Development of Pitch, Roll and Yaw axis Controller for Autonomous Flying Vehicle

Professional Application Project

Electronic and Information Systems Engineering Department

Diego Fernando Montoya Orozco

Ulm, Deutschland

Fall/Winter Semester

2008

Table of Contents

Acknowledgments

It would not have been possible for me conclude my Engineering studies without the many efforts that my father did throughout all of his life. I thank him for everything he gave me and I dedicate the work done here to him especially.

I also have to thank the professors who supported me during the development of the project, Frank Steiper and Bernardo Cotero. Thanks to my mother Carmen, my sister Mela and brother Juan Pablo.

Introduction

Goal

The Duocopter project intends to achieve the development of an Autonomous Flying Vehicle (AFV) capable of completing a given task (going from point A to B for example) analyzing its environment and taking decisions along the way to sort out obstacles and stay in a safe state. It is a helicopter with two propellers located in the same plane, one on each side of the aircraft, with the possibility to control the tilt of each rotor in the pitch degree of freedom as well as the propeller speed independently for each side.

Because of the extension of the Duocopter development process it had to be divided in defined modules. This part of the project is going to be focused in the development of model of the behavior of the helicopter during flight, a controller to stabilize the orientation of the helicopter based on this plant model and a signal processing algorithm capable of getting useful information out from the noise vulnerable sensor signals. All of this requiring computer, software, hardware, mathematics, mechanical and physics. Thus we will further call this part of the Duocopter project, witch is a project itself, as the Duocopter controller project, or just “the project”.

Importance and Justification

Robots are capable o doing tasks that are too dangerous, dirty or dull to be suited for humans. Today, the development of robots has encountered new possibilities as well as new challenges with the birth of new technologies such as Micro Electronic Machines (MEMs), wireless communications, Real Time Operating Systems (RTOS) and LiPoly batteries. The autonomous flying vehicle area is one that has been always been explored since the beginning of robotics, but has not accomplished important results until the last decade, regrettably only with military purposes.

With the Duocopter, the goal is to develop an AFV capable of assisting rescue, exploration, industrial, surveillance, service and artistic tasks. Such could be used to go to difficult to reach places during an earthquake or hurricane emergency, exploring a newly discovered mine or cave, keeping an eye on a residential area or making what were before impossible shots for a film. The global needs for technological solutions are every day more demanding and the mass production of electronic components make this solutions affordable, leaving just a great gap to be filled with innovative design. The development of this project intends to meet those needs.

Additionally, the present world demands cooperating markets and globalization is taking part in education. The fact that this project is done in cooperation between the Iteso and the Hochschule Ulm makes it possible for more students to participate in more projects of the kind. The result is mutual exchange of knowledge and experiences that can only enrich the academic and personal formation, and then that student is capable of giving more to the society.

Part I Professional application report

Chapter 1. Background and current problems

2.1 Institutional background

The Ulm University of Applied Science (Hochschule Ulm, HSU) is an institute of superior education with a technical-technological focus located in Ulm, a city in the province of Baden-Württemberg. Located in the south of Germany, is right of Europe's technological technological development area, keeping close contact with several companies the university achieves a very efficient academic-industrial cooperation. Currently, projects for energy management, artificial intelligence, traffic analysis as well as autonomous vehicles are being developed within its installations.

Instituto tecnológico y de estudios superiores de occidente, or Iteso for its initials, is also a university with a technological focus. It is located in Guadalajara, Mexico; the city named as the Mexican Silicon Valley because of its technological development and ”second strongest economic potential of any major North American city”[1]. This institution has already close interaction with the local industry, having industry experts as well as high level academic professors participating in the education process.

This universities have had a student exchange agreement and every year theres students participating in the Engineering International Programs in the HSU coming from Guadalajara, as well as students participating in regular courses in Iteso, coming from Ulm. Both institutions participate in this part of the project, and is the first time one of their students goes to the other university to make the final project. This is possible in part thanks to the new studies plan offered in Iteso, witch involves a final Professional Application Project (PAP, *Proyecto de Aplicación Profesional* in Spanish) instead of a thesis, witch is more compatible with the Diploma Project (Diplom Arbeit) system of the HSU. This broadens the link between both universities and opens the possibility for more students from both places to elaborate their final projects in the other university.

2.2 Projects background

The Hochschule Ulm is the main developer of the whole project, providing the previous developments in the project, the laboratories and hardware, and the project will be continued here. Professor Frank Steiper of the Hochschule Ulm has been in charge of the Duocopter project all along. Iteso participates in this part of the project through Professor Bernardo Cotero's supervision of my work. The project started in the year 2007 and since then has been an open project where students collaborate in small parts at the same time that Prof. Steiper participates with his own work to continue developing the Duocopter.

Before this part of the project, the helicopter had already a mechanical structure that had the two rotors in place, a central housing for circuitry and hardware, a base to hold the rotor battery and to serve as the helicopters stand and a tail that assists in the mechanical stability (Figure 1).

1 *Cities of the Future*, FDi magazine(2007-04-23).

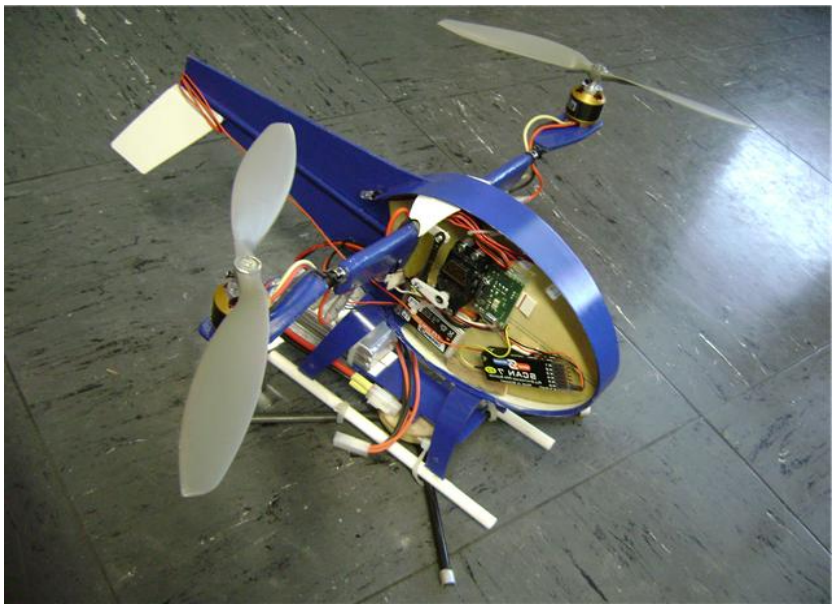


Figure 1. Duocopter's target prototype

The structure is mainly solid, since previous experiments proved that flexible structures reduced stability. It also counted with a microcontroller in a development board, gyroscopes, accelerometer, servos, a radio controller (RC) receiver and a LiPoly battery for the digital hardware. All this is to be mounted in the central housing. Previously, a structure with a 'T' shape, mounted in a cage with all the mentioned equipment was constructed, in such a way that it would only let the dummy helicopter rotate in one degree of freedom (Figure 2).

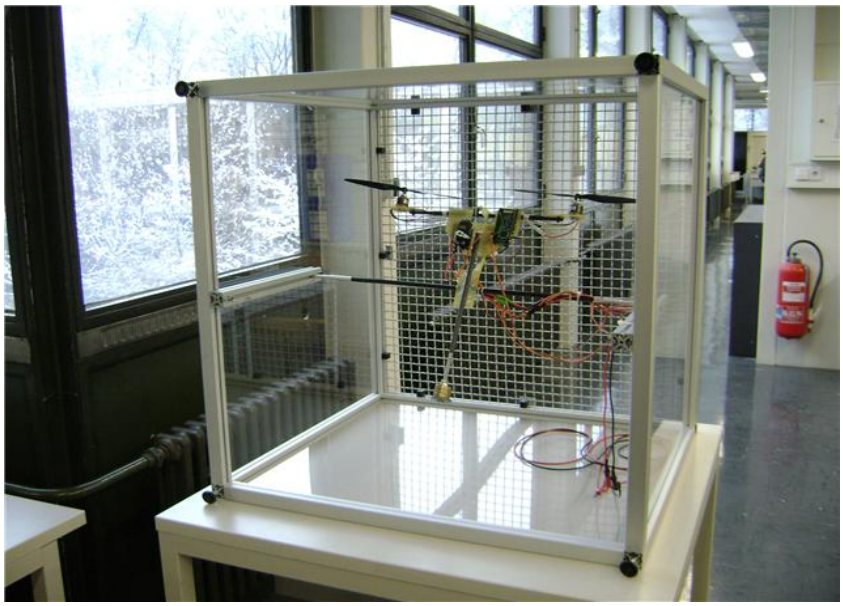


Figure 2. Duocopter's test prototype inside protection cage

It would be used as test dummy to observe the behavior of the pitch controller. The hardware is further discussed in Chapter 3. Finally, before starting, there was already a base software that included the initialization of the different modules, test routines, interrupt handling for RC communication and a USART interface to assist the process. There was also an basic PID digital controller used to be tested once everything was put in place.

2.3 People involved

As mentioned before, Professor Frank Steiper of the HSU is involved actively in the Duocopter development and monitored closely the process of the Duocopter controller project and Professor Bernardo Cotero from Iteso University worked as an advisor. Myself, Diego Montoya, student of Electronic Engineering in Iteso University, worked throughout this part of the project as to accomplish my Final Project, to further improve my engineering capabilities and to reinforce my social compromise.

Indirectly, other students that participated in the Duocopter development previously are involved. So are the students that will continue working on it. Also the international office of the HSU is involved, since they assisted me in my integration to the campus and the city, and the people there helped me find a scholarship. Since it is a collaboration between Iteso and HSU, both academic communities are indirectly involved, and they benefit of the cultural exchange and enrichment that such collaborations always bring.

Last, the society will be benefited by the final product of this whole project, since the resulting product aims to be of general public reach and to serve in several areas as mentioned before, putting robotics to the service of the common citizen.

Chapter 2. Project's Administration

2.1 Planning and Time

The Duocopter is an open project, and as an academic research and development project, it doesn't have a previously selected product launch date. It is more a free project that allows the students that wish to participate to apply new ideas and to work in their respective fields. For that reason, once a student (or group) is willing to participate and the professor considers that he (them) has got the right knowledge and abilities, that particular part of the project is planned for that case.

The general plan for this project consisted several different phases, starting the first of September and finishing the 19th of December, witch comprises a 16 week period, equivalent to that of the Iteso, but shifted a couple weeks later to be compatible with the HSU dates at the same time. The plan looked like this, with the number at the end indicating the number of weeks dedicated to that phase:

i. Documentation and Hardware (2)
ii. Integration of new accelerometer module LIS3LV (1)
iii. Model design (1.5)
iv. Controller design, simulation and implementation (3.5)
v. Signal processing algorithms design, simulation and implementation (4)
vi. Tests – redesign (4)
vii. Report and presentation (Attachment Duocopter Presentation.ppt) (2)

In my case it was initially planned to develop a controller for the pitch degree of freedom only. But as the project advance, we realized that it was possible to develop similar controllers for the roll and yaw degrees of freedom, witch would allow to make the first on flight tests. This was done in the tests and redesign phase.

2.1.1 Human Resources

I worked in this project full time while Prof. Steiper constantly supervised my activities and advised me on the next steps to take. Both him and me worked in the hardware build up and modification. Prof. Cotero assisted in other activities concerning the project management and developing in Iteso.

2.1.2 Cost

And an overall budget of all the project specific hardware comes up to around 2000 euros, without taking into account the price of things that the HSU had before starting the project (PC, Oscilloscope, etc.) and the price of man working hours.

The costs of the project is mainly funded by resources given to Prof. Steiper by the HSU to develop various projects. Once the Duocopter project reaches its final stages, it will be used to demonstrate the products of the whole process and get private funds. On the other hand, my expenses during my stay in Germany are mostly funded with a Deutscher Akademischer Austausch Dienst (DAAD) scholarship.

2.1.3 Risk

The risk in the project was mainly distributed in 4 different activities. The hardware functioning, model, signal processing and tests. Since they are not highly dependent on each other, the risk of high impact (when it comes about effort) is moderate. The test an signal processing activities are the most dependent activities, thus they get higher amount of time in the planning. There is no actual economic risk because the project is not funded. Because of the research nature of the Duocopter project every knowledge acquired is of great value if a correct analysis is done.

2.2 Monitoring and Control

2.2.1 Methods

The progress is monitored through weekly meetings and occasional reports. The meetings serve to keep the progress smooth and share ideas for the current work. The reports illustrate concrete pieces of work and help deciding what the best path to follow.

2.2.2 Real measurements: Effort, time, resources, costs and others.

The effort in man working hours dedicated to the project can be calculated as follows:

$$number\ of\ weeks * \left(\frac{lab\,,\,office\ hours}{week} + \frac{additional\ hours}{week}\right) = 16 * (30+4) = 604\ working\ hours$$

Plus the working hours of the professors and other people that assisted.

In conventional time, it lasted 16 weeks of full time work, going from September to mid December, as explained in the planning section.

A complete resources list include the following material:

- Personal Computer with Internet access.
- Laptop
- Robostix development board
- Accelerometer
- Gyroscopes
- RC receiver
- Remote control
- Servos
- LiPoly battery and charger
- Test structure with cage
- Prototype structure
- Rotors
- Propellers
- Power source
- Oscilloscope
- Heat gun
- Industrial Glue
- Several discrete electronic components
- Books
- Office supplies

Chapter 3. Project's technical execution

3.1 Project development

The way the development of the project is explained here differs a little from the way the plan looks and from the actual practical development of the project. That is because the explanation of the project is more illustrative and easier to follow if some of the components of the process are rearranged. So first the plant model is going to be explained as well as some simulations that help show the consistency of the model. In the second part the sensing process is explained, so the hardware has to be described in more detail. In the third part two different signal processing algorithms are going to be presented and compared. The fourth part explains how both the signal processing and the controller were implemented in C code. And last, the result of the experiments in the actual helicopter structures are discussed.

3.1.1 Controller Modeling and Simulation

3.1.1.1 Plant Model

First we need to have a model of the plant, that let us predict the behavior of the apparatus. So, in principle we are going to explain the modeling of the plant only taking into account the pitch degree of freedom, and afterwards the symmetric resemblance between this axis and the other two rotational axis will be explained. The plant has a structure that has two rotors with propellers on top, and the rest of the body lays underneath. If the plant is observed from the side, so the pitch axis comes perpendicular out of the page and we establish a right hand

coordinate system, the plant would look like Figure 3. The alpha shows the positive sense of acceleration.

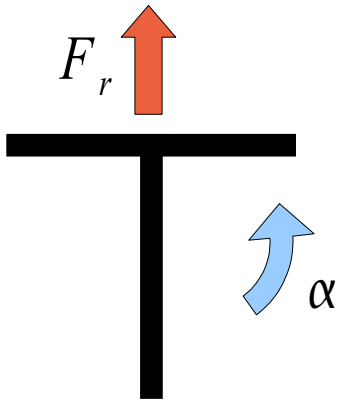


Figure 3. Basic Plant Model

Then it is know that with no controller yet, there will be no action taken but the one directed directly by the user of the helicopter. That is, if nobody manually commands an action, the plant will remain unchanged. The center of mass will be taken as the origin, and it is for this model placed in the middle of the vertical black line. With the rotors propellers placed in a plane perpendicular to vector coming from the origin, there is no resulting force coming from the rotors that would induce a torque. The only force capable of applying a torque is the resistance of the wind. The equation that models the plant would be

$$\alpha = -R\left(\frac{d\theta}{dt}\right)^2 \tag{1}$$

In ideal conditions, no wind, rotors perfectly perpendicular to earth and perfect mechanical components, the helicopter would take off straight up. But then, when we have any other condition, such as non horizontal take off, wind, non parallel propellers, etc., the situation changes and a control maneuver has to be done. For this two states are taken into account. They are the plant tilt θ_p and the plant angular speed ω_p. A reference inclination is further defined as θ_{ref}. Figure 4 illustrates the states and shows an example with zero (horizontal) reference tilt, positive plant tilt and and negative plant angular speed.

