Quadcopter Dynamics

DigitalLab Course Project
EP 315

Guide: Prof. Pradeep Sarin

Department of Physics
Indian Institute of Technology, Bombay

Team Members

1. Abhishek Ukey (110260006)
2. Ankur Agrawal (11D260003)
3. Digvijay Wadekar (110260016)
4. Hardik Godara (110260011)
5. Jash Banker (110260003)
6. Mukul Sholapurkar (110260008)
7. Tushar Sinha (110260019)

Acknowledgments

This report was written as part of the Course Project in the Arduino Electronics Lab offered by the Physics department, and taught by Professor Pradeep Sarin. We are extremely grateful to him for his guidance and insight. We would also like to thank Swaroop Hangal for providing us with the technical help related to simulation testing.

Contents

1 Introduction **5**

2 Motivation **5**

3 Preliminary Approach **5**

4 Hardware **6**

4.1 Frame . . . 6
4.2 Motors . . . 6
4.3 Propellers . . . 7
4.4 ESC – Electronic Speed Controller . . . 8
4.5 Battery . . . 9
4.6 IMU (Inertial Measurement Unit) . . . 9
4.6.1 Components of the IMU . . . 9
4.6.2 Interface . . . 10
4.6.3 Arduino 10DOF with 3axis Gyro, 3 axis Accel, 3 Axis Compass and Barometer . . . 10
4.7 Shield . . . 11

5 Flight Dynamics **11**

5.1 Four Inputs . . . 11
5.2 Six Output States . . . 12
5.3 Overall Propeller Speed . . . 12

6 Modules **13**

6.1 Kinematics . . . 13
6.2 Receiver-Transmitter . . . 14
6.2.1 Obtaining Signal from Receiver . . . 14
6.2.2 Output to Servos . . . 15
6.3 Filter . . . 16
6.3.1 Converting the data to angles and filtering . . . 16
6.3.2 Inputs to this block . . . 16
6.3.3 Important Functions . . . 16
6.4 First PID . . . 18
6.5 Second PID . . . 19

7 Cost Estimate **20**

8 Work Division **20**

9 Remaining Work **21**

1 Introduction

A quadcopter, also called a quadrotor, is basically a helicopter with 4 rotors instead of the 2 used in a standard helicopter. Quadcopters have gained tremendous popularity in the field of "Unmanned Autonomous Vehicle (UAV)" research. Quadcopters depend on electronic sensors and control systems to stabilize their flight. The main advantage that a quadcopter has over a helicopter is that since it requires four small rotors, each rotor possesses less kinetic energy and hence will cause less damage if it hits something. Each motor must lift only a quarter of the weight so we can use cheaper motors. Also quadcopters are easier to build and control and due to their small size, can be maneuvered easily in closed spaces too.

2 Motivation

We realize that the quadcopter is a well-researched topic and models like the one we wish to design have already been implemented. So we intend this project to be solely for educational purposes. We believe that we will gain tremendous knowledge about micro-controllers and control systems via this project. We will also learn a lot about flight dynamics and aero-modelling. Last but not the least, it is extremely exciting to build a device that can fly! Also we realize that once the quadcopter has taken flight, there are a number of uses that we can put it too.

3 Preliminary Approach

In this section, we will discuss the mechanics of flight control. The quadcopter's movement is controlled by varying the relative thrusts of each of the 4 rotors. These rotors are aligned in a diamond. Motors on one diagonal rotate in the clockwise direction and on the other in the anti-clockwise direction.

- Moving forward

  To move forward we reduce the power in the front motor. This tilts the quadcopter forward and the rotors provide sufficient thrust to move forward. Speed is decided by the power given to the rotors.

- Moving back

  To move backwards we reduce the power in the back rotor, tilting the quadcopter backwards and then again the thrust from the motors takes it in reverse.

  In addition to this, the quadcopter has 3 degrees of freedom namely:

1. Yaw

   Yaw (turning left and right) is controlled by turning up the speed of the regular rotating motors and taking away power from the counter rotating; by taking away the same amount that you put in on the regular rotors produces no extra lift (it won't go higher) but since the counter torque is now less, the quadcopter rotates and control becomes a matter of which motor gets more power and which one gets less.

2. Roll

   Roll (tilting left and right) is controlled by increasing speed on one motor and lowering on the opposite one.

