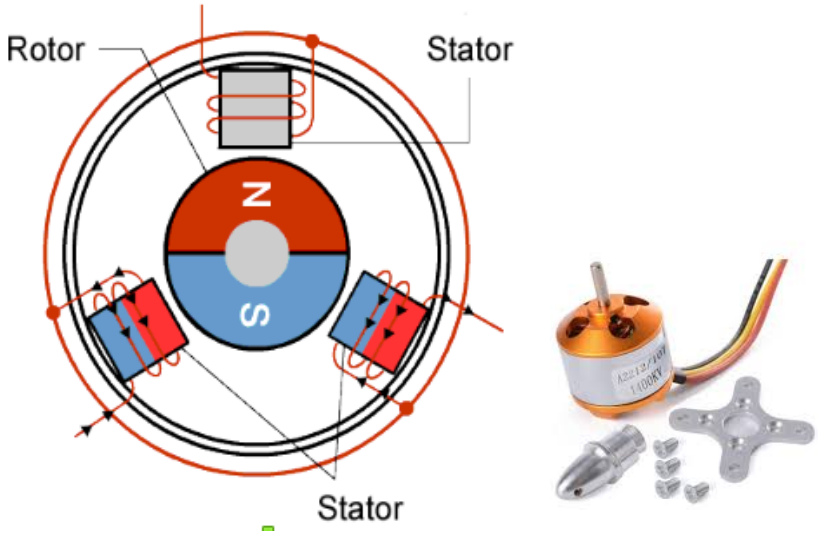
Advantages of BLDC Motors

A BLDC motor with three coils on the stator will have six electrical wires (two to each coil) extending from these coils. In most implementations three of these wires will be connected internally, with the three remaining wires extending from the motor body (in contrast to the two wires extending from the brushed motor described earlier). Wiring in the BLDC motor case is more complicated than simply connecting the power cells positive and negative terminals; we will look more closely at how these motors work in the second session of this series. Below, we conclude by looking at the advantages of BLDC motors.

One big advantage is efficiency, as these motors can control continuously at maximum rotational force (torque). Brushed motors, in contrast, reach maximum torque at only certain points in the rotation. For a brushed motor to deliver the same torque as a brushless model, it would need to use larger magnets. This is why even small BLDC motors can deliver considerable power.

The second big advantagerelated to the firstis controllability. BLDC motors can be controlled, using feedback mechanisms, to delivery precisely the desired torque and rotation speed. Precision control in turn reduces energy consumption and heat generation, andin cases where motors are battery poweredlengthens the battery life.

BLDC motors also offer high durability and low electric noise generation, thanks to the lack of brushes. With brushed motors, the brushes and commutator wear down as a result of continuous moving contact, and also produce sparks where contact is made. Electrical noise, in particular, is the result of the strong sparks that tend to occur at the areas where the brushes pass over the gaps in the commutator.



LIPO(Lithium Polymeric Battery)

Specifications

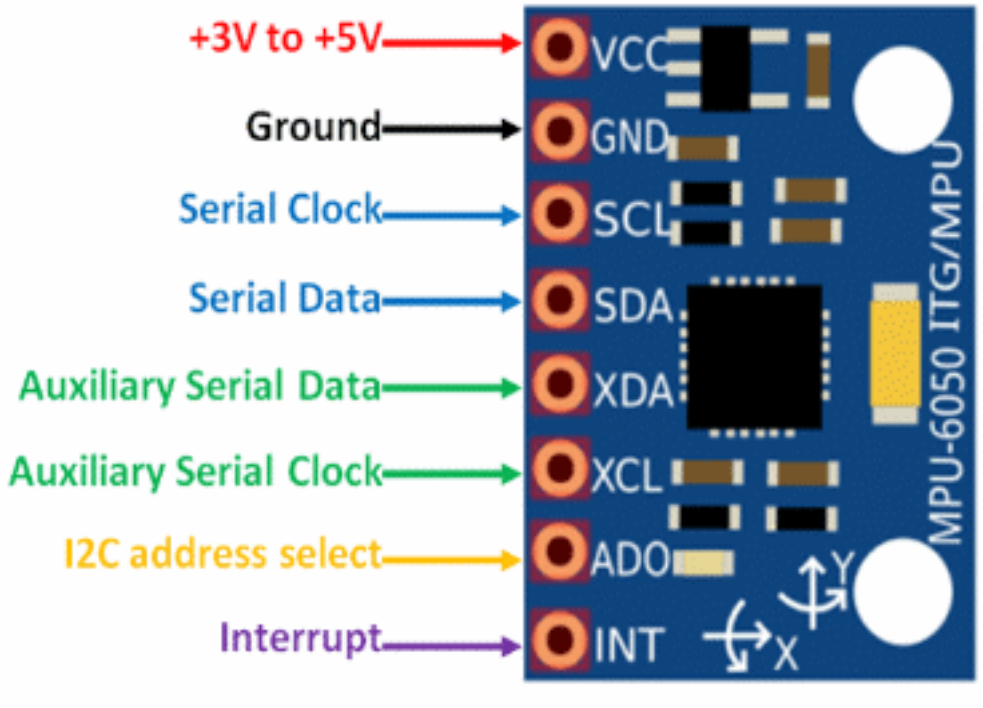
- Model No: ORANGE 2200/3S-30C
- Weight : 175.0g
- Voltage : 11.1V
- Dimensions : 23x34x106(mm)
- Max Continuous Discharge : 30C(66.0A)
- Balance Plug : JST-XH
- Max Burst Discharge : 60C(132.0A)
- Discharge Plug : XT-60
- Charge Rate : 1-3C Recommended, 5C Max