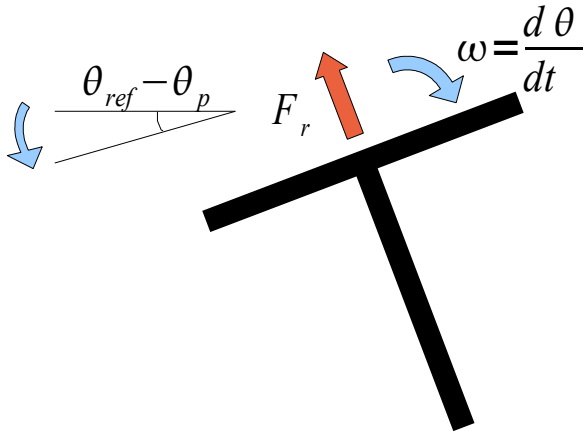


Figure 4. Plant model with two states

3.1.1.2 Controller model

Then we can use this states to take a control action. The inclination of the rotors to inflict a torque and cause angular acceleration can be used. Now, let α_θ be an angular acceleration proportional to the inclination error θ_{ref} - θ_p, α_ω and be an angular acceleration proportional to the angular speed, but with a negative sign (Figure 5).

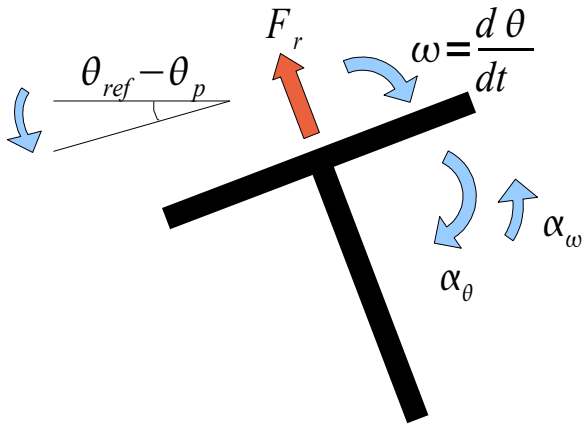


Figure 5. Model with computed accelerations

Now if a total acceleration α is obtained adding both accelerations α_ω and α_θ, a new second order differential equation is obtained. The constants k_i and k_d are added to give different weights to the angular accelerations and to be able to further modify the step response.

$$\alpha = k_i(\theta_{ref} - \theta_p) - k_d \frac{d\theta_p}{dt} - R\left(\frac{d\theta_p}{dt}\right)^2 \quad (2)$$

Since a stable 0 angular speed and $\theta_{ref} = \theta_p$ is intended, the squared angular speed becomes really small. Also, the helicopter is more or less aerodynamic (thus $R \rightarrow 0$), so the third member of the right side of the equation becomes insignificant and can be ignored.

$$\alpha = k_i(\theta_{ref} - \theta_p) - k_d \frac{d\theta_p}{dt} \quad (3)$$

In a case where the angular acceleration corresponding to the inclination error is more significant that the angular acceleration corresponding to the angular speed, the resulting acceleration α would be that of the Figure 6, and the control action to be taken to produce this would be to tilt both rotors counterclockwise, resulting in a force coming from the propellers that is perpendicular to a vector coming from the origin. This tilt action is also known as common cyclic tilt.

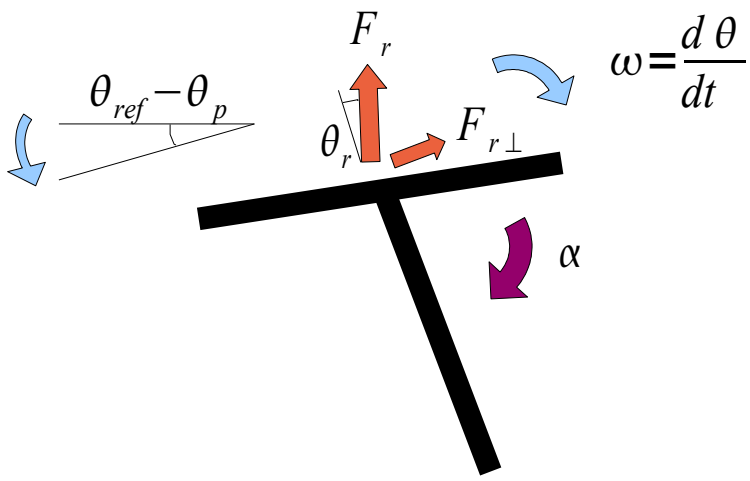


Figure 6. Control action taken

This states where used for two reasons. First, they can be estimated from the accelerometer and gyro signals. Second because the speed measurement can be used to make artificial resistance and the tilt can be used to make artificial energy storage. That is, the plant will be forced to behave as a second order, ordinary differential equation system, as shown in (3).

For a case where the angular speed acceleration is more significant, the control action would be to do a common cyclic tilt in a clockwise sense. That doesn't seem as a logic action at a first glance, but the case may arrive were it is the best action (for example if the rotational speed is very high and a 'break' maneuver has to be done). The right values for the k_i and k_d constants are found through mathematical analysis of the differential equation (3). First, remembering that

$$\alpha \equiv \frac{d^2\theta_p}{dt^2}$$

equation (3) is reordered

$$\frac{d^2\theta_p}{dt^2} = k_i(\theta_{ref} - \theta_p) - k_d\frac{d\theta_p}{dt} \tag{4}$$

$$\frac{1}{k_i}\frac{d^2\theta_p}{dt^2} + \frac{k_d}{k_i}\frac{d\theta_p}{dt} + \theta_p = \theta_{ref} \tag{5}$$

Equation (4) with a step input with amplitude $|\theta_{ref}|$ as tilt reference and initial conditions

$$\theta(t=0^-) = 0, \quad \frac{d\theta}{dt}(t=0^-) = 0$$

the solutions are the following

$$\theta_p(t) = |\theta_{ref}|u(t)[1 - e^{-\sqrt{k_i}*t} - te^{-\sqrt{k_i}*t}] \qquad if\ k_d = 2\sqrt{k_i} \rightarrow D = 1 \tag{6}$$

$$\theta_p(t) = |\theta_{ref}|e^{-\sqrt{\frac{k_i}{2}}t}[1 - \cos(\sqrt{\frac{k_i}{2}}t) - \sin(\sqrt{\frac{k_i}{2}}t)] \qquad if\ k_d = \sqrt{2k_i} \rightarrow D = \frac{1}{\sqrt{2}} \tag{7}$$

That way, by simply modifying k_i and k_d the desired step response is obtained, with a speed response only limited by the maximum speed and tilt of the actuators and the digitalization process. Note that if (5) is put in an integral-differential way, the k_i sets the weight of the integral term and k_d the weight of the differential term, thus they names are decided. Note that k_i has to have units of rad/s^2 and k_d of rad/s. Another way to see it is that the undamped resonant frequency is

$$\frac{1}{\omega_o^2}=\frac{1}{k_i} \quad \Rightarrow \quad \omega_o=\sqrt{k_i}$$

The magnitude controlled common tilt of the rotors has to be found using classic rotational physics.

$$\alpha=\frac{\tau}{I} \tag{8}$$

$$\tau=F_{r\perp}\, r=F_r \sin(\theta_r) r \tag{9}$$

$$\Rightarrow \alpha=\frac{F_r \sin(\theta_r) r}{I} \tag{8)+(9)=(10}$$

for a known moment of inertia I and a steady weight distribution it can be said that

$$I=mr^2 \tag{11}$$

$$\Rightarrow \alpha=\frac{F_r \sin(\theta_r) r}{mr^2}=\frac{F_r}{mr}\sin(\theta_r) \tag{10)+(11)=(12}$$

also is know that for a hover flight

$$F_{rhover}=mg$$

so as long as we operate with a rotor induced force close to that of the hover, we can obtain the rotor inclination from (12) as

$$\theta_r=\sin^{-1}\left(\frac{\alpha mr}{F_r}\right)=\sin^{-1}\left(\frac{r}{g}\alpha\right) \qquad if\ F_r=F_{rhover}$$

$$\theta_r=\sin^{-1}\left(\frac{r}{g}\alpha\right) \tag{13}$$

3.1.1.3 Plant & Controller simulation

A simulation of the plant was required to observe the effects of the non linearities of the plant with the controller attached. MatLab's tool Simulink was used to create a block model of the plant and controller described before, but leaving place to add the non linearities such as the limit tilt of the the actuators, rotor speed variations and the zero order hold of the microcontroller's ADCs. The resulting model (Attachment Duocopter_non_linear.mdl) is shown in Figure 7.

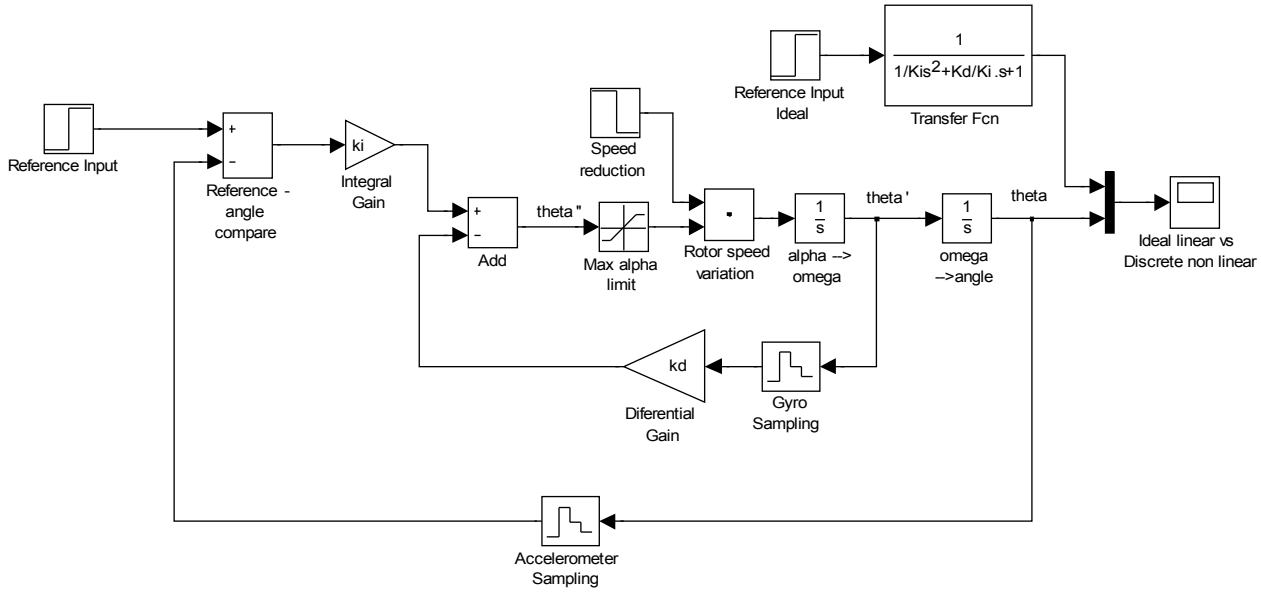


Figure 7. Model of plant and controller

This Simulink model is based on equation (4). To explain it let us start in the *theta*'' signal, witch is the controlled acceleration α. It is limited accordingly to the maximum rotor tilt (45°), witch results in a maximum acceleration calculated as follows, using values extracted from the prototype and servos characteristics:

$$\alpha_{max}=\frac{F_{hover}\sin(\theta_{max})\cdot r}{I}=\frac{mg\sin(\theta_{max})\cdot r}{mr^2}=\frac{g}{r}\sin(\theta_{max})=\frac{9.8}{0.1}\sin(\frac{\pi}{4})=69.3\,rad/s^2$$

Then it is multiplied to a *Speed reduction* factor. This is initially one and then at time 0.4 seconds becomes 0.8. This is to simulate a significant propeller force reduction to 80% of hover propeller force. The resulting signal is integrated first to obtain *theta*' (ω_p) and then *theta* (θ_p) both passed through a 40Hz zero order hold to simulate the digitalization of the signal. Then *theta* is compared with the *Reference Input* (θ_{ref}) to obtain the tilt error and multiplied by the weight factor k_i. *Theta*' times k_d is subtracted to the resulted signal to generate once again *theta*''. The two blocks in the upper right are a second reference input identical to the first one and a transfer function equivalent to the ideal model of the controller. They are used to make a comparison.

After running the simulation with a k_i=50, k_d=2*√k_i and then k_d=√(2*k_i), the results of graphics in Figure 8 a) and b) respectively, are obtained as response for a step input of 0.2 radians of magnitude.

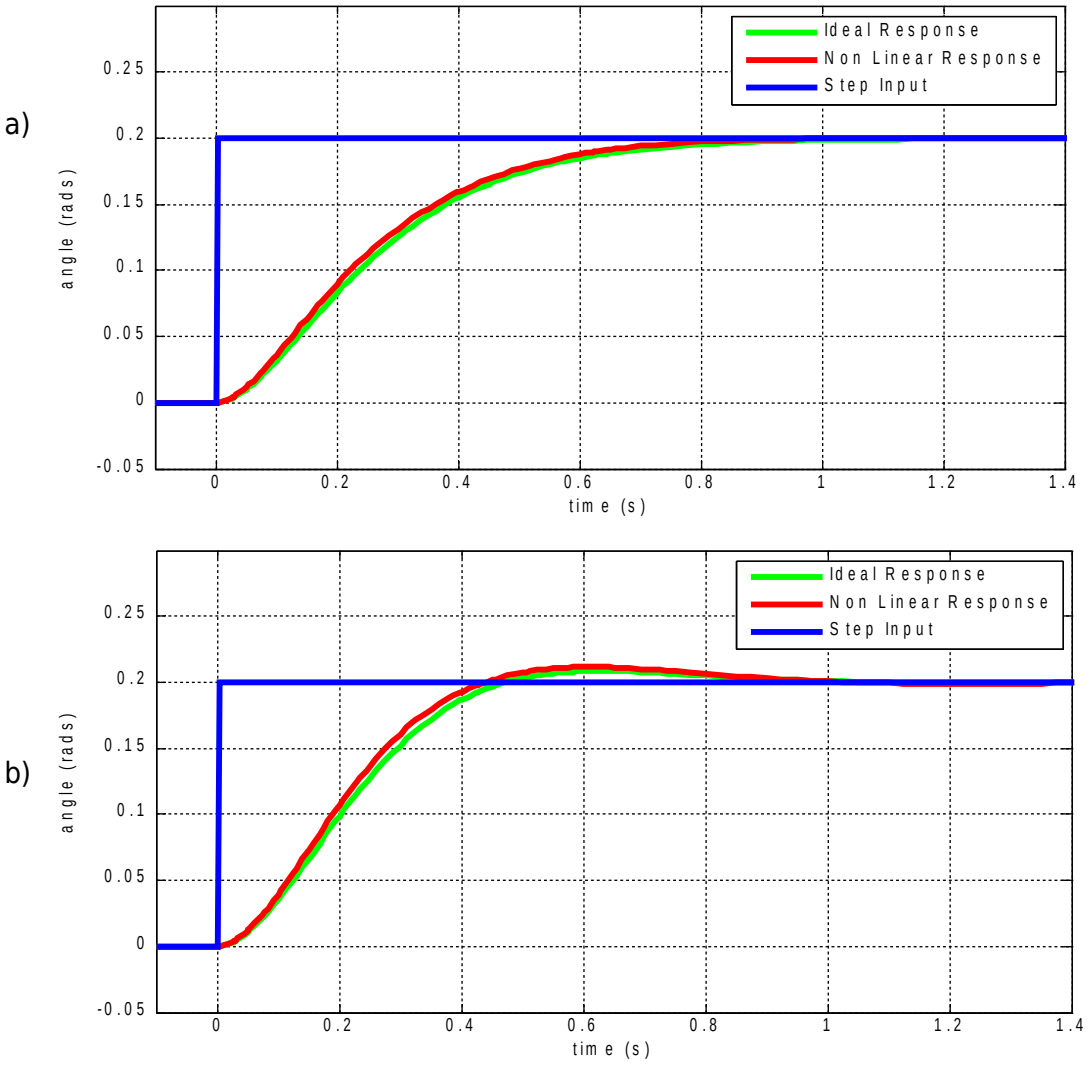


Figure 8. Model simulation results

In both graphics the step is compared with the response of a linear second order system and the non linear system model of the Duocopter. In the first case (Figure 8 a)) it can be observed that a critical damping will result in an smooth curve with no overshoot, that can be said to have reached 90% of the desired state in time 0.6 seconds. In the case of Figure 8 b) it is observed that a damping of $1/\sqrt{2}$ results in a faster response but since the damping is less than 1, we get a small overshoot, witch is found to be of less that 10%. I this case we can say we reached 90% of the desired state in time 0.4 seconds. It is important to observe that in both cases the speed reduction applied in time 0.4 seconds is almost unnoticeable, so we can say that the controller response is practically rotor speed independent. The non linearities seem to be non significant and they only result in a small variation on the speed of the response. At this response times it is difficult to appreciate if is the zero order hold, the saturation of the actuators or the rotor speed reduction what is causing the variation. So a new simulation is run, this time with a $k_i=50$ and $k_d=\sqrt{(2*k_i)}$ but with no speed reduction and the results are shown in Figure 9.