3. Pitch

   Pitch (moving up and down, similar to nodding) is controlled the same way as roll, but using the second set of motors.

4 Hardware

4.1 Frame

A frame is a structure that holds all the components together. It should be rigid, and be able to minimize the vibrations induced by the motors. Our quadcopter frame consists of three parts:

- The centre plate where the electronics and batteries are mounted
- Four arms connecting to the centre plate
- Four motor brackets attaching the motors to the end of the arms

We have chosen carbon fibre frame since its beginner friendly and due to the small size of the frame, the cost is also reasonable.

4.2 Motors

We have used brushless motors. Ordinary DC motors have coils and magnets which are used to drive the shaft. They have a brush on the shaft which takes care of switching the power direction in the coils. Brushless motors don't have this brush. Instead they have coils on the inner side (centre) of the motor, which is fixed to the mounting.

On the outer side, they contain a number of magnets mounted to a cylinder that is attached to the rotating shaft. So, the coils are fixed which means wires can go directly

Figure 1: Types of frames

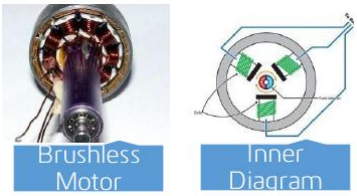

Figure 2: Types of motors

to them and therefore there is no need for a brush. They spin at a much higher speed and use less power than DC motors (at same speed). Also there is no power loss due to brush transition.

Brushless motors come with a Kv-rating which indicates how many RPMs (Revolutions per minute) the motor will do if provided with V volts under no load. The RPMs can be calculated in this way: RPM=$Kv \times V$

We are using 1240 Kv with max power of 150 watts. These motors can provide a thrust of 400 gm at 10 Amps and upto a maximam of 700 gm at 16 Amps. Thrush of about 250 gm (from each motors) would be sufficient for hovering.

4.3 Propellers

On each of the brushless motors there is mounted a propeller. In a quadcopter, propellers have opposite tilts. One is for clockwise motion and one for anti-clockwise. This makes the yaw angle stable. Propellers come in various sizes and pitch. We plan to use either 1045 (10 diameter and 4.5 pitch) or 8045 (8 diameter and 4.5 pitch). These are the most commonly used propellers for mid-sized quadcopters.

The diameter gives the area and the pitch gives effective area. With same diameter and larger pitch the propeller would generate more thrust and lift more weight but requires more power. A higher RPM of the propeller gives more speed and manoeuvrability but lifts less amount of weight. The power drawn by the motor increases as the effective area

Figure 3: ESC

of the propeller increases. At bigger diameter or higher pitch one will draw more power at the same RPM, but also produces much more thrust and lift more weight. In choosing a balanced motor and propeller combination, you have to figure out what you want your quadcopter to do. If you want to fly around stably with heavy objects like a camera, you would probably use a motor that manages lesser revolutions but can provide more torque and a longer or higher pitched propeller (which uses more torque to move more air in order to create lift). For a quadcopter to fly we need 1:2 ratio for weight and thrust.

$Power(watts) = K_p * D^4 * P * RPM^3$

Here D is diameter, P is pitch and K_p for mid sized propellers are around 1.2

4.4 ESC – Electronic Speed Controller

The brushless motors are normally 3 phased, so direct supply of DC power will not turn the motors on. Electronic Speed Controllers generate three high frequency signals, with different but controllable phases, continuously to keep the motor turning.

ESC has a battery input and a three phase output for the motor. Each ESC is controlled independently by a PPM signal (similar to PWM). The frequency of the signals vary a lot, but for a Quadcopter it is recommended that the controller should support high frequency signal, so the motor speeds can be adjusted quick enough for optimal stability (i.e. at least 200 Hz or even better 300 Hz PPM signal). We are using 30 Amp ESCs with PWM control.

4.5 Battery

In Quad rotors, Lithium Polymer (LiPo) Battery is the most commonly used Power Source because of its light weight and its high current rating. NiMH Battery is a cheaper alternative but is much heavier than LiPo Battery. A single LiPo cell can provide a voltage of upto 3.7 volt.

A LiPo battery has two characteristic parameters:

1. Capacity

   It is measure of how much energy is stored in battery. It is measured in mAh (Amp hour). A battery with capacity of 4000 mAh can power a 0.8 kg Quad rotor for 5 minutes of full throttle and 20 min of hovering.