MPU6050

Introduction

- MPU6050 is a Micro Electro-mechanical system (MEMS), it consists of three-axis accelerometer and three-axis gyroscope. It helps us to measure velocity, orientation, acceleration, displacement and other motion like features.
- MPU6050 consists of Digital Motion Processor (DMP), which has property to solve complex calculations.
- MPU6050 consists of a 16-bit analog to digital converter hardware. Due to this feature, it captures three-dimension motion at the same time.
- This module has some famous features which are easily accessible, due to its easy availability it can be used with a famous microcontroller like Arduino.
- This module uses the I2C module for interfacing with Arduino.
- MPU6050 is less expensive, Its main feature is that it can easily combine with an accelerometer and gyro.

MPU6050 Pin Configuration



Pin	Pin Name	Description
01	Vcc	This pin used for Supply Voltage. Its input voltage is +3 to +5V.
02	GND	This pin use for ground
03	SCL	This pin is used for clock pulse for I2C compunction
04	SDA	This pin is used for transferring of data through I2C communication.
05	Auxiliary Serial Data (XDA)	It can be used for other interfaced other I2C module with MPU6050.
06	Auxiliary Serial Clock (XCL)	It can also be used for other interfaced other I2C module with MPU6050.
07	AD0	If more than one MPU6050 is used a single MCU, then this pin can be used to vary the address.
08	interrupt (int)	This pin is used to indicate that data is available for MCU to read.

MPU6050 Features

- MEMS 3-axis accelerometer and 3-axis gyroscope values combined.
- Power Supply: 3-5V
- Communication : I2C protocol
- Built-in 16-bit ADC provides high accuracy
- Built-in DMP provides high computational power
- Can be used to interface with other IIC devices like magnetometer

- Configurable IIC Address
- In-built Temperature sensor

Matlab Simulink

MATLAB stands for MATrix LABoratory,since the basic building block here is the MATRIX. MATLAB is basically a software package for high-performance numerical computation and visualization. It provides an interactive environment with hundreds of built - in functions for technical computation, graphics and animation. Best of all, it also provides easy extensibility with its own high level programming language.

Its built in functions provide excellent tools for

- Linear algebra computations
- Data analysis
- Signal processing
- Optimization
- Numerical solution of Ordinary Differential Equations ((ODE).
- Quadrotors and many other types of scientific computations.

Simulink is one of many tools that are provided with Matlab.Simulink, developed by MathWorks, is a graphical programming environment for modeling, simulating and analyzing multi domain dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries.Simulink is widely used in automatic control and digital signal processing for multidomain simulation and Model-Based Design.

It is easy to program an arduino from simulink. We need to download and install Simulink Support Package for Arduino Hardware for the Simulink library . Once we have it, just search for arduino common blocks from the sim library where we will find all the basic blocks like digitalwrite , read etc., and can easily connect those blocks and do Arduino programming in minutes.

Toolboxes such as CONTROL SYSTEM TOOLBOX, CURVE FITTING TOOLBOX, PID TUNER are used to design the control system of the quadcopter. Whole control system discussed in this documentation is designed and simulated on Simulink. Through Simulink, we deployed our code in the arduino hardware.

Future Aspects

- Using STM Microcontrollers and Raspberry Pi
- Trajectory planning
- Delivery of Medicines
- Swarm Robotics

Reference Links

- Matlab Documentation (or) https://in.mathworks.com/help/index.html
- MPU6050 Datasheet (or) https://pdf1.alldatasheet.com/datasheet-pdf/download/1132807/TDK/MPU-6050.html
- https://www.arduino.cc/
- https://in.mathworks.com/help/supportpkg/arduino/examples/getting-started-with-arduino-har html
- BLDC Principle
- https://www.tomshardware.com/reviews/multi-rotor-quadcopter-fpv,3828-2.html
- https://www.seeedstudio.com/blog/2020/01/17/what-is-imu-sensor-overview-with-arduino-usag