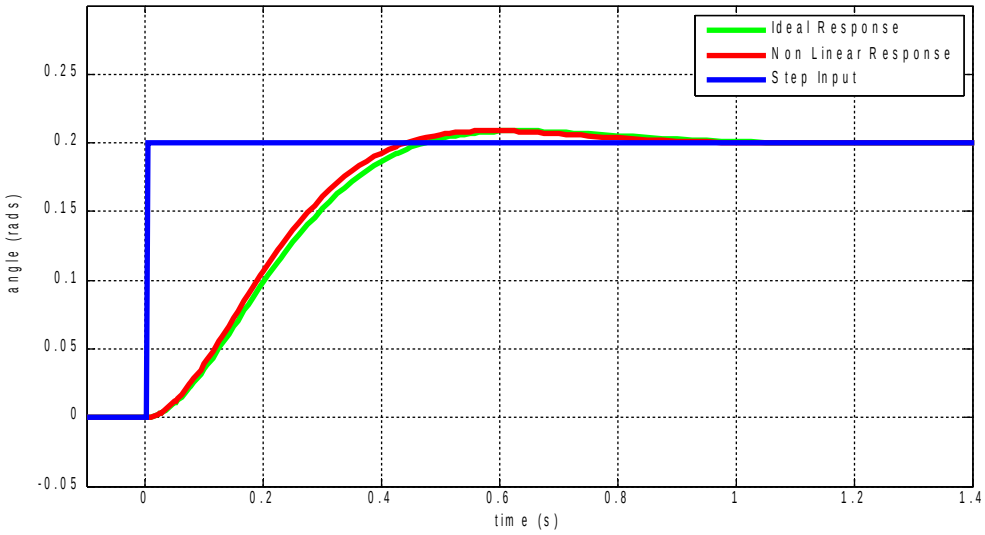


Figure 9. Model simulation results without rotor speed reduction

It can be observed that in the second simulation (Figure 8 b)), the non linear response is ahead of the Ideal response until around t=0.45, when the ideal response overshoot is smaller for the ideal case. In the last simulation, the non linear response is always ahead of the Ideal response. That way it can be concluded that the speed variation only inflicts a non significant variation in the speed of the the response. A last simulation is run, this time with k_i=1000 and $k_d=\sqrt{(2*k_i)}$ and no speed reduction in order to appreciate with more detail the effect of the zero order hold and saturation blocks (Figure 10).

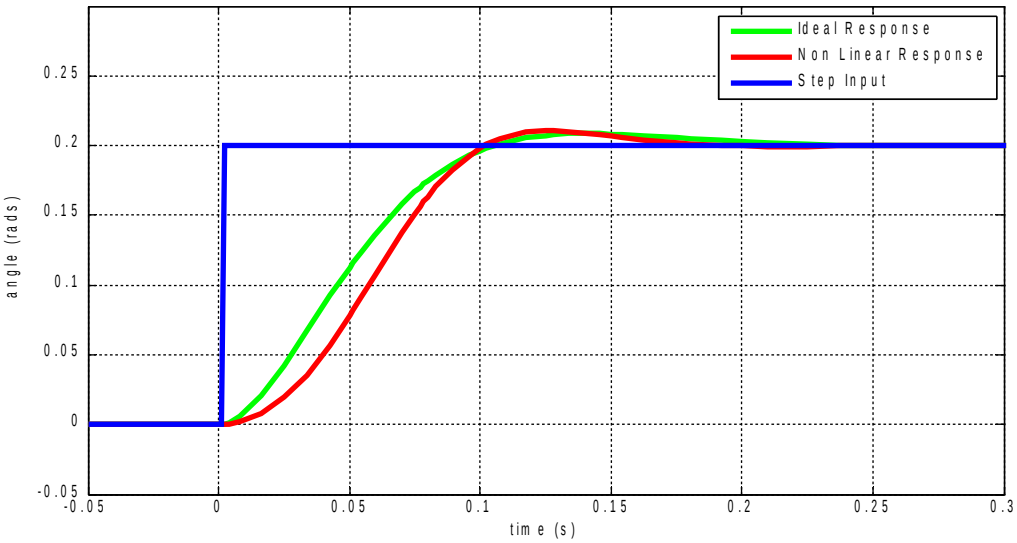


Figure 10. Model with augmented response speed

The graphic shows that now a slew rate effect is present, because faster responses demand higher values of acceleration, witch are limited for the non linear case. Then, for high speed responses, the saturation block is limiting the speed. This also let us deduce that is the zero order value that is causing a faster response in the first three simulations, because its holding an error value for every 1/40 seconds, witch is bigger when it is first sampling, and this is enough to accelerate the response.

That way it is concluded, according to the simulations, that the controller model is usable and stable, and that for starting values we can use a k_i=50 and k_d=2*√k_i and expect smooth responses of less than a second since the values used for the model where estimated from the test prototype.

3.1.2 Hardware & State Estimation

To be able to control the helicopter we need to have a state estimation system. This is achieved with several pieces of hardware. This includes a microprocessor to do on board calculations and signal processing; a RC receiver module to receive input commands and signal the microprocessor; servos to control the propeller inclination and rotor speed; and sensors to measure the inclination and angular speed both directly and indirectly.

3.1.2.1 Microprocessor

The microprocessor used is an Atmega128 manufactured by Atmel (Figure 11 a)). It features an 8-bit Reduced Instruction Set Computer (RISC) architecture, 16MHz clock signal, 128KB flash, 4KB EEPROM, two 16-bit timers, two 8 bit timers, two 8-bit PWM channels, six programmable resolution PWM channels, dual programmable USARTs, JTAG Interface, SPI, Two-Wire Serial Interface, 8-channel 10-bit ADC and several GPIO pins, among another features.

a) 

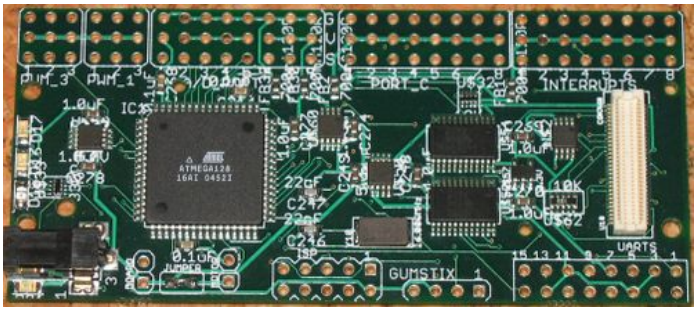
b) 

Figure 11. Atmega128 and Robostix board

The ATmega128 is mounted in a Robostix board (Figure 11 b)), witch gives easy access to the different ports but has some other disadvantages as restricting the use of the SPI pins.

In order to program the microprocessor, the JTAG interface was used together with the AVR Studio software, specially designed to develop applications for Atmel's microprocessors. It was also in AVR Studio that the code was written and debugged. Further discussion about the code implementation is included in the Implementation section.

The ATmega128 was initially intended to be the final processor for the target prototype, but afterwards the rise of computation need and new project possibilities brought the idea of using a multi core configuration. But that is not to be analyzed in this project, and for the tests done here the Atmega128 was still used.

3.1.2.2 RC Module

The chosen RC receiver module was a SCAN-7 manufactured by SimProp (Figure 12 a)). It is used together with a remote controller mx-16smanufactured by Graupner (Figure 12 b)). The receiver is mounted in the test or target prototype while the remote controller is used to command it.

a) b)

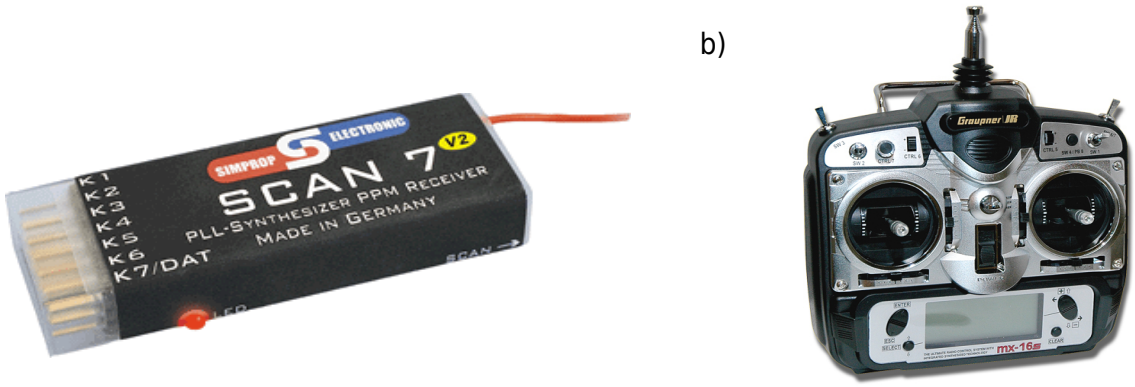


Figure 12. RC receiver module and Remote Controller

The receiver had to be modified to be able to extract a summed signal of all the channels. The pins that can be observed in Figure 12 a)) give a PWM signal with a 2ms period and a zero value of 50% duty cycle. That is the same standard used by the servos and the rotors. In order to save pins, a wire is connected directly to the central processing unit of the RC receiver unit. That is again a Atmega128. This cable is the connected to a GPIO pin of the Robostix board to read all channels using an interrupt service routine.

3.1.2.3 Servos

The servos used for the rotor inclination were a pair of BMS-705MG manufactured by BLUE BIRD (Figure 13). They are commanded through PWM and the position of the actuator arm is proportional to the pulse width.

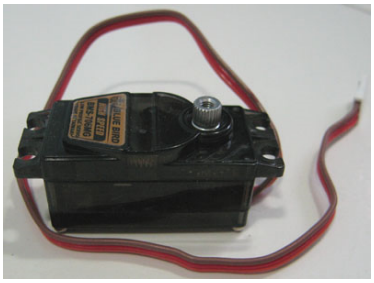


Figure 13. Servo

3.1.2.4 Sensors

Two different kind of sensors were used, to sense both acceleration and angular speed. To sense the acceleration a LIS3LV manufactured by STM (Figure 14 a)) was used. It is a three axis 2-6g configurable accelerometer with SPI and I2C interfaces. The gyros are three ADXRS manufactured by Analog Devices (Figure 14 b)). Each of one has a analog voltage output for one axis with a 5-12.5mV/°/s sensibility.

a) b)

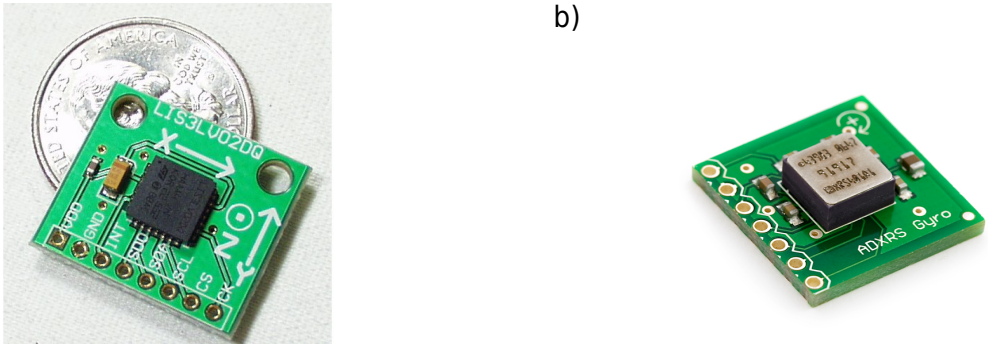


Figure 14. Sensors

3.1.2.5 Rotors and Propellers

The motor used are two brush less AC 2212/20 manufactured by AXI (Figure 15 b)). For the speed of the rotors a pair of Roxxy BL_CONTROL 825 controllers manufactured by robbe are used (Figure 15 a)). They are much more complex as the inclination servos since they prevent over current and switch-on propeller rotation, they provide pulses to the stepper motors and they work with a 12V source. They are also commanded with a PWM, but the response to the pulse with is more or less a square root function, that is the speed of the rotor is proportional to the square root of the pulse with. Also the both controller characteristic curves are not identical, so measurements had to be made.

a) b)

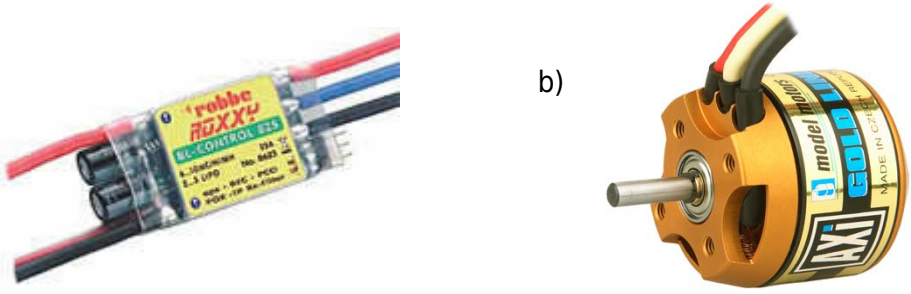


Figure 15. Rotor controller and rotor

Curves for both controllers were approximated and a empirical found simple gain was applied to adjust both curves. The fitting curves and adjusted values are shown in Figure 16. First, a) shows the measured Revolutions Per Minute vs. Duty Cycle measured in both rotors (doted lines, with punctual values in the triangles) and the best fit curves using a square root function. Then b) shows the result after a gain adjustment of 1.1176 to the previously measured values. Last c) shows the actual measured values after implementing that same gain in software to the pulse with commanding the rotors.