2. Discharge rate

   This is the rare at which battery can discharge. It is also called C-rate and expressed in C units. The maximum current that can be drawn from a battery is simply product of Discharge rate and Capacity. A 4000mAh 30C 3S LiPo can give up to 120 Amps of maximum current

The specifications of the battery we are using are as follows -
Capacity: 4000mAh
Voltage: 3 Cell / 11.1V
Discharge: 25C

4.6 IMU (Inertial Measurement Unit)

The IMU is an electronic sensor device which measures the velocity, orientation and acceleration along different directions. This sensor allows the control system to navigate the bot in the environment. The readings of the IMU are fed to the main controller which are then compared with the set points and then appropriate action is taken by the motor controller system. The IMU is a combination of a 3-axis gyroscope and a 3-axis accelerometer, which together makes it a 6 degree of freedom sensor. Sometimes a 3-axis magnetometer is also included to get an absolute yaw control relative to the Earth's magnetic field. This makes the IMU a 9 degree of freedom sensor.

4.6.1 Components of the IMU

1. Accelerometer

   The accelerometer measures the acceleration relative to the gravitational force. So, the 3-axis accelerometer basically gives us components of acceleration in all the 3

directions. It can be used to measure the orientation, vibration and shock and hence is critical for the stability of the Quad copter (in our case). The disadvantage of only using the accelerometer is that it may become extremely sensitive to unwanted vibrations and noise (for example motor vibrations) and may lead to instability of the bot.

2. Gyroscope

   A gyroscope is a device which measures and maintains the orientation based on the principle of angular momentum. It measures the angular velocity i.e., how fast something is spinning about an axis as well as rotational acceleration. Unlike accelerometers gyroscopes are not affected by gravity, so they make a great complement to each other. Accelerometer measure acceleration **along** the specified axes whereas Gyroscopes measure acceleration **about** the axes.

3. Magnetometer

   A magnetometer measures the strength and the direction of magnetic fields and hence can be used to determine the orientation of the bot with respect to Earth's magnetic field. This helps in additional Yaw stability.

4.6.2 Interface

Interfacing an IMU is one of the critical tasks since it depends on the availability of embedded protocols in the respective microcontroller. Most of the IMUs available in the market give output data via an analog, serial or I2C (a multi-master two wire serial bus) interface.

Graphical representation of the output data and trade-offs between the components of the IMU are shown in figure(4)

4.6.3 Arduino 10DOF with 3axis Gyro, 3 axis Accel, 3 Axis Compass and Barometer

We are using a 10 DOF sensor breakout board to make our own IMU. Arduino 10DOF contains ITG-3205 (gyroscope), ADXL345 (accelerometer), HMC5883L(magnetic compass), and BMP085 (Barometer). We have written our own algorithm to get fused data from Arduino 10DOF. We are also using magnetometer for additional yaw stability. Interfacing is done using I2C communications standards with help of wire library on arduino.

Figure 4: In the above graph, the green line corresponds to the actual kinematical measurements of the device whereas the blue and red lines correspond to the measurements given by the gyroscope and the accelerometer respectively. We clearly see that the gyroscope readings are less prone to the noise but may drift from the actual data by some offset. The accelerometer readings don't drift much but are quite sensitive to unwanted vibrations and noise.

4.7 Shield

Shield is a PWB(Printed Wiring Board) which is used to mechanically support and electrically connect electronic components using conductive pathways(copper tracks).

It consists of a non-conducting substrate, printed copper tracks and holes for attaching the Arduino and other components (using screws & nuts). The substrate provides enduring firmness to the connections involving the Arduino which is being stacked over by other components. The motive being to make the whole body detachable and compact, which helps while debugging and thus, ultimately saving time.

