

AUTONOMOUS QUADCOPTER

SUMMER PROJECT '13

PROJECT MENTOR:

Nikhil Upadhye

Ayush Mittal

Sonu Agarwal

TEAM MEMBERS:

Kartik Agarwal

Karthik Korada

Piyush Sahoo

Rushikesh Chaudhari

Vikram Singh

INTRODUCTION

A Quadcopter is a multicopter lifted and propelled by four rotors. In this project we aimed to build an Autonomous Quadcopter which can balance itself while flying. This quadcopter also consists of manual control system (transmitter-receiver)

.For our project we decided to build the quadcopter frame in a simpler way having symmetrical four arms on which a motor with a propeller is mounted on every arm.

To make this Quad copter autonomous, we used Arduino platform to program and applied PID algorithm to calculate the output values of motor commands by using input values from transmitter and sensors. We used an Inertial Measurement Unit (IMU) sensor which give values regarding angles and angular velocities of quad copter frame.

MECHANISM

Each rotor produces a thrust and a torque about its center of rotation and these forces are used to fly and move Quad copter. Two rotors mounted on opposite arms of quad copter are set into clockwise and the another two anticlockwise. These orientation of motors and their direction of rotation cancels all the torque generated given the speed of the motors are same. Now, if we change the speed of the motors attached on the left and right of quad keeping the speed of the other two same, creates 'Yaw' motion. Similarly, 'Pitch' and 'Roll' movements are gained by changing the speed of different motors. See the images below

WHAT IS AUTONOMOUS ?

In general, Quad copter are very much difficult to control by

only manual control without using any sensors. So, if we use some specific motion sensors and apply some algorithms to the values read from these sensors, the Quad copter can itself balance its motion. If we use the above technique to control the Quad copter, it can then be called as "Autonomous".

MANUAL CONTROL

All the motors are connected to Electronic speed controllers (ESC) which control the speed of the rotors and supply motor the required power. In the manual control, the signals to the ESC's are directly sent by the transmitter remote control through receiver. Transmitter is used by the user to control the quad manually. If the user changes the input, transmitter sends radio signals to receiver and receiver changes these signals to PWM signals. These PWM signals are sent to ESC's causing the change in the speed of the motors. For our project, we are using 'Futaba' transmitter and receiver.

OUR APPROACH

In our project, we decided to use Arduino Mega 2560 as our programmer and IMU sensors (9 DOF and 6 DOF IMU) as the motion sensors. We decided to use PID algorithm in our code to get the final motor commands. There are different modes of PID algorithms which can be used to control a quad copter such as Stabilize, Attitude and GPS control. We first aimed to use

gyro rates in our PID algorithm i.e. the Quad copter can itself reduce vibrations generated in its frame due to non-zero angular velocities to acquire a stable current position.

IMU

We have used both "9 DOF razor" and "6 DOF" IMU sensors to get values of angles changed from initial position along all the three axis and angular velocities of the same. These values are used as input values to our arduino code. Generally an IMU consists of an accelerometer, gyroscope and magnetometer (in case of 9 DOF).The values read from this components are calibrated to get the desired sensor values. The libraries used for reading calibrated data from 9 DOF and 6 DOF are "RAZOR AHRS" and "FreeSixIMU" respectively.

CODING and PID algorithm

We have done all our programming on arduino mega in C / C++ language. Arduino code consists of a setup part and a loop which runs continuously containing the calculations part. The main part of program of Quad copter was PID algorithm. PID is used for minimizing the error between the target position and the current position the quad has achieved. PID algorithm is basically Proportional-Derivative Integral controller used for loop feedback mechanism. PID consists of mainly three

constants 'P', 'I' and 'D'. The 'P' term produces an output value that is proportional to the current error value. The 'I' term is proportional to both the magnitude of the error and duration of error. The last 'D' term is used to calculate the derivative of the process error. We wrote the code to read the signals from the RC receiver using "PinChangeInt" library. We also wrote the code to read sensor values. Please see the "IMU" section. Then we wrote a function that includes receiver, sensor and calculation of PID algorithm. Then we formed a loop that runs approximately on 100 Hz and continuously does the calculations. We also wrote a function that writes motor commands to the individual motors in terms of PWM signals.

INTEGRATING ALL THE FACTORS

At first we had a wooden frame on which four motors with propellers were mounted. The arduino circuit board along with IMU sensors and RC receiver was mounted on the centre of quad frame. Then we uploaded the code and tested each component on the quad frame to check if it's working properly. At first we checked the basic working of PID algorithms through 'Roll', 'Pitch' and 'Yaw' motion. Then we tested for PID tuning setting one by one the 'P', 'I' and 'D' terms by checking its different motion behaviour. It was a step by step process to get to the final testing phase.

PROBLEMS FACED

- 1) The data reading frequency of 9 DOF was too much lower to run the code perfectly. 9 DOF communicates with Arduino through SPI communication which is relatively slower than I2C communication used in 6 DOF IMU. Because of this we decided to use 6 DOF IMU which gives data at approximately 333 Hz.
- 2) Due memory overload in our main code there was a problem occurred in which arduino code was setting up again and again.
- 3) We had a problem of breaking of the frame many times because of our wooden frame. But it reduced the vibrations effectively allowing the IMU sensors give accurate values.
- 4) There was a problem in reading signals from the RC transmitter, we had to find a suitable library function for this task. Finally we used "PinChangeInt" library.
- 5) Firstly it was easy to read only one channel of RC Receiver. But we had a problem in reading multiple channels. To solve this problem we used values of multiple channels of RC receiver directly in the PID algorithm.
- 6) We first used "servo.write" function to give PWM signals to ESC's which gave lesser accuracy , because it worked only in the range of 40-170 degrees. To resolve this problem, we used "servo.writemicroseconds" function.

RESULTS

Having integrated all the things nicely we have a good result of our project. We are now able to fly and stabilize of our Quad copter for 50-60 seconds which is considered as a good flight. We are using only gyro-rates in our PID codes. We are yet to use Euler angles used for the horizontal stabilization of the quad copter by using stabilize PID mode.

FUTURE PLANS

In the future we are looking forward to make our Quad copter fully autonomous without using any other sensor. Also we are thinking of using GPS for long range autonomous flight or ultrasonic sensors for object avoidance. By using these techniques we can reach to the level of autonomous Quad copter.

REFERENCES

<https://dev.qu.tu-berlin.de/projects/sf-razor-9dof-ahrs/wiki/Tutorial>

<http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>

http://www.starlino.com/imu_guide.html

https://github.com/AeroQuad/AeroQuad/blob/master/Libraries/AQ_Motors/Examples/MotorTest/MotorTest.ino

<http://blog.mobileapes.com/2010/04/control-all-four-motors-from-ard>

[uino.html](#)

<http://rcarduino.blogspot.co.uk/2012/04/how-to-read-multiple-rc-channels-draft.html>

<http://www.astro.hr/library/Magnetometer.pdf>