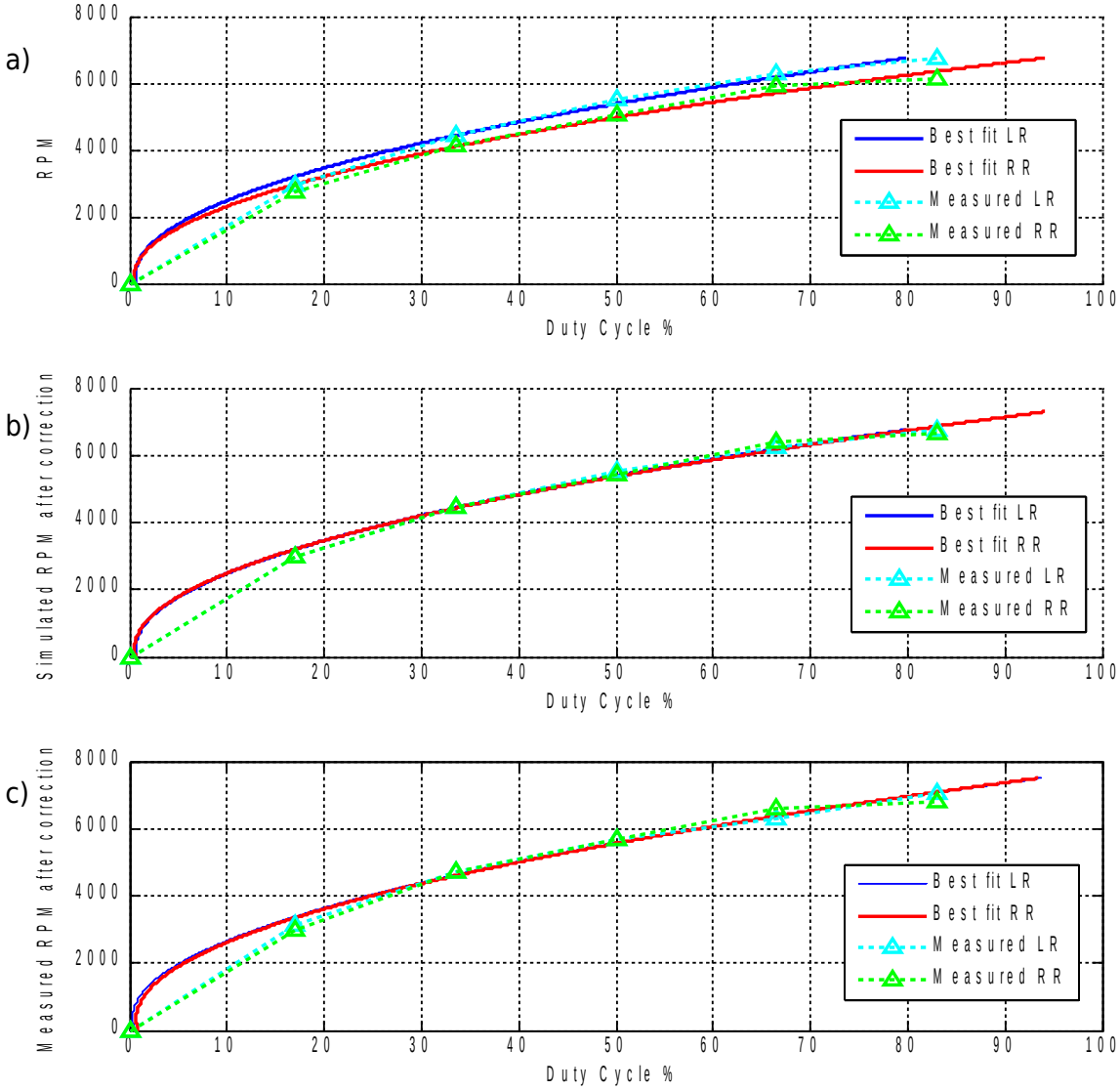


Figure 16. Motor operation curves coupling

3.1.2.6 Hardware Layout

In the test prototype, one gyro was mounted in the central part of the structure to sense the pitch angular velocity, as well as a an accelerometer to get a reference from earth's gravity. For this case there is only one servo to control both rotors inclination because the common tilt is the only one needed to control the pitch. The microcontroller is also mounted in the central part. Figure 17 illustrates where were they mounted.

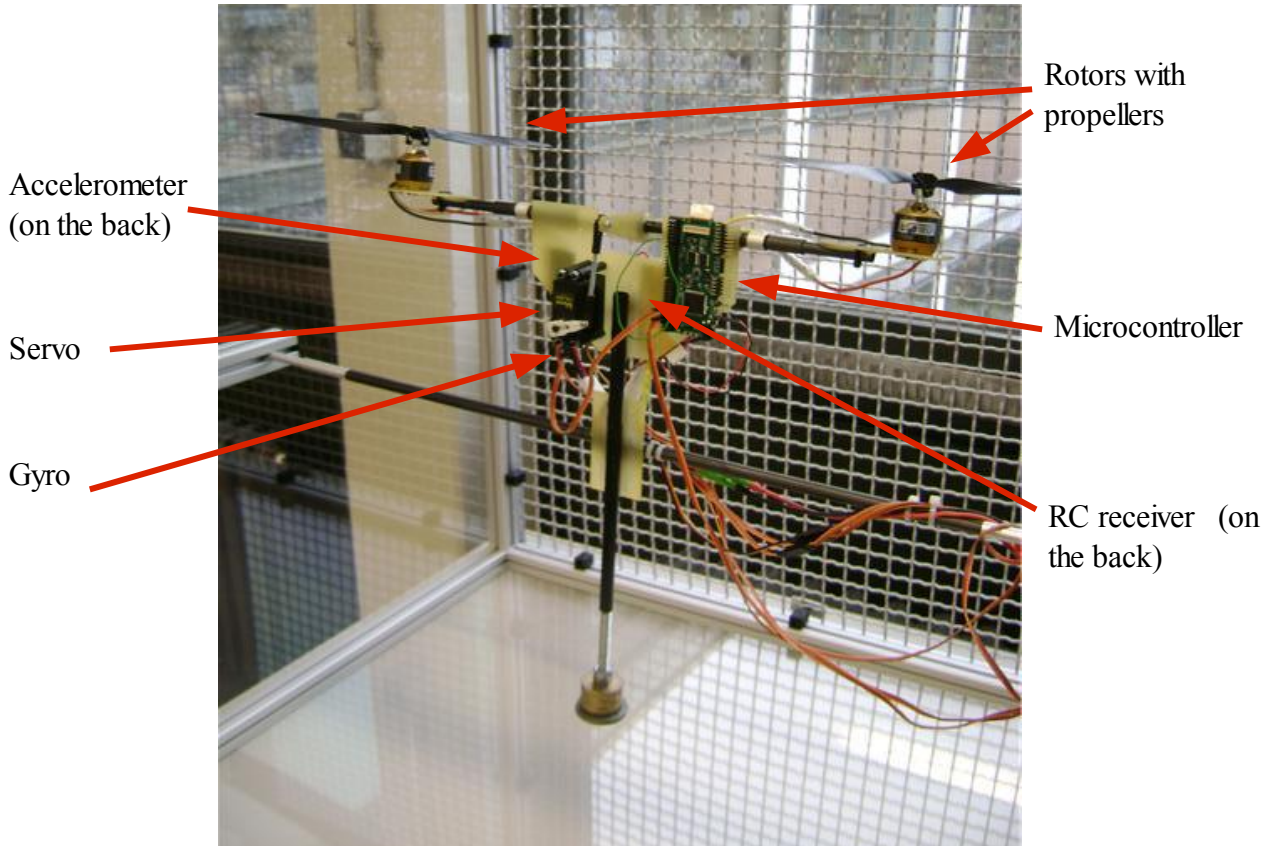


Figure 17. Test prototype with hardware mounted

The target prototype is a little bit different because it has one servo on each side, two batteries and two more gyros mounted in the central housing as it can be seen I Figure 18. The three gyros are mounted as three perpendicular faces of a cube to sense the different rotational speeds. There are two batteries because the digital circuitry and rotors need different sources to avoid malfunctioning risk. The target prototype is also more helicopter resemblant.

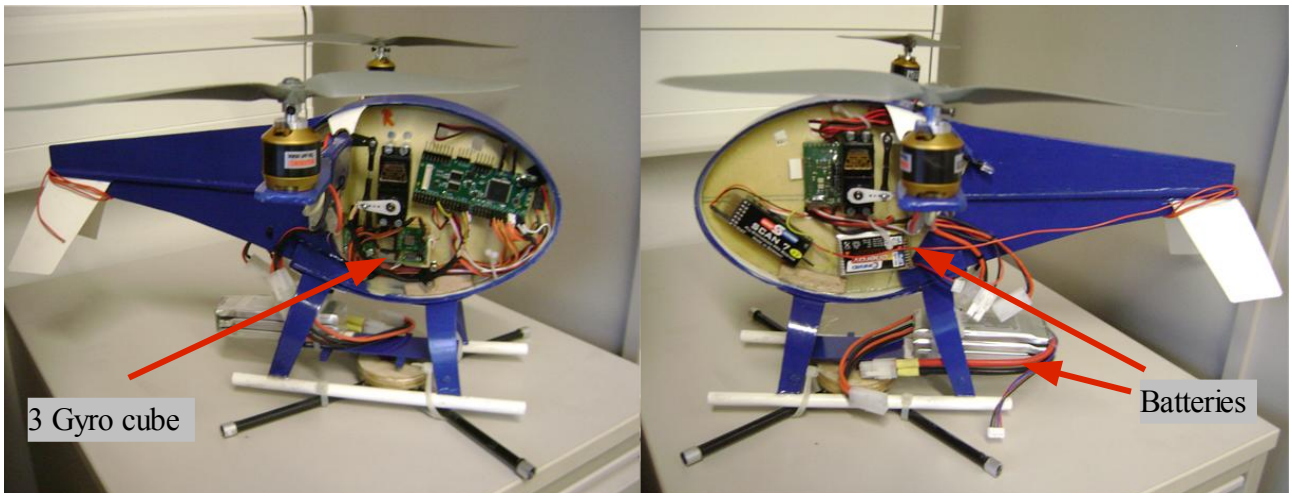


Figure 18. Both sides of the target prototype

3.1.3 Signal processing

For the signal processing there were three different perspectives contemplated. First, apply a low pass filter because high frequency noise is to be expected coming form the rotors ad electrical noise because of battery's non linearities. Second perspective is to apply a filtering filter with a probabilistic algorithm. There are many noise sources, and the capacity of the microprocessor and priority of RC communication limits the use of interrupts. This makes the closed loop execution rate time to have a jitter, since not time interrupts where used to ensure the contrary. There is also control noise involved. All this makes us expect noise in all frequencies, and also expect that it will be more or less Gaussian noise. This way the state of the helicopter is not directly measured, but a probability distribution of the state is found, with the advantage that probabilistic robotics algorithms "can even actively chose to reduce their uncertainty when it appears to be the superior choice"[2]. The third perspective is to apply a combination of both filters, depending on the results obtained in the first two cases.

Samples of the raw data extracted from the sensors are taken, and using this data the filters where applied and compared.

3.1.3.1 FIR Filtering

The low pass filter was implemented using a Finite Impulse response filter. To make an effective low pass with at least 40dB attenuation in the rejected band and with a cut frequency that could be significantly low at least a 15th degree filter should be applied. The problem is that for a 15th degree low pass filter the delay is of 7.5 samples, or around 0.1875 seconds, witch could be critical to the controller.

3.1.3.2 Bayes Algorithm

The Bayes Algorithm is a probabilistic robotics algorithm used to calculate a probability distribution (or belief) from measurement and control data. The basic algorithm comprises the pseudo code lines stated below in Table 1:

$$
\begin{aligned}
&1: \quad Algorithm\ BayesFilter(bel(x_t), u_t, z_t)\\
&2: \quad\quad for\ all\ x_t\ do\\
&3: \quad\quad\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}\\
&4: \quad\quad\quad bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)\\
&5: \quad\quad end\ for\\
&6: \quad\quad return\ bel(x_t)
\end{aligned}
$$

Table 1. Bayes Filter algorithm

The parameters sent to the function are the present belief distribution, the present control action and the present measurement. That is bel(x_t), u_t and z_t, respectively. That means that this procedure needs to know how is the probability density distributed in the present moment 't'. It also needs to know witch is the control action taken at the moment of the calculation. Finally the result of the present (last) measurement is needed. The variables u_t, x_t, and z_t are in general vectors of length 'n', where 'n' is the number of states measured in the system. In consequence

2 *Probabilistic Robotics*, Sebastian Thrun, Wolfram Burgard, Dieter Fox, p. 5, 2005

bel(x_t) is also a vector. The *for* in line 3 only means that the algorithm should be done for every state in the x_t vector. Line 3 calculates an *a priori* belief distribution using the total probability theorem, together with a conditioning of Baye's Rule for combining probabilities of independent random variables[3]. That way it produces a new belief distribution that is influenced by the control action. That is answering the question, how is the probability distribution changed by the control action u_t? Then line 4 updates the probability distribution simply multiplying the *a priori* belief and the probability of the measurement given a present state. In other words, it integrates the measurement with the computed belief to produce an *updated* belief. This last one is what the function returns. There are a few assumptions that have to be taken in order for this algorithm to be correct. First, that all the probability density distributions all integrate to 1. Thats why in line 4 theres a 'η' that works as a normalizing factor. That means the probability of an event over all the space is exactly 1. That also means that this functions only have a meaning in the integration sense, so even if it has a value over 1 somewhere in the function, the actual probability of that event in a given interval of the space would be equal to that of the integral over that interval, and that cannot be bigger than 1. Another important assumption is that the model works as in a *Markov world*[4] and the states are complete, witch means that it is enough to know the current state to predict the measurement z_t. This is a somehow confusing algorithm, but it will be much better understood after it is explained in a concrete case in the next section.

3.1.3.3 Kalman Filter

The Kalman filter is a recursive algorithm that estimates the state of a dynamic system from a series of noisy measurements. It relies on the premise that the noise involved in the measurement and the control action is Gaussian, and uses the properties of this well know probability density functions to prove that the resulting estimation is also a Gaussian distribution[5]. It also can be proven that as the filter is applied, it minimizes the variance of the estimation error[6]. It beginning it was thought to be the ideal signal processing algorithm for the Duocopter, since there were so many noise sources and that noise was expected in all the bandwidth of the sensors. The general algorithm is that shown in Table 2.

1: $Algorithm\ KalmanFilter(bel(x_t), u_t, z_t)$
2: $\bar{\mu}_t = A_t \mu_{t-1} + Bu_t$
3: $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
4:
5: $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
6: $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
7: $\Sigma = (I - K_t C_t) \bar{\Sigma}_t$
8: $return\ \mu_t, \Sigma_t$

Table 2. Kalman Filter Algorithm

3 *Probabilistic Robotics*, Sebastian Thrun, Wolfram Burgard, Dieter Fox, p. 17-27, 2005
4 *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*,Lawrence R. Rabiner *Proceedings of the* IEEE, 77 (2), p. 257–260, February 1989
5 *A new approach to linear filtering and prediction problems*, Kalman, R.E. (1960). Journal of Basic Engineering 82 (1)
6 *Kalman Filtering*, Dan Simon, Embedded Systems Programming, June 2001, p.72-79

Once again, it needs the probability distribution of the states, the present control action and the present measurement. The system has to have a known linear dependency for state transitions with added Gaussian noise, as well as a linear dependency for observations. That means that:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$
$$z_t = C x_t + \delta_t$$

Table 3. System linear model

The algorithm works assuming all Gaussian distributions whose first moment is the mean μ and second moment is the covariance Σ. It first calculates the *a priori* mean and covariance using the matrices A and B from the state transition equation, and an *a priori* Covariance with matrix A and R (the covariance of the posterior state noise). Then in line 5 it computes an optimal Kalman gain based on the observation matrix C and covariance of the measurement noise matrix Qt. Line 6 updates the predicted mean and line 7 updates the covariance using the optimal gain.

The simpler way to explain it is using graphics that let us appreciate the procedure in a more empirical perspective. Figure 19 has graphical sequence of the Kalman Filter algorithm.