5 Flight Dynamics

5.1 Four Inputs

1. $U_1 \rightarrow$ Affects Altitude

2. $U_2 \rightarrow$ Affects a Rotation in Roll Angles

3. $U_3 \rightarrow$ Affects Pitch Angle

4. $U_4 \rightarrow$ Affects Yaw Angle

5.2 Six Output States

$X,\ Y,\ Z,\ \Theta(Roll),\ \Psi(Pitch),\ \Phi(Yaw)$

5.3 Overall Propeller Speed

1. Throttle:
$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{1}$$

2. Roll:
$$U_2 = bl(\Omega_4^2 - \Omega_2^2) \tag{2}$$

3. Pitch:
$$U_2 = bl(\Omega_3^2 - \Omega_1^2) \tag{3}$$

4. Yaw:
$$U_4 = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \tag{4}$$

On transformation using inverted Matrix Method

1.
$$\Omega_2^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_3 - \frac{1}{4d}U_4 \tag{5}$$

2.
$$\Omega_2^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \tag{6}$$

3.
$$\Omega_3^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_2 - \frac{1}{4d}U_4 \tag{7}$$

4.
$$\Omega_4^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \tag{8}$$

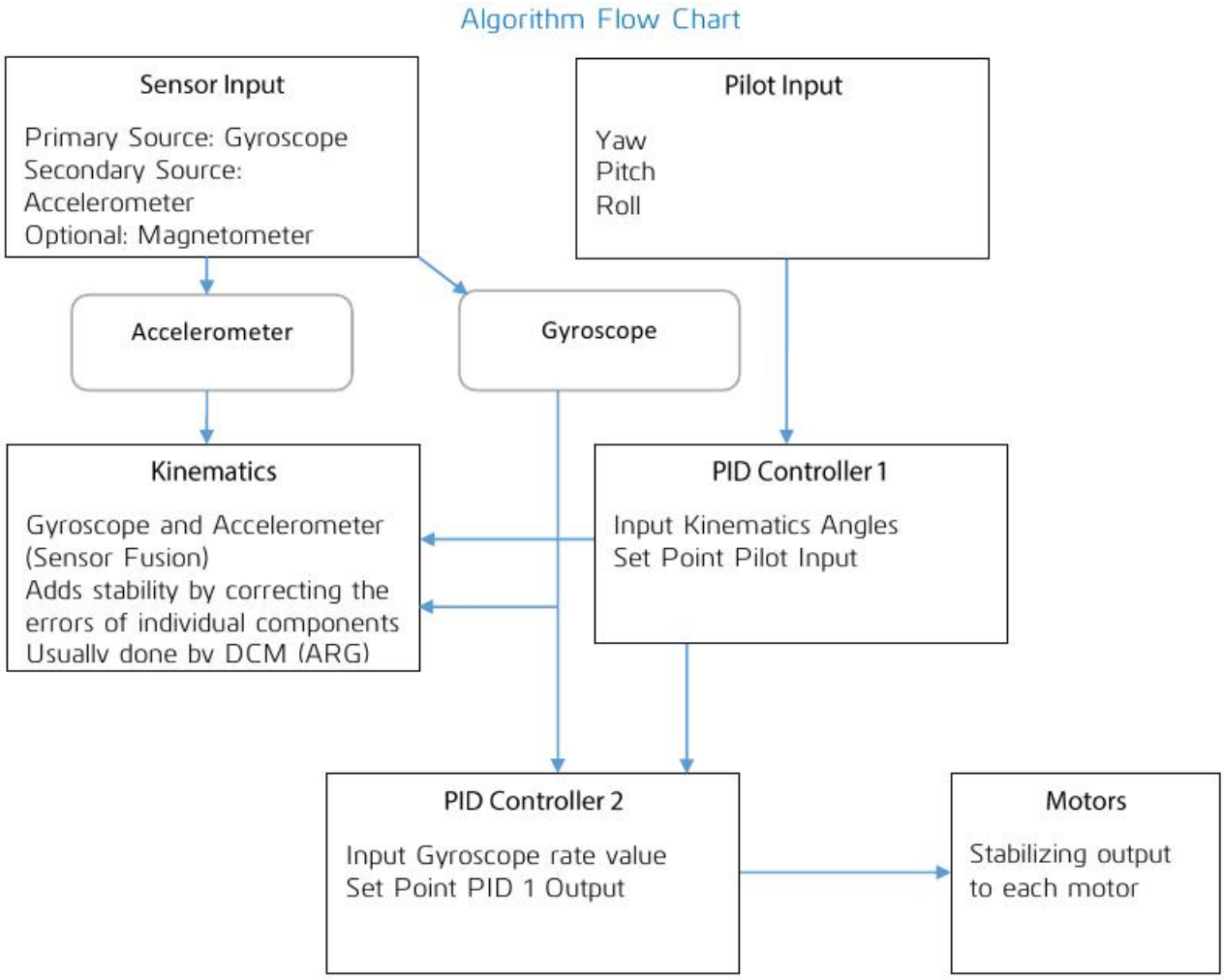

Figure 5:

6 Modules

6.1 Kinematics

Right now we have raw gyroscope data and raw accelerometer data, but neither one of these sensor outputs give us an accurate enough estimate to be used in our stabilization algorithm.