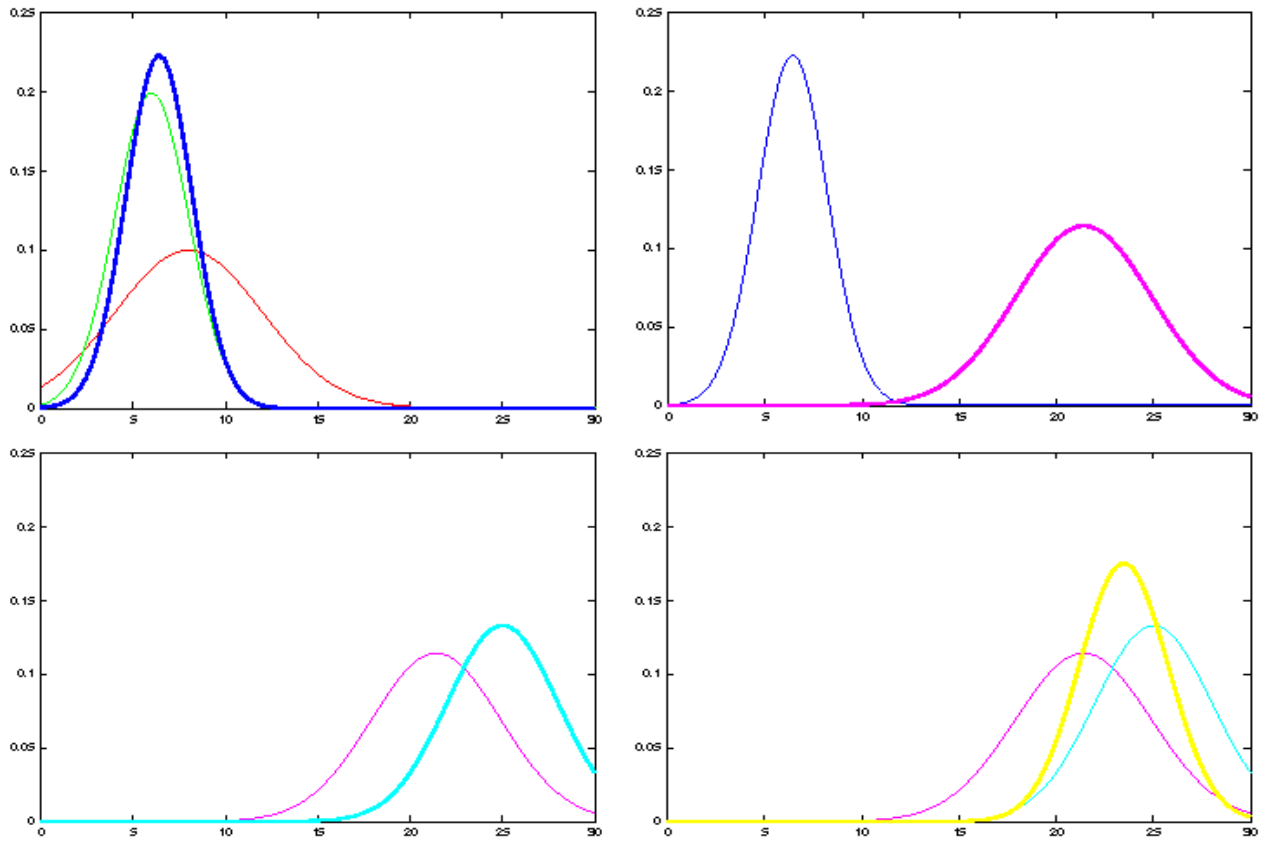


Figure 19. Graphical sequence of Kalman Filter algorithm

All the curves in the figure are probability density distribution with a Gaussian shape. Starting in the upper left corner, observe the curve in blue. This same curve is repeated (only with a thiner line) in the upper right corner. Lets call this curve belief in time 't-1'. Lets observe that the mean is of around 6. Then, a known control action with its related noise is taken. In this case it could be a movement to the right of around 15 units. Thats why, in the upper right corner, the blue curve is shifted to the right and becomes the pink curve, with mean around 21 units. It is also more spread because the noise involved in the control action causes it to have a bigger covariance. The pink curve would be the *a priori* belief in time 't'. The new mean and covariance values are those calculated in lines 2 and 3 of the Kalman Filter algorithm, and are the equivalent of line 3 in the Bayes algorithm. In the next step, there is a measurement with its related mean and covariance. That is the cyan colored curve in the lower left graph. If a new predicted state is computed, call it belief in time 't', using the measurement and *a priori* belief, a curve something like the yellow colored one in the lower right corner is found. If the model is right, the new curve has a smaller covariance than that of the curves used to compute it. Lines 5, 6 and 7 of the Kalman Filter algorithm find the parameters of the new predicted state (yellow curve), and are the equivalent of line 4 in the Bayes algorithm. So thats how a predicted state is found and the process can be repeated after a new control action and a new measurement are taken. The Kalman Filter algorithm is just a specific case of the Bayes algorithm where all the distributions are Gaussians. The mathematical derivation is straight forward and it can be found together with a more detailed explanation in [7].

3.1.3.4 Alpha-Beta Filter

The Kalman Filter can be really useful in noisy applications. It is really robust and can fit numerous applications. But one of the main problems when implementing the Kalman filter is the demanding computation power. For this project all operations are done in floating point to assure portability, but that consumes time and doing matrix operation may not be the best option. Also there is a high implementation complexity involved, and some matrix operations like inversion may run into numerical problems. Thats why the need for a simpler filter arose. One that still has some of the advantages of the Kalman filter.

The Alpha Beta filter is some how that. It works in noisy signals, is based in a state space model, and uses previous states, present control action and measurement to predict a state. The difference with the Kalman Filter is that it uses a set of steady gains related to the noise covariance, instead of the optimal Kalman gain matrix 'K' and partly giving up some of the adaptability of the filter. Also, the Alpha Beta filter uses the mean as a predicted state directly, instead of giving a probability distribution. But other than that, both filters are pretty much the same. It is proved that the optimum time-invariant second-order Kalman filter for tracking position and velocity has the same architecture as the deterministic Alpha-Beta filter[8].

In this project the Alpha Beta Filter is the chosen algorithm because the simplicity of implementation and low computation needs, and it will be used to predict the angular speed and tilt of the Duocopter. The algorithm for a is described in Table 3.

7 *Probabilistic Robotics*, Sebastian Thrun, Wolfram Burgard, Dieter Fox, p. 40-54, 2005

8 *Reconciling Steady-State Kalman and Alpha-Beta Filter Design*, **JOHN H. PAINTER,** Senior Member, IEEE Xxas A & M University **DAVID KERSTETTER,** Member, IEEE General Dynamics, Pomona Division STEVE JOWERS McDonnell-Douglas Astronautics Co.

1: $Algorithm\ AlphaBetaFilter(x_m(t), x_p(t), v_m(t), v_p(t), a(t))$
2: $x_s(t) = x_p(t) + \alpha \cdot (x_m(t) - x_p(t))$
3: $v_s(t) = v_p(t) + \beta \cdot (v_m(t) - v_p(t))$
4:
5: $x_p(t+1) = x_s(t) + \Delta t \cdot v_s(t) + \frac{\Delta t^2}{2} \cdot a(t)$
6: $v_p(t+1) = v_s(t) + \Delta t \cdot a(t)$
7: $return\ \ x_p(t+1), v_p(t+1)$

Table 3. Alpha Beta Filter

Let 'x' be the position (or tilt angle) and 'v' be the velocity (or angular speed). First, in lines 2 and 3, a smoothed position and velocity have to be computed using the previous predicted state ($x_p(t)$) and the measurement $x_m(t)$. The α and β are just two fixed gains that range from 0 to 1. If a value of 1 is set, the smoothed value is equal to the measurement, so history of states has no relevance. If a value of 0 is set, the smoothed value is equal to the previous prediction, so the measurement has no relevance. In simple words, gains α and β are used to set how much do you 'trust' the measurements. The next steps, lines 5 and 6, are to predict the new states for time 't+1' using the basic formulas relating position, speed and acceleration, where acceleration is the known control action.

This seemed to be the best suited algorithm for this application. It basically works like an Infinite Impulse Response low pass filter, but with the advantage of allowing high frequency components to pass when they are expected according to the control action. Also, the algorithm an be expanded if there is another measurement for the acceleration. This is called Alpha Beta Gamma filter. The process is exactly the same, just now working in three states.

3.1.3.5 Filtering real signals

A set of samples for the angular velocity and tilt where produced by software. 100 samples at 40Hz with Gaussian noise of variance σ=1 and mean μ=0, and an angular acceleration of 100 rad/s^2 through samples 50-55.Then an Alpha Beta Gamma algorithm (ABG) was implemented in MatLab (Attachment AlphaBetaGamma.m) as well as coefficients for a 9th degree FIR low pass filter with cut frequency of 0.07π. Figures 20 and 21 show the result obtained after the FIR filtering.

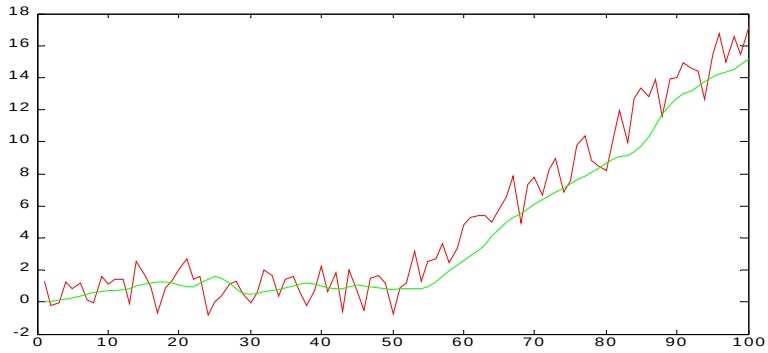


Figure 20. Simulated raw inclination (red) and FIR filtered signal (green) vs. samples

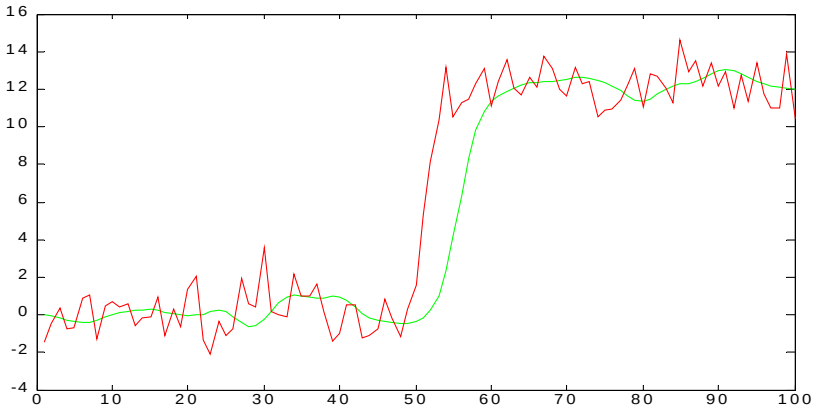


Figure 21. Simulated raw angular speed (red) and FIR filtered signal (green) vs. samples

A very low cut frequency had to be set for the FIR filter in order to get a more or less decent output signal. It can be appreciated that because of the properties of the Gaussian noise, there are low frequency components that the FIR filter can't eliminate. Afterwards, the same samples where passed through the Alpha Beta Gamma Filter. Figures 22 and 23 illustrate the results.

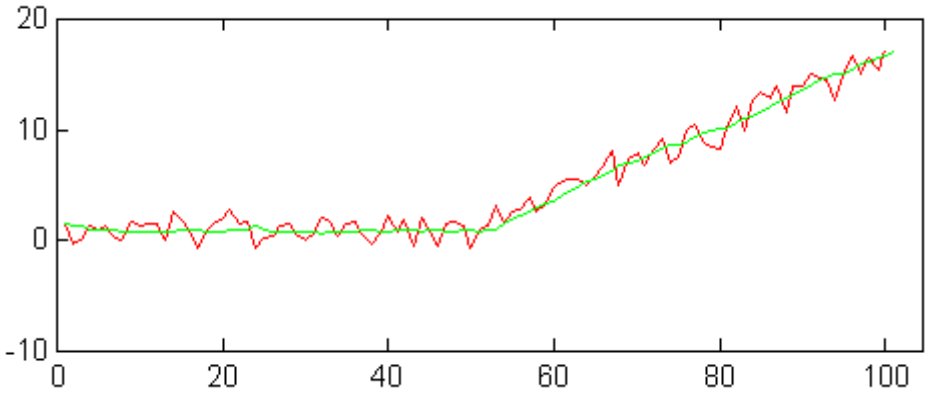


Figure 22. Simulated raw inclination (red) and Alpha Beta Gamma filtered signal (green) vs. samples

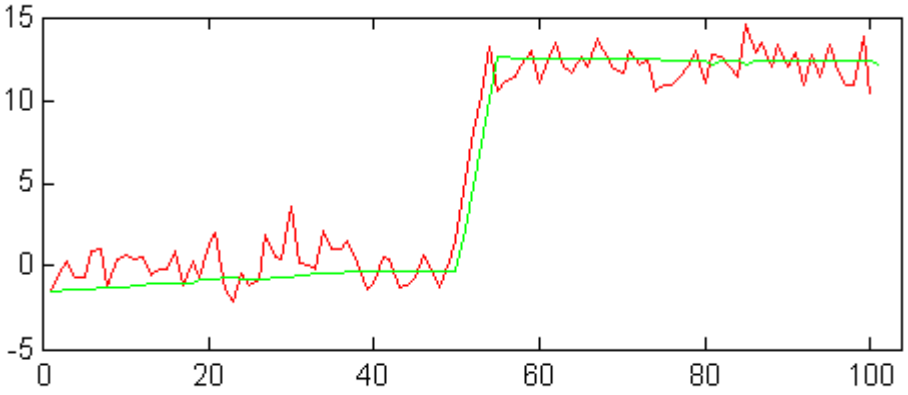


Figure 23. Simulated raw angular speed (red) and Alpha Beta Gamma filtered signal (green) vs. samples

It is Easy to appreciate how the Alpha Beta Gamma algorithm is much more effective when noise is expected to be in all the bandwidth. The resulting signal is very clean and the delay is smaller in comparison with the FIR filter, because of the predicting nature of the ABG filter.

Using the USART interface, a set of samples from the sensor's raw data was extracted with a 10Hz sampling rate to make some tests of the algorithms over the real data. While extracting the data, the test prototype was shifted between -45° and 45° every ten seconds. The gyros provided the angular velocity directly and the accelerometer provided the value of a component of gravity in one of its axis. With the known value of gravity, the component in a axis tangent to the pitch and using the inverse sinus function, the inclination is obtained. After extracting the data, portions of it was analyzed to see the characteristics of the noise. Figure 24 a) shows a histogram and b) shows a spectral analysis (magnitude of Fast Fourier Transform) of a portion of noisy data from the samples taken with the rotor on.

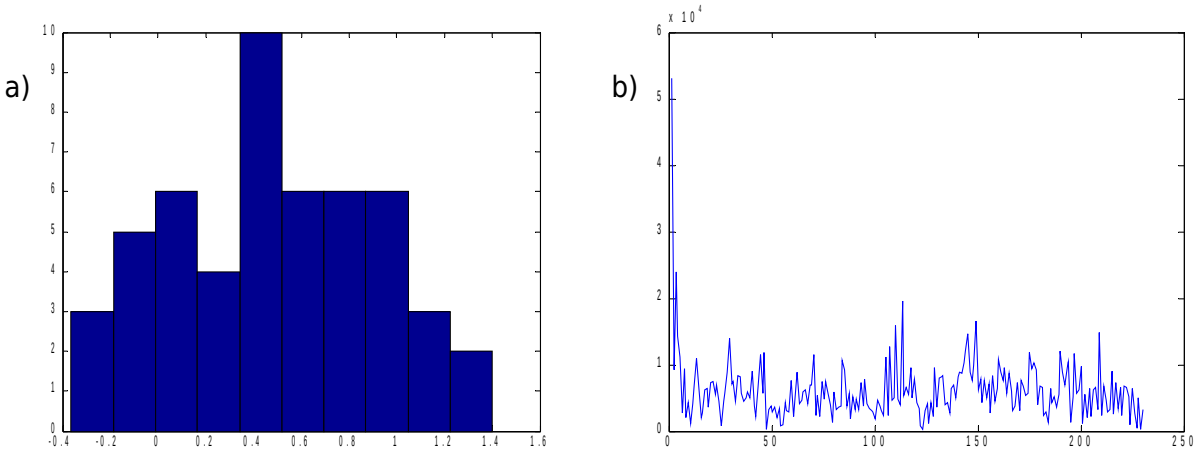


Figure 24. Analysis of noisy signal, Histogram and magnitude FFT

The histogram, extracted from a part of the data taken when the prototype had an inclination of around 0.6 rads shows a more or less Gaussian distribution, and the FFT proves that there is noise in all the frequencies. The low frequency peak is caused by the shift of position every 10 seconds. This facts suggest that the ABG filter will work.

Then the AB algorithm was applied to the data extracted with the rotors off and choosing $\alpha = 0.1$ and $\beta = 0.15$ (Figure 25). Position is in radians, angular velocity in rads/s and angular acceleration in rads/s^2.

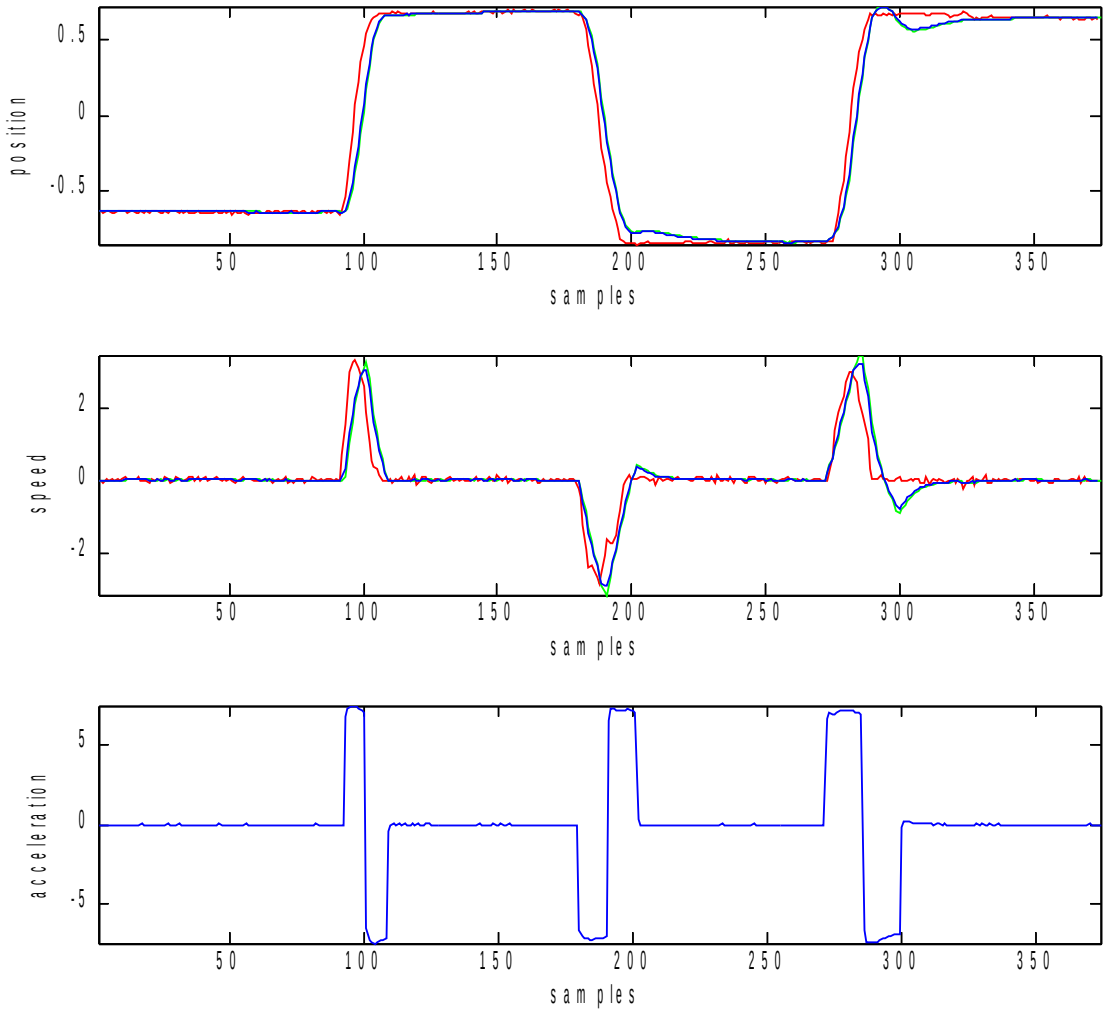


Figure 25. Measurements (red) and predicted states (blue) vs. samples, rotors off

As an extra, a third state is given by the Alpha Beta Gamma filter, witch is a composite angular acceleration, taken in account a measurement extracted from the derivative of the angular speed. Gamma is the 'trust' level for this state and the value used above is of γ=0.05. It can be clearly seen that the signal measured is already very clean, but still the filtering had to be tested to see any side effects. There is a slight mean value variation and more noticeably a delay of around three or four samples. In the actual prototype the sampling would be at least four times bigger, so three or four samples shouldn't be relevant. The next step (Figure 26) is the same signal processing with the samples taken with the rotors on.

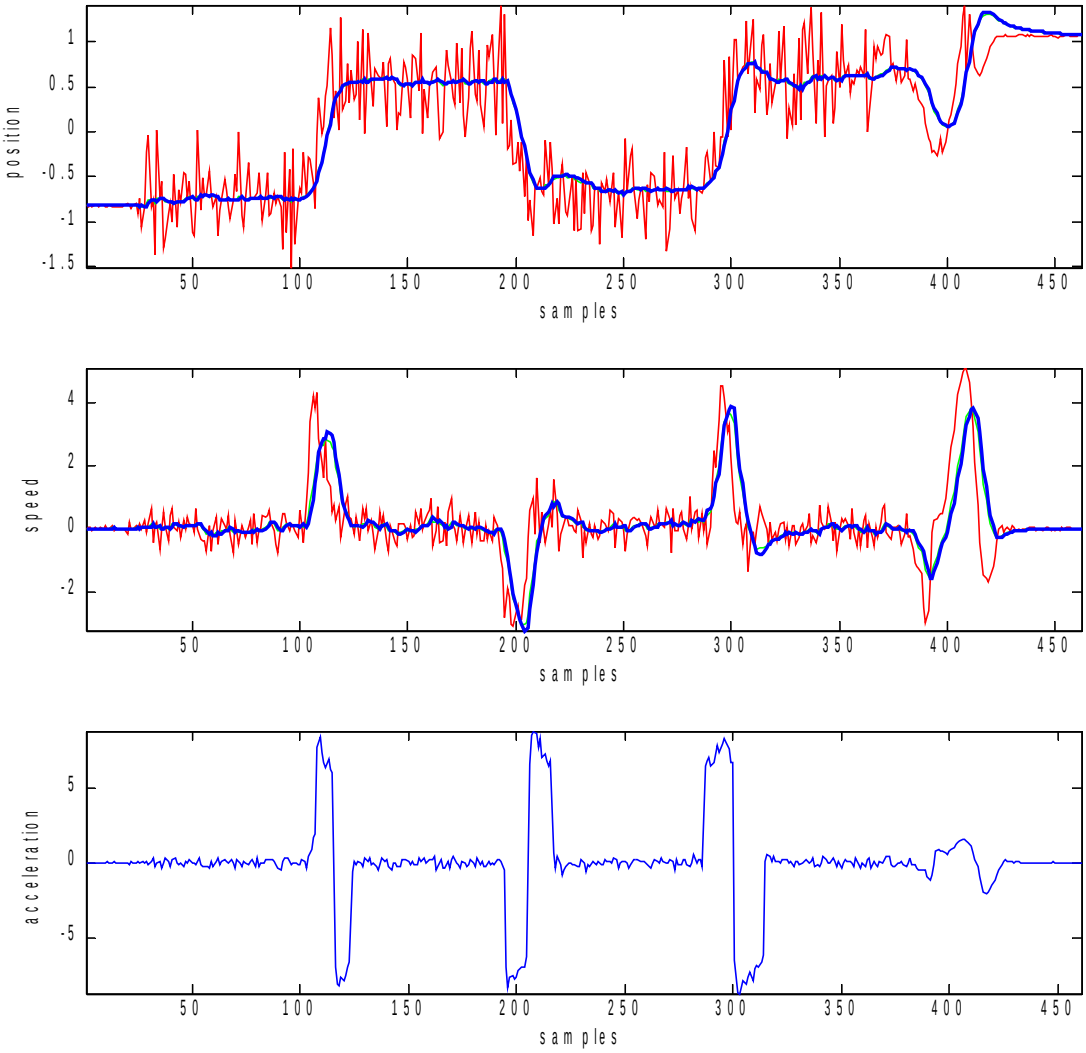


Figure 26. Measurements (red) and predicted states (blue) vs. samples, rotors on

The effect of the rotors rotation is evident and the signals are no longer usable without previous processing. Here, values of $\alpha = 0.05$, $\beta = 0.15$ and $\gamma = 0.10$ where used. The position measurement was more noise susceptible and had to be less trusted to get a good state prediction. Also a slightly bigger gamma proved handy to get a more accurate estimation of the real acceleration and provide better state estimation for position and velocity.

To better appreciate the improvement on the quality of the signal, the position vs. samples graph is once again put in Figure 27, but this time with a black line showing the mean of the predicted signal while the prototype is in the 0.6 rad position and two green lines that show the standard deviation of the filtered signal for that same interval.

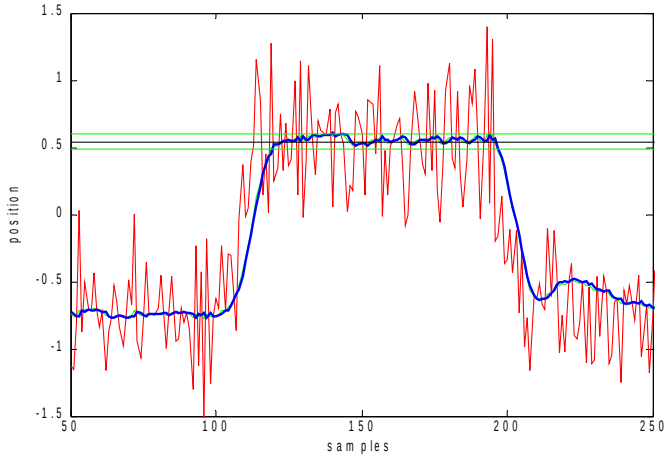


Figure 27. Mean and standard deviation of filtered signal

The ABG filtering proved its effectiveness and the algorithm for the signal processing is chosen. The probabilistic method is the most fit for this application.

3.1.4 Implementation

Now that the controller and signal processing have been discussing, the next step is to explain how was all translate to code and put into the microprocessor. As mentioned before, many parts of the software where already implemented and most of the hardware had already the necessary layers to make them ready to use. Nonetheless, a new less noise susceptible LIS3LV accelerometer needed to be connected. Then Signal processing is put in C code and the controller too. To make tests in the target prototype it was necessary to develop controller for the roll and yaw, so the helicopter could take off.

3.1.4.1 SPI simulation layer

The new accelerometer could be connected via Inter Integrated Circuit interface or a Serial Peripheral Interface, having both hardware modules built on-chip. Regrettably, both modules are disabled in the Robostix module even though the Atmega128 has them. That is because some of the microprocessor's pins used for those interfaces are connected with the JTAG and USART interfaces and are reserved for other purposes. So for this project's realization, a SPI simulation software layer was developed (Attachments simSPI.c and simSPI.h).

The SPI is a slave-master communication normally 4 wire configuration with the possibility to work with only 3 wires. The accelerometer is a slave and it has to receive a clock signal form the master (the microcontroller in this case) to synchronize itself and respond accordingly.

It consists of three functions, initSPI, writeSPI and readSPI. The first one initializes the accelerometer module by writing E7h to the control register one to activate the device, put it in 640Hz operation, enable x-y-z channels and disable self test. Then a 02h is written to control register 2 to enable ±2g mode, continuous update, little endian mode, 3 wire SPI, 12 bit sign right justified data and disable auto reset and data ready signal.

The writeSPI function provides chip select signal through one pin, a 500kHz clock signal through another one and the register address and data to be written with a third pin.

The read SPI function provides a chip select signal through one pin, a 500kHz signal through another one and a third pin serves first as a register output to indicate was is to be read and then as input to read the contents of the register.

3.1.4.2 Signal Processing

The ABG filter was put then in C code (Attachments pitch_ctr.c and pitch_ctr.h). The output of the controller, the control action, has to be known by the ABG algorithm to do a proper signal filtering, thats why it was decided to put in the same function the controller and the signal processing. This function has a series of input parameters:

ki – constant for the controller
kd – constant for the controller
acc_sig – accelerometer signal
des_pTilt – reference value, desired inclination in radians
gyro_sig – ADC converted raw signal coming from one gyro
pA – Alpha constant for ABG filter in pitch controller
pB – Beta constant for ABG filter in pitch controller
pDeltaT – Loop execution time Delta t constant for ABG filter in pitch controller
pG – Gamma constant for ABG filter in pitch controller

The ki and kd constant are part of the controller, so they will be further explained in the next section. The acc_sig is the accelerometer signal coming from the main routine. In the main routine, the accelerometer data read through the SPI is multiplied by a constant called LIS3CONST in the code. This constant is used to convert the raw signal form the accelerometer to units of 1/100 g. To get the value of this constant the operation is:

$$\frac{100\,cg}{1024\,LSB} = 0.09766$$

One hundred cg, or one hundred 1/100 g, are equivalent to 1024 Less Significant Bits (LSB) of a 12 bit signed measurement in ±2g mode. So when the value coming from the LIS3LV accelerometer is read, is then the value obtained in LSBs is multiplied by this constant to get a International System compatible measurement in cg. Later on this constant was fine tunned by getting measurements of the three axis and getting the square root of the sum of squared values to get the absolute value of gravity. It was found that with a value of 0.0950 the absolute value of the geometrically added acceleration measurements was the closest to 100cg.

At the start of the input signal conditioning part, the angular velocity filtered signal pOmega_predicted[0] is integrated to get a current inclination angle pTilt_measured. This integrated value was used as a θ_p instead of directly the accelerometer data because the accelerometer measurements can lead to misinterpretation and false state estimation. This will be further explained in the tests section. The code looks like this:

```
pTilt_measured += (pOmega_predicted[0] * pDeltaT);
```

Then the accelerometer signal is converted to a tilt angle using the inverse sine function and passed through a simple IIR low pass filter using the Alpha value to say how much the value is 'trusted'. Some noisy measurements give as a result more than 100cg, so the inverse sine function returns a Not A Number value. To prevent this, if the measurement could fit into the range of the inverse sine function, a simple Taylor approximation was done, as is well know that $\sin(x) \approx x$.

```
if (acc_sig < 100 && acc_sig > -100)
    pTilt_acc = ((1-pA)*pTilt_acc) + (pA*asin(acc_sig/100));
else
    pTilt_acc = ((1-pA)*pTilt_acc) + (pA*acc_sig/100);
```

Then the measured tilt value is passed through another IIR filter, again using the Alpha value, to let the accelerometer tilt value influence over the tilt state, but only in the long term.

```
pTilt_measured = (pA*pTilt_acc) + ((1-pA)*pTilt_measured);
```

Finally the measured gyro signal is converted to rad/s. In this case the sensibility is of 5mV/°/s, the ADC is in ten bit mode and the full voltage range is of 5V. So the value of the converting constant P_GYROCONST is

$$Resolution_{gyro} \cdot Resolution_{adc} \cdot \frac{360°}{2\pi} = 5\frac{mV}{°/s} \cdot \frac{1\text{LSB}}{10\text{mV}} \cdot \frac{360°}{2\pi} = 28.6479\frac{rad/s}{LSB}$$

That way an estimated tilt angle and angular velocity are found in radians and rad/s respectively. It is important to notice that to integrate the angular speed an already filtered signal was used, the filtering explained in the very next paragraph, and a IIR filtered accelerometer signal.