What we will do, is combine accelerometer and gyroscope outputs via complementary filters. Output from our kinematics will feature strongly suppressed noise from the accelerometer along with the gyroscope output without the "drift".

6.2 Receiver-Transmitter

6.2.1 Obtaining Signal from Receiver

This algorithm along is illustrated in the flow chart given in figure (6)

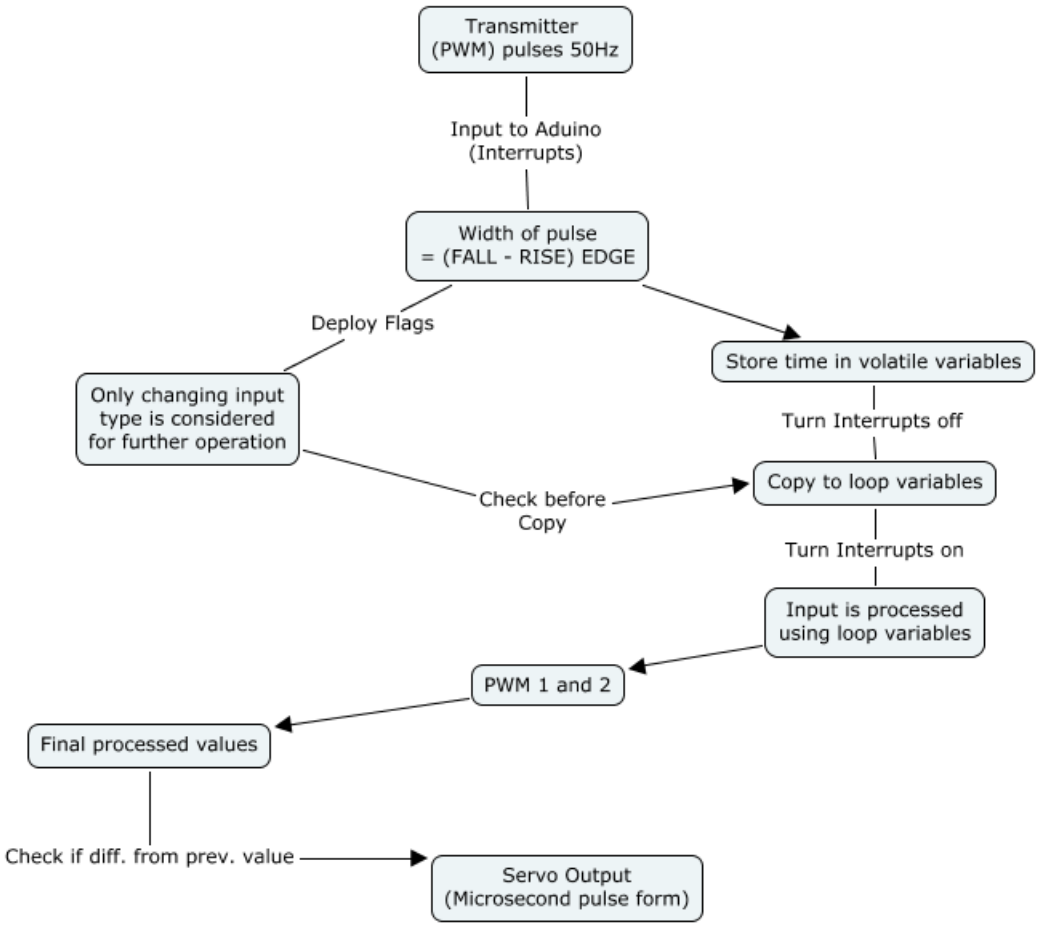

Figure 6: This is the complete block diagram for inputs from Transmitter and output to Servos

The output from Transmitter is of PWM format where the length of pulse width determines the power output to the motors (shown in figure[7]).

The motors are powered through the ESCs using PWM signal which can be generated from the Arduino board using the Servo library. A servo expects a pulse of between 1 and 2 milliseconds to be sent about every 20 milliseconds.