The Alpha Beta Gamma filter here implemented, has no Alpha, since the position (or angle) value is not directly extracted from a sensor, but from the integration of other signal. The Alpha value is used, as previously explained, to smooth the accelerometer data. First the angular velocity and acceleration values are smoothed using the Alpha and Beta values. Note that the as raw measured angular acceleration, a discrete derivative of the angular velocity is used; and that there is a known pAlpha, or acceleration, witch corresponds to the control action taken at the moment.

```
pOmega_smooth = pOmega_predicted[0] + (pB * (pOmega_measured -
                  pOmega_predicted[0]));
pAlpha_smooth = pAlpha + (pG * ((1/pDeltaT * (pOmega_predicted[0] -
                  pOmega_predicted[1]))-pAlpha));
```

Then the angular velocity predicted value is saved for later use and the angular velocity is finally updated using the smoothened control action value.

```
pOmega_predicted[1] = pOmega_predicted[0];
pOmega_predicted[0] = pOmega_smooth + pDeltaT * pAlpha_smooth;
```

That concludes the signal processing algorithm. It does not strictly obey the ABG filter algorithm, but the changes were done because of the experimental results, and improvements where obtained. After this signal processing, the result are predicted values for the angular velocity and angle of inclination, witch are the two necessary things for the controller.

3.1.4.3 Controller

The implementation of the controller is very straight forward once we have a right estimation of the angular velocity and angle of inclination of the helicopter. The basic step consists in obtaining an angular acceleration value that is congruent with the states and the desired step

response. If equation () is remembered, the implementation is obvious.

$$\frac{d^2\theta_p}{dt^2} = k_i(\theta_{ref} - \theta_p) - k_d\frac{d\theta_p}{dt} \tag{4}$$

and to code it is directly written

```
pAlpha = (ki*(des_pTilt-pTilt_measured)) - (kd*pOmega_predicted[0]);
```

Where the result is a given acceleration in rads/s^2 to be taken as a control action. The next and final step is to convert that acceleration to a servo signal. The value that the function returns is an integer that will be added to a register that indicates the length of the active period of a pulse. So if a positive value is returned, it is added to the register and the duty cycle grows, resulting in a counterclockwise inclination of the rotor and a force that inflicts a positive angular acceleration. If a negative values is returned, the duty cycle decreases and the resulting acceleration is negative. To make a consistent conversion, two constants are needed. First a constant that will convert the control acceleration to a non dimensional quantity to use inverse sine function and obtain a rotor inclination angle, and second a constant that will convert the resulting angle into servo tics.

Remembering equation (13)

$$\theta_r = \sin^{-1}(\frac{r}{g}\alpha) \tag{13}$$

The first constant P_ALPHACONST has a value of

$$\frac{r}{g} = \frac{0.1\text{m}}{9.8\text{m}/s^2} = 0.0102\,s^2$$

The squared seconds will eliminate the units of the angular acceleration and a tilt angle for the rotor will be obtained with the inverse sin function. Then, to convert the angle to tics (LSBs to add to the register), a relation between tics and angle is needed. For this servos, 512 tics give a 45° inclination so

$$\frac{512\,LSB}{pi/4} = 651.8992 \quad LSB/rad$$

Also it has to be thought that sometimes the acceleration value can fall out of the range of the inverse sine function, so it has to be limited. Even more, the way the servos are mounted, an angle of 45° is not desired because in combination with the yaw controller it can go over that limit. So, to avoid damage, the acceleration is limited to +-62.43 rad/s2, or 450 tics. The last lines of code are

```
if(pAlpha < -62.43)
       pAlpha=-62.43;
if(pAlpha > 62.43)
       pAlpha=62.43;

pAlpha_tics = asin(pAlpha * P_ALPHACONST) * P_TICCONST;
```

```
return ((int)pAlpha_tics);
```

And that is how the controller was implemented in C code.

3.1.4.4 Roll and Yaw controllers

To be able to do tests in the target prototype, it was necessary to implement controllers for the roll and yaw axis. That way way it would be possible to do the first flight attempts. To do it, the same model of the plant and controller was used, and the pitch controller code was used as a base. The signals from the other two gyros and axis from the accelerometer were used. The problem is symmetric, since the dynamics of the plant can be modeled likely.

Most all of the variables in all the controllers are global, because they need to conserve the last calculated value for the next control loop. But they are declared inside the respective controller file, so their scope is only inside that file and their value is protected. Since the names are similar, the p's in the pitch controller are substituted by r's in the roll controller and y's in the yaw controller. Same thing is done with the constants even when sometimes they have the same value for different controllers, to keep the code ordered and readable.

First for the roll controller (Attachments roll_ctr.c and roll_ctr.h) the algorithm is basically the same, but the last part is different. This time is not the rotor inclination what is actuated to make the control maneuver, is the rotor speed what is modified. There is a main rotor speed commanded by the user using the RC controller, but the difference between the speed of the rotors is set by the roll controller. When a rotor is faster than the other one, an angular acceleration in the roll axis is inflicted and that is what works as a control action (Figure 28). There is also a torque in the yaw axis inflicted by the varying speed of the rotors, but it is expected to be compensated by the yaw controller.

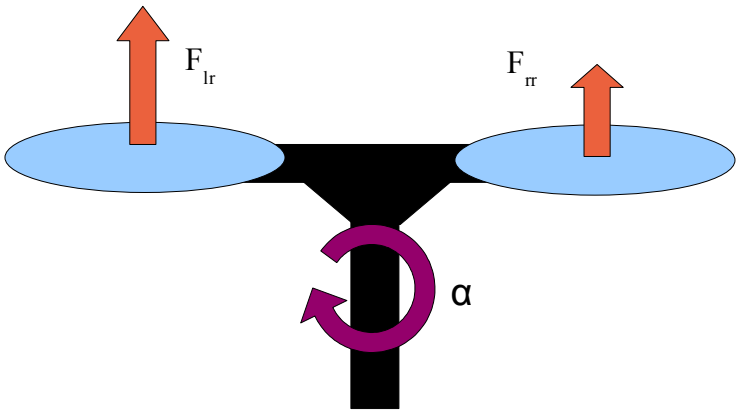


Figure 28. Model seen from the front

So, the last part of the roll controller code uses a square root function instead of the inverse sinus function because it is the function that better fits the pulse rotor speed vs. rotor force curve. A new constant R_TIC_HOVER had to be found to adapt the acceleration value found to a right number of LSBs for the rotor speed servo controllers. Since the speed of the rotors is proportional to the pulse width and for a hover flight

$$F_{hover} = mg = C_1 \omega_{hover}^2 \quad , \qquad \omega = C_2 LSB$$
$$\Rightarrow C = \frac{mg}{LSB_{hover}^2} \tag{14}$$

And a controlling force has to be calculated

$$F_{ctr} = \alpha \frac{I}{r} = C \cdot LSB_{ctr}^2 \quad , \qquad I = mr^2$$
$$\Rightarrow LSB_{ctr} = \sqrt{(\frac{\alpha m r}{C})} \tag{15}$$

Substituting C in (15) with (14)

$$LSB_{ctr} = LSB_{hover} \sqrt{(\frac{r}{g} \alpha)} \tag{14)+(15)=(16}$$

So R_ALPHACONST is equal to P_ALPHACONST, that is r/g; and the new constant R_TIC_HOVER has to have a value equal to the number of LSBs needed to sustain a hover flight. Experimentally the value found was of 250 tics.

That way the last lines of code for the roll controller are the following:

```
if(rAlpha < -15)
    rAlpha=-15;
if(rAlpha > 15)
    rAlpha=15;

rAlpha_tics = sqrt(abs(rAlpha) * R_ALPHACONST) * R_TIC_HOVER;

if (rAlpha < 0)
    rAlpha_tics *= -1;


return ((int)rAlpha_tics);
```

The maximum value is limited to 15 rad/s^2, or 214 tics, to avoid extreme differences between rotor speeds that could jeopardize the stability of the aircraft. Also, the square root operation is done without the sign of the acceleration to avoid non valid operations and then the sign is recovered before returning the value.

For the yaw controller (Attachments yaw_ctr.c and yaw_ctr.h) some other things had to be changed. A less sensible gyro is used and Y_GYROCONST takes a value of 71.6198 LSB/rad/s. Also the radius is bigger (20cm) so Y_ALPHACONST is of 0.0204.

This controller sets the differential cyclic tilt, that is the difference between the inclination of the two rotors. If a non zero difference is set, then one rotor will be more tilted than the other, causing a resulting torque in the yaw axis. Figure 29 illustrates this phenomena.

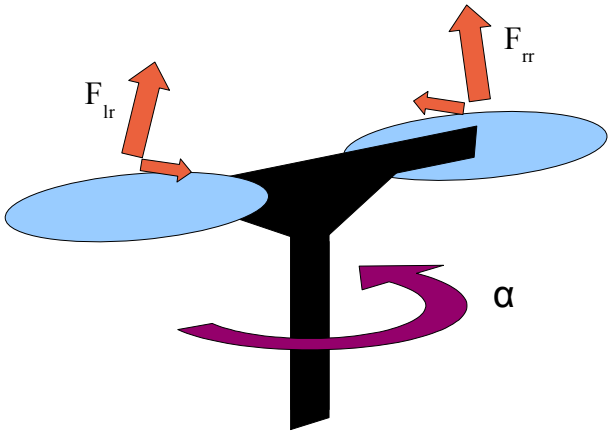


Figure 29. Duocopter model seen from a ¾ perspective

Here the Duocopter is seen form a ¾ perspective. The yaw axis is the one that goes along the vertical black rectangle. Lets suppose the pitch controller is disabled and the common cyclic tilt is 0. Now, if a differential cyclic tilt is set by the yaw controller, the right rotor will tilt forward and it will look like in Figure 29 and the right rotor force will have a component perpendicular to the yaw axis (small red arrow on the left). At the same time the left rotor will tilt backwards and the propeller force will have a component perpendicular to the yaw axis (small red arrow on the right). This two components are parallel and have the same sense in the yaw axis so they will add up in a resulting force that will cause and angular acceleration.

A problem that surges with this controller is that the yaw is perpendicular to the gravity, so it is not possible to use a signal coming from the accelerometer to use as an absolute reference. The way to counter this is using purely the integrated gyro signal. But there is a risk of drift cause by not perfectly calibrated gyro. There is also the problem of having to hold a particular position of the stick in the RC controller to stay in a desired position. To correct for this two problems, the commanded desired position is used directly but also is slowly integrated to establish a modifiable zero reference. If a turn left command is received, the controller will react instantly, but it will also integrate the signal to slowly shift its zero position to the left. This way the drift can be compensated and there is no need to re accommodate the helicopter to be able to take of without turning. The next line do the explained procedure:

```
des_yTilt_zero += des_yTilt / 200;
```

And the controller line suffers a small change to use this shifting zero feature, adding the des_yTilt_zero variable to the calculation:

```
yAlpha = (ki*((des_yTilt_zero+des_yTilt) - yTilt_measured))-(kd
                * yOmega_predicted[0]);
```

That is how the other two controllers where implemented, but they also had to get newly assigned Alpha, Beta, Gamma and DeltaT constants to give a proper behavior to the correspondent axis.

3.1.4.5 Mixer

To put all the control signals together and set the proper values to the PWM registers the need for a mixer function surged (Attachments mixer.c and mixer.h). It simply receives the output of all three controllers as parameters, together with a floating point quantity coming directly from one of the RC channels form the remote. The last value that goes from -50 to 50 sets the user commanded rotor speed. Registers OCR1A, OCR1B, OCR3B and OCR3C are the register that indicate the time period of the active part of the pulse. The first two registers save the time value in milliseconds and are used to control the tilt of the rotors. The other two in 500 microseconds units and are used to control the speed f the rotors. All the pulses are of two milliseconds as the servos demand it. The ROTOR_TILT_OFFSET constant is there to compensate in case the rotor's common cyclic tilt is not perpendicular to the pitch axis when a pulse with 50% duty cycle is sent to the servos. The ROTOR_CORR_GAIN is the value calculated in section 3.1.2.5 to get congruent rotor speeds.

Registers OCR1A and OCR1B get the values from the pitch and roll controllers' output. They both get the output of the pitch controller (pitch_tilt), to set a common cyclic tilt plus/minus half of the yaw controller output (yaw_tilt) to set a differential cyclic tilt. Note that all the signs in the second line are changed because the servos are placed mirrored on the sides of the helicopter.

```
OCR1A = 1500 + ROTOR_TILT_OFFSET + pitch_tilt + (yaw_tilt/2);
OCR1B = 1500 - ROTOR_TILT_OFFSET - pitch_tilt + (yaw_tilt/2);
```

The other two registers just need to have added/subtracted half of the output of the roll controller (roll_tilt) to get a differential roll force. Just one of the two values has to be adjusted with the correction factor to get the right rotor speeds. The times two multiplication is to convert the microseconds units to milliseconds, and the times 16 is to get a maximum total value of 1900 milliseconds witch is the maximum an active (or non active) part of the pulse can last.

```
OCR3B = 1100*2 + (int)(des_rotor_speed * 16) + (roll_tilt);
OCR3C = 1100*2 + ((int)(des_rotor_speed * 16) - (roll_tilt)) *
ROTOR_CORR_GAIN;
```

3.1.4.6 Main

The Duocopter's main function (Attachments PWM.c and pwm.h) does a series of initializations and test before starting the control loop. The USART is initialized and tested; the LIS3LV accelerometer module is initialized and tested; three ADC channels for the gyros are initialized, tested and calibrated; the registers for PWM corresponding to the inclination of the rotors get a zero value; the rotor speed controllers are initialized through PWM; and the RC signals are read an calibrated. The whole process can be monitored through the USART interface.

To initialize the rotor speed controllers, first a pulse with a minimum duty cycle is sent and hold for two seconds. Then the controllers produce a beeping sound indicating that the minimum pulse value has been registered. Afterwards a pulse with a maximum duty cycle is sent and hold for two seconds. Then the controllers produce a double beeping sound that indicate the maximum pulse is registered. Just then, if the signal coming from the remote controller is in its minimum position, a third single beeping sound is produced, indicating that the propellers are in a zero speed, and that any augmentation of the pulse's duty cycle will result in rotor motion. This software initialization is done for two reasons. First because of safety. If the rotors would start whenever they are turned on and receiving a more than minimum pulse,

it would be easy to get hurt every time you switch the helicopter on. With this controller required initialization, you insure you have the tame to leave a considerable distance between your hands and the rotors. The second reason is to insure equal initialization for both controllers. They register the value they get as a minimum duty cycle for two seconds as a reference, and they so the same with the maximum. So the two controllers can get unequal working ranges if the initializing pulses are previously processed, causing the rotor speed coupling done before to become useless and reducing the controller's routine effectiveness.

In the same file where the main function is located, there are two Interrupt Service Routines implemented. They are both dedicated to the RC signal capture. The most important of this subroutines insures frame synchronization and reads the channel individually. Since the communication is critical and getting wrong readings can cause severe lost of stabilization and lead the helicopter to a crash, this is the only functionality implemented with interrupts.

In the last part of the main function, and infinite loop is written. There measurements of all the sensors are done, the signals coming from the remote controller are manipulated to get reference values for the controllers and to modify the controller parameters if needed. In the end the controller functions are called and the produced output is sent as parameter to the mixer. That way a three axis control is achieved. Once the program is running, an average of 127Hz is registered with the three controllers running. Some of the parameters had to be adjusted to this value, as the time interval is changed to DeltaT=1/127Hz.

3.1.5 Tests

Several tests where run throughout the development of the project. Once all the hardware was functioning together and there was an implementation of the helicopter's controller, a test of was run in the test prototype. Then when the signal processing was ready, new test where done and tunning of the parameters was required. Finally, the most exhaustive tests where done with the target prototype, once there was a software version with the three controllers and the mixer on it.

3.1.5.1 Prior to Signal Processing

With all the hardware running and mounted in the test prototype, some test where done. At the time, two controllers implemented in the microprocessor, both for the pitch axis. As mentioned in the very beginning, there was a previously implemented digital PID controller. There was also the newly design controller. The goal of this first test was to compare both controllers and decide whether they were good or not for the next phase of the project.

Once everything in place, both controller seemed to respond accordingly when moving the test prototype with the hand while the rotors were turned off. If the prototype was spinning clockwise, both controllers responded with a counterclockwise inclination of the rotors, proportional to the speed. If the prototype was stopped in a position different to the reference, the rotors where commanded by the controllers to an inclination that would exert a propeller force that would bring the prototype to the reference position.

The next step was to turn the rotors on, and see what was the effect of the noise in the sensors and what it caused to the controller response. AS it was expected, the controller became noisy and the goal was to get the prototype to move around a fixed point set using the reference. The movement wasn't a clean oscillation, and the reason is because the combination of the noise sources and measurement noise caused the noise to have components in all the

frequencies.

The PID controller had many troubles to stabilize while functioning with the rotors off. The controller seemed to be responsive to the changes of speed and stay moving around an apparently stable point. But the problem was that that that point was not that of the reference. If an inclination of 0 degrees was commanded by the user through the remote controller, it was not really reach by the helicopter automatically. It had to be somehow force by the user with extreme reference modifications. Further more, even if the prototype moved around the reference point, it drifted slowly and had to be forced back to the initial point.

The new developed controller had a better response. It reacted accordingly to commanded references, and moved over around the desired point without drifting. Also, the response was faster in comparison with the PID controller. This controller has the advantage of integrating two related signals (acceleration and angular speed) in a single controller, while the other one is just a PID for each signal to try to stabilize it independently.

Following the results, it was decided to only continue to develop the new controller, and leave the PID controller aside.

3.1.5.2 After signal Processing

The next step was to come up with a signal processing algorithm that improved the quality of the signals in noisy conditions and allowed the test prototype to stabilize in a more fixed point. Once the Alpha Beta Gamma filter was implemented in C and uploaded in the microprocessor's memory, tests were done.

This time the movement around the reference point was reduced significantly but the results where quite not as good as expected according to what was saw in the MatLab filtering process. So the parameters had to be adjusted. The gamma value turned out not to be very critical because the commanded and measured acceleration where very congruent, and a small value of around 0.05 to avoid noisy estimations worked out good. If very low values for alpha and beta were set, the response was slowed down. One of the things that came up while adjusting the parameters is that an oscillation can be induced when the DeltaT parameter is not well adjusted. That causes the prediction to be delayed, causing a positive feedback that resulted in a more or less clean oscillation. Also, if a very fast response was demanded by giving high values to the ki and kd constants, the test prototype oscillated again. This time because of the delay cause by the response speed of the servos, witch were not able to keep up with the motion of the prototype and commanded inclination.

On the other hand, a very positive result of this phase's results is that the controller was responsive and much less noisy when it had properly adjusted. Furthermore, it reacted accordingly to the proportions of the ki and kd constants, giving a low pass response with a good time response and no overshoot or a slightly under damped response with improved time response and a moderated overshoot.

To improve even more the noise filtering, the signal processing was modified to be an Adaptive Alpha Beta Gamma (AABG) filter. It was thought that one of the main disadvantages of the ABG filter is its lack of adaptability. Inspired by the Kalman filter, a simple adaptability feature was added, where the equivalent of the covariance, parameters alpha and beta, change their value whenever it is considered to have a good state estimation. So, whenever the prototype is close to the stable position and has a close to zero angular speed, no abrupt changes are expected and the alpha and beta can be reduced to get a smoother signal. If a

measurement is found to be outside a certain threshold for any of the two states, there is a high probability that there was a significant change of one of the state, so the measurement has valuable information and bigger values for alpha and beta are set again to give a swift response. This was code in a few C lines, and experimentally was found that a reduction to one fourth of the original value for the alpha and beta parameters when within a 0.1 rad and 0.5 rad/s range gave a good smoothing.

```
if(fabs(pTilt_measured - pTilt_predicted) < 0.1)
    pA = pA / 4;
if (fabs(pOmega_measured - pOmega_predicted[0]) < 0.5)
    pB = pB / 4;
```

With this AABG filter, the response of the controller was fairly improved, giving a was very good and smooth stability around the reference point., but there was still some low amplitude, very low frequency noise present, witch is believed was originated by some non linearities introduced by the test prototype structure, such as hysteresis and resistance to turn in the axis tube, and non perfectly balanced center of mass. So it was decided that the result was satisfactory. The pitch controller functioned correctly and could now be tested in the target prototype.

3.1.5.3 In the target prototype

Controllers for the other two orientation axis were implemented and everything was ready for the tests in the target prototype (Figure 30).

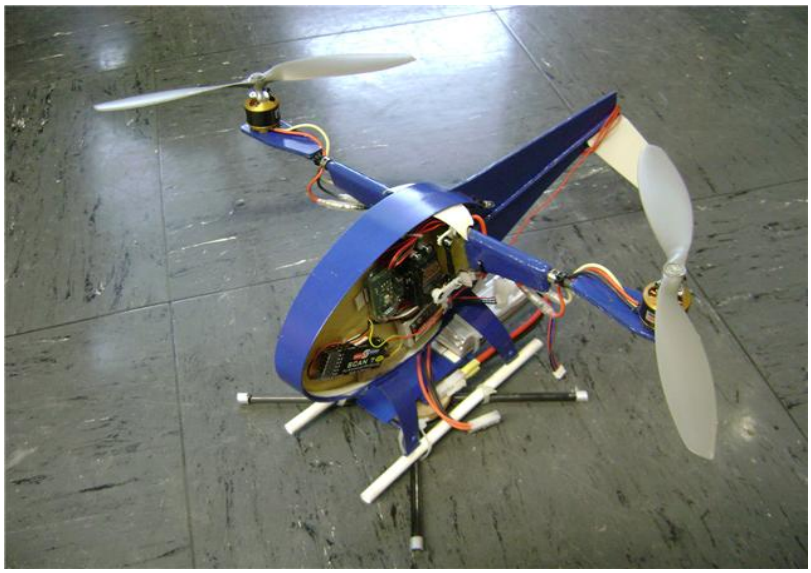


Figure 30. Target prototype ready for tests

There where some difficulties at the beginning because It was hard to tell witch of the parameters from witch of the axis was the one to be adjusted to improve the stability. Also, here is when the it was decided to integrate over time the gyro signal and combine it with a filtered accelerometer signal to get a tilt reference instead of the just the filtered accelerometer signal. That was because when the helicopter moved forward, the accelerometer sensed that

movement, but combined with earths attraction. Then the controller misinterpreted this acceleration going forward as an inclination of the helicopter backwards, because both signals are identical for the corresponding axis of the helicopter. The output of the controller is then that to set a common cyclic tilt forward, witch resulted in more acceleration in that sense. This positive feedback made the aircraft very unstable. Since it is impossible to distinguish between static acceleration (gravity) and dynamic acceleration (displacement of the center of mass), the over time integrated gyro signal had to be the main reference.

Problems arise because there is no x-y-z coordinates controller and it is very sensible to changes in he reference, causing it to drift from one side to the other. If tests were done inside, they had to be quickly aborted because (even before taking off) the drift in the x and y directions took the helicopter close to the walls. If test were done outside, the wind caused more drift and there risk to hit a car existed too.

Finally after multiple tests and implementing a small algorithm in the main function to be able to control the parameters while using the helicopter, an orientation steady flight can be achieved. See Adjusting Parameters attachment to see how they can be adjusted.

3.2 Professional products obtained

There are mainly two professional projects obtained from this project.

A model for a controller that works independently for any of the axis of the helicopter and requires low computation power. The controller itself is almost non load dependent and has very low rotor speed dependency. It forces the plant to behave as a second order system, so theoretical stability is guaranteed because both poles of the transfer function are always in the left side of the complex plane. The controller is in C code, and explained in this document in detail, so it can be easily implemented in any other platform and adapted for other type of aircrafts.

An Adaptive Alpha Beta Gamma state estimator that effectively reduces noise and has very good performance. The algorithm is implemented in an .m file and can easily be tested and used for other applications. It is also implemented in C code using only standard functions and is very readable. All the operations are in floating point, so the code should have good performance in any modern microcontroller but it can still be change to fixed point logic for optimization purposes if thats' the case.

3.3 Technical results and conclusions

An program capable of stabilizing the orientation of the Duocopter was developed, with the disadvantage of strong parameter linear dependency witch results in the need of fine parameter tunning. A fine tune up becomes really difficult when the only way to do it is to manually modify the parameters according to what was observed while commanding the Duocopter to fly and trying to keep it out of danger at the same time. Other means of tunning will have to be found to make the most out of the controller.

Part II Social afterthought

Chapter 4. Personal and Social benefits of the project

4.1 Expected and obtained benefits throughout goal achievements.

4.1.1 In favor of people involved in the project

Through the development of this project I improved my abilities as an Engineer. I learned about the organization and requirements of a complex and broad project. I learned to work in a research and development atmosphere, to continue someone else's work and to manage the responsibility of having in my hands an important part of the project.

Professor Steiper gets to further develop his project and now contemplates the possibility of inviting more people form Iteso to continue it. For Professor Cotero, there is now the possibility to do more projects in cooperation with the Hochschule, weather be it sending more students to Germany, or to welcoming students from Ulm to participate in projects in Guadalajara.

Personally, there are other benefits obtained through the development of this project.

With the completion of this project, I fulfill all the requirements to receive my degree as an Electronic Engineer in Iteso University. As a proof, I will receive a Diploma, and this significant document represents all that I have been through, all that I have learned about science, self formation, companionship, society and life. My title will backed up by a recognized institution as the Iteso, and it will be proof too, that my personal values are in concordance with those of Iteso. Getting this document does not mean I learned all there is to learn in my career, the nature of the career itself is to always keep learning, to always find new ways to solve problems. Now as a conclude this important era in my life, I can say that I have the right tools to perform as an Engineer and that have the criteria to use them responsibly.

I previously participated in a student exchange program in the Hochschule. This motivated me to do my professional application project here. Through its development, my bonds with people in the field are strengthen, my knowledge on the area is broaden and I know better now what are my fields of interest. Because of this, the possibility of further academic formation will be an important option for me in the future.

4.1.2 In favor of social groups

Se closest social group benefited by this project are the students of both participating institutions. A new door opens in the cooperation process of the universities, and the possibility of students going to the partner university to accomplish a project of the kind is now a reality. This enriches the mutual knowledge and the recognition of the engineering and social reality of the foreign university directly influences the perspective of the soon to be engineers. Thus, their personal development possibilities grow.

The contents of this paper could be used to model solutions for similar applications. It may also show the areas where new courses could be developed for the engineering plan in Iteso. Areas as robotics, aeronautics and modern signal processing and control are not widely explored

in the courses, but there seems to be interest and the students have the right bases.

Another benefited social group is the people that will be serviced or assisted by a helicopter or other robot that comes from this project. As mentioned in the beginning, the product of this project are to be used to help people.

Final conclusions and recommendations

Parting from the findings of this project, there are many things that can be done to continue the development of the Duocopter. Other approaches to tune the parameters in a less empirical way could be helpful. A different hardware architecture could possibly bring benefits. Also the next logical step concerning the x-y-z coordinates can help to make the most of the controller developed in this part of the project.

If an interface that represents not much overhead to the present program could be mounted, it would be possible to make a dynamic monitoring an manipulation of the parameter of the controller, and it would be much easier to accomplish a fine tunning and find other possible problems. For example, a graphical interface could be developed in a PC to graphically monitor the state estimation and appreciate the magnitude of the noise present. That same interface could be used to modify the parameters with complete freedom, without need to recompile and reprogram. An extreme case could me modify the MatLab model (or make a Labview one) to work as the controller itself and make all the computation in a PC, while the equipment in the helicopter just serves as communication, sensing and actuation hardware. Bluetooth and Zigbee are examples of such interfaces capable of providing low overhead wireless communication, suitable for this application. An automatic tunning algorithm is also an interesting option, but it is much more complicated and a more powerful microprocessor may be needed.

Another possibility for the future is to adapt the controller to function as a a Flatness Based[9] or an Intelligent PID[10] with a fast algebraic estimation approach controller. The first option would help to compensate for non linearities that could make the behavior of the helicopter more parameter sensible, and the second one to achieve automatic and dynamic parameter adjustment within the controller itself. In both cases, the signal processing algorithm here developed can be integrated to achieve maximum performance.

In the future more computation power will most probably be needed. If a multiprocessor architecture is implemented, task division could help reduce sensibility on some of the time dependent parameters. In a single core architecture, a Real Time Operating System could do the same.

A 'must' develop part of the project is a state estimator for the x-y-z coordinates. The development of such estimator, and a corresponding controller for the same coordinates will help to observe if a fine tunning of the parameters is enough for a longer flight or a more general approach is needed. It will also help for the tunning of the orientation controller itself, because the user would have to worry less about the safety of the aircraft.

9 *Nonlinear Vehicle Dynamics Control – A Flatness Based Approach*, Stefan Fuchshumer, Kurt Schlacher and Thomas Rittenschober, 44th IEEE Conference on Decision and Control, 2005

10 *Intelligent PID Controllers*, Michel Fliess and Cedric Join, 16th Mediterranean Conference on Control and Automation Congress Centre, Ajaccio, France 2008

References

Cities of the Future, FDi magazine(2007-04-23). p.2

Probabilistic Robotics, Sebastian Thrun, Wolfram Burgard, Dieter Fox, p. 5, 2005 p.19

Probabilistic Robotics, Sebastian Thrun, Wolfram Burgard, Dieter Fox, p. 17-27, 2005 p.20

A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,Lawrence R. Rabiner Proceedings of the IEEE, 77 (2), p. 257–260, February 1989 p.20

A new approach to linear filtering and prediction problems, Kalman, R.E. (1960). Journal of Basic Engineering 82 (1) p.20

Kalman Filtering, Dan Simon, Embedded Systems Programming, June 2001, p.72-79 p.20

Probabilistic Robotics, Sebastian Thrun, Wolfram Burgard, Dieter Fox, p. 40-54, 2005 p.22

Reconciling Steady-State Kalman and Alpha-Beta Filter Design, JOHN H. PAINTER, Senior Member, IEEE Xxas A & M University DAVID KERSTETTER, Member, IEEE General Dynamics, Pomona Division STEVE JOWERS McDonnell-Douglas Astronautics Co. p.22

Nonlinear Vehicle Dynamics Control – A Flatness Based Approach, Stefan Fuchshumer, Kurt Schlacher and Thomas Rittenschober, 44th IEEE Conference on Decision and Control, 2005 p.41

Intelligent PID Controllers, Michel Fliess and Cedric Join, 16th Mediterranean Conference on Control and Automation Congress Centre, Ajaccio, France 2008 p.41

List of Figures

Attachments

Duocopter Presentation.ppt

A power point presentation that covers the technical part of the development of the controller of the orientation of the Duocopter.

Duocopter_non_linear.mdl

A Simulink model of the plant and controller of the orientation of the Duocopter. Parameters ki and kd need to be initialized in MatLab.

AlphaBetaGamma.m

MatLab implementation of the Alpha Beta Gamma filter.

simSPI.c and simSPI.h

Code to simulate a SPI layer in the Atmega128. Is application specific, so if it wants to be used for other purposed it may need to be reviewed and expanded.

pitch_ctr.c and pitch_ctr.h

Code to process the signals coming from the sensors and calculate the PWM output to achieve control of the pitch axis.

roll_ctr.c and roll_ctr.h

Code to process the signals coming from the sensors and calculate the PWM output to achieve control of the roll axis.

yaw_ctr.c and yaw_ctr.h

Code to process the signals coming from the sensors and calculate the PWM output to achieve control of the pitch axis.

mixer.c and mixer.h

Code to set the correct values to the PWM registers according to the output of the pitch roll and yaw controllers.

PWM.c and pwm.h

Main function of the Duocopter's program.

Adjusting Parameters

Text that explains the procedure to best adjust the parameter of the different parts of the controller of the orientation of the Duocopter.

Duocopter before fine parameter tunning.avi

Video of the Duocopter without precisely adjusted parameters.

Duocopter after fine parameter tunning.avi

Video of the Duocopter flying, with precisely adjusted parameters.