We need to measure the width of input pulses from the receiver using interrupts along with the **PinchangeInt** library [10]. Reasons for choosing this library are enumerated as-

1. By using the conventional functions like pulseln() or normal interrupts. We need to

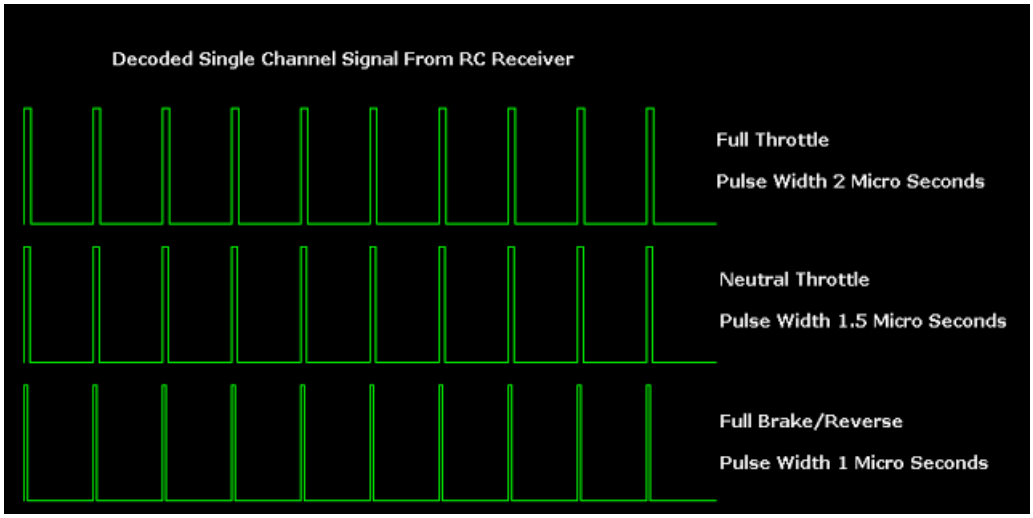

Figure 7: Types of signals obtained from Radio Receiver

stop receiving signals from the transmitter for atleast 2 milli second duration and thus, we have wasted computational power equivalent to 32,000 operations on an Arduino. [1]

2. Provides individual interrupt handler functions for upto 21 external interrupts and we are able to use the timer function directly in the interrupt block

3. It provides the additional functionality of pin number and state to be available for access inside the interrupt handler

4. The Servo library for the Arduino UNO can be used for controlling upto 12 Servos at a time on a single Arduino UNO.

6.2.2 Output to Servos

We have to take into accont the following precautions before giving output to servo motors:

1. Shared variables are updated by the ISR and read by loop. Before execution of the main program loop we must immediately take local copies so that the ISR can keep ownership of the shared ones and stop the interrupts[2] during this copying process and immediately switch them back on so we are able to receive new signals.

[1] It gets worse for pulseln() because you will have to wait a whole 20 milliseconds for the next pulse to arrive and complete. Thats a full 320,000 operations useful operations your code could have completed!!

[2] If the interrupts are switched on during this time, some bites could change during copying process causing the copied value to have parts of old and new variables

2. Shared variables should be kept volatile because they are used both in main loop and ISR. The local values can be kept static and only they should be used in main code processing. [3]

3. Updating a servo output continuously can consume system time and power so, we have deployed appropriate flags at transmitter inputs which change only when the incoming input varies. Thus only the changed input values are copied to servos (or used for processing in the main code)

4. The copying of the inputs to the servos will consume system time. So, we will only start copying when the relevant input is given and rest of the time the copying is stopped. This will increase the system efficiency manifold.

6.3 Filter

6.3.1 Converting the data to angles and filtering

We set a global frame comprising of the magnetic North, West and Zenith as the 3 axes. This is illustrated below

6.3.2 Inputs to this block

1. Magnetometer

   Provides the 3 element vector giving the (x, y, z) coordinates of global North in the body frame

2. Accelerometer

   3 element vector giving the (x,y,z) components of the acceleration in the body frame with the assumption that when hovering the reading is +g along z.

3. Gyroscope

   Angular velocity components (x,y,z) of the rotation of the global frame as viewed in body frame.

6.3.3 Important Functions

1. Obtain North from magnetometer

   North is obtained directly from magnetometer input reading

[3] Shared copies could be updated anytime

2. Obtain Zenith from accelerometer

   To obtain Zenith we make the assumption that the quadcopter acceleration is always along the body 'z'. Thus any components along the 'x' and 'y' axes are due to gravity and this enables us to obtain the orientation of the vertical (Zenith) of the global frame.

3. Obtain North from gyroscope

   We know the angular velocity of the zenith in the body frame. This can be used to obtain the x, y and z velocities of the Zenith vector in the body frame. Integrating this one can obtain the coordinates of the zenith vector in the body frame.

4. Obtain West from gyroscope

   Follow the exact same procedure as above

5. Obtain Thurst

   Subtracting out the gravity vector from the vector formed by the accelerometer data, one can get the thrust acting along the body 'z' axis.

6. Filter the data

   We use a simple complimentary filter which is a weighted average of the two inputs. We use it to filter the north obtained from the gyroscope and from the magnetometer as well as the zenith obtained from the accelerometer and the gyroscope.

7. Obtain West

   This is done by simply taking a cross product of the filtered North and Zenith directions.

8. Correct North

   It may be possible that our North, West and Zenith may not be perfectly orthogonal. So we recalculate the North direction by taking a cross product of West and Zenith.

   At this point we have obtained the axes of the **global frame** in the **body frame**. The next step is to normalize all the vectors.

   We then note that one can body 'z' axis by taking the 'z' components of the above obtained North, West and Zenith vectors and similarly for the other body axes.

9. Get the pitch angle

10. Get the roll angle

    This algorithm along is illustrated in the flow chart given in figure (8)

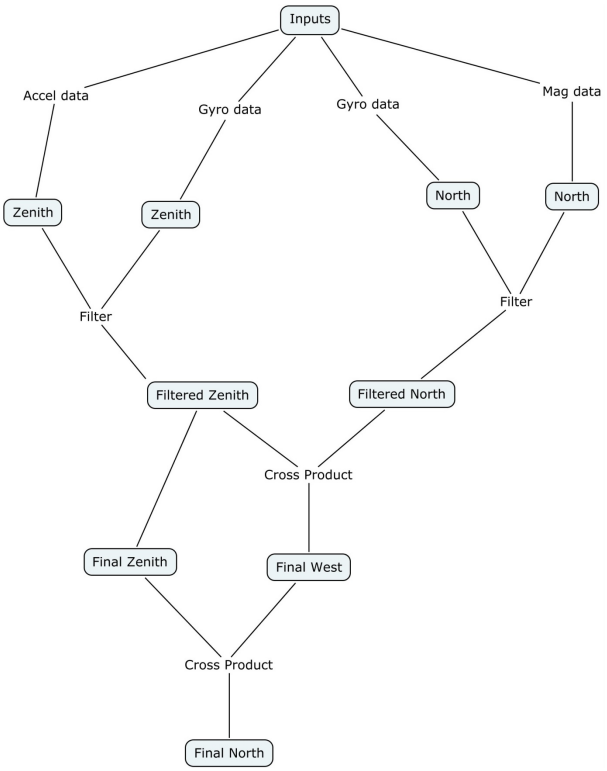

Figure 8: This is the block diagram for

6.4 First PID

From the diagram below you can see that our first PID controller will take output from our pilot as "Set Point" and kinematics (containing current estimation of yaw, pitch and roll angles) as input.

Output from our first PID controller will contain the angle desired by pilot +- current kinematics angle. This acts like an "accelerant" for the second PID.

In this case "accelerant" means that the value from our first PID controller will determine how "fast" we want to correct for the current stabilization error.

We are directly controlling Thrust and Yaw angular velocity but for pitch and roll we need to have a control on angle as piloting by giving angular velocity is a very difficult. To do this we implement control of roll and pitch in two steps.

There are two PID loops running here:

1. Roll (angle) PID

   Input - Roll angle from filter

   Setpoint - Roll angle from Receiver

   Output - Setpoint for Roll(velocity) PID

2. Pitch (angle) PID

Input - Pitch angle from filter

Setpoint - Pitch angle from Receiver

Output - Setpoint for Pitch(velocity) PID

6.5 Second PID

The second PID controller takes the "accelerant" from the first PID and Receiver as "Set Point" and current gyroscope output (gyro Rate) and Thrust from filter as input.

The resulting output from the second PID controller is the decimal value representing force that has to be applied to each of the axes to correct for the stabilization error.

In our case this force is generated by spinning propellers, and can be controlled by adjusting the speed of the rotating propellers.

There are four PID running controlling one degree of freedom each.

1. Roll (velocity) PID

   Input - Roll velocity from Gyroscope

   Setpoint - Output of Roll(angle) PID

   Output - U2 (PID sum controlling Roll)

2. Pitch (Velocity) PID

   Input - Pitch velocity from Gyroscope

   Setpoint - Output of Pitch(angle) PID

   Output - U3 (PID sum controlling Pitch)

3. Thrust PID

   Input - Thrust from Filter

   Setpoint - Thrust from receiver

   Output - U1 (PID sum controlling Thrust)

4. Yaw (velocity) PID

   Input - Yaw velocity from Gyroscope

   Setpoint - Yaw angular velocity from Receiver

   Output - U4 (PID sum controlling Yaw)

7 Cost Estimate

Description	Quantity	Unit Price (USD)	Total Price (USD)
10x4.5 SF Props 2pc Standard Rotation/2 pc RH Rotation (Black)	2	3.15	6.3
8045 SF Props 2pc Standard Rotation/2 pc RH Rotation (Black)	4	2.79	11.16
Arduino 10DOF with ITG-3205, ADXL345, HMC5883L, and BMP085	1	28.63	28.63
Turnigy Talon Carbon Fiber Quadcopter Frame*JIT-Item (this item takes 3-5 days to dispatch)	1	29.99	29.99
Arduino MEGA ProtoShield V3 Expansion Board	1	5.6	5.6
Turnigy Aerodrive SK3 - 2826-1240kv Brushless Outrunner Motor	4	16.41	65.64
TURNIGY Plush 30amp Speed Controller	4	13.13	52.52
HXT 4mm to 6 X 3.5mm bullet Multistar ESC Power Breakout Cable	1	4.8	4.8
Turnigy 6X FHSS 2.4ghz Transmitter and Reciever (Mode 1)	1	29.9	29.9
Turnigy nano-tech 4000mah 3S 25~50C Lipo Pack	1	26.87	26.87
Shipment	--	67.74	67.74
Taxes	--	0	0
Customs	--	129.57	129.57
Total	--		458.72
Total(in Rupees)	--		**Rs. 28899.36**

8 Work Division

1. Tushar Sinha (110260019)
   (a) Components Selection
   (b) Codes for interfacing Sensors
   (c) PID control Algorithm
   (d) Sensor testing and debugging
   (e) PID simulations
2. Jash Banker (110260003)
   (a) Filter

(b) PID control algorithm

(c) PID simulations

3. Mukul Sholapurkar (110260008)

   (a) Filter

   (b) PID control algorithm

   (c) PID simulations

4. Digvijay Wadekar (110260016)

   (a) Receiver-Transmitter: Direct Angle output Code , Testing , Debugging

   (b) ESC transmission to motors

   (c) Documentation of Final Latex Report

5. Abhishek Ukey (110260006)

   (a) Receiver-Transmitter: Testing and Debugging

   (b) Complete hardware assembly

6. Ankur Agrawal (11D260003)

   (a) Receiver-Transmitter: Constraints for code

   (b) MultiWii Simulation Software but didn't use it finally

   (c) CDR Documentation

7. Hardik Godara (110260011)

   (a) Components Selection

   (b) IMU interfacing

   (c) PID control algorithm

   (d) PID simulations

9 Remaining Work

1. PID tuning on simulations
2. Filter testing and debugging
3. Assembling individual modules
4. Flight test of Quadcopter

References

[1] Arducopter project (model used for Flight gear simulations)

[2] http://oddcopter.com/2012/02/06/choosing-quadcopter-motors-and-props/

[3] http://blog.oscarliang.net/build-a-quadcopter-beginners-tutorial-1/

[4] http://www.ctn-dev.org/index.php?page=news&article=6

[5] http://www.forkrobotics.com/2013/06/using-the-gy80-10dof-module-with-arduino/

[6] http://www.flybrushless.com

[7] https://www.sparkfun.com/products/10724

[8] Arduino wire library

[9] http://rcarduino.blogspot.co.uk/2012/01/can-i-control-more-than-x-servos-with.html

[10] Arduino PinchangeInt library