ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

II FACOLTA' DI INGEGNERIA

Dipartimento delle Costruzioni Meccaniche,
Nucleari, Aeronautiche e di Metallurgia

DOTTORATO di RICERCA
IN DISEGNO E METODI DELL'INGEGNERIA INDUSTRIALE

CICLO XIX

Progetto e realizzazione del sistema di gestione autonoma del volo e controllo in remoto per un velivolo UAV ad ala rotante.

S.S.ING-IND/03 MECCANICA DELVOLO

Coordinatore: Chiar.mo Prof. Ing. Franco PERSIANI

Relatore: Prof. GianMarco SAGGIANI

Dottorando:

Ing. Barbara TEODORANI

Esame Finale Anno 2007

On the Development of a Rotary Wing UAV Platform:

Avionics and Onboard Software Set-Up

PhD Thesis

by

cand. Barbara Teodorani

S.S.ING-IND/03 FLIGHT MECHANICS

Coordinator: Chiar.mo Prof. Ing. Franco PERSIANI
DIEM-University of Bologna

Advisor: Prof. GianMarco SAGGIANI
DIEM-University of Bologna

ALMA MATER STUDIORUM
Department of Mechanical, Nuclear and Aerospace Engineering
II Faculty of Engineering
DIEM– University of Bologna
2007

ACKNOWLEDGMENTS

Expert people in UAV development know well that it is above all a science of integration of different disciplines, skills and know-how. No successful results could be achieved without the precious contribution of numerous people working together. That's why we, me and Roberto Pretolani, are very grateful to all people working with us, during the years we spent at the Hangar Laboratories of the University of Bologna.

In the first place, we would like to thank Professor Ing. Franco Persiani who, for first, supported and believed in the rotary UAV project and Prof. Gianmarco Saggiani for the rotary team coordination.

But we cannot forget also the support of:

Ing. Veronica Rossi: support in onboard software implementation

Ing. Filippo Zanetti: rotary team new entry and support in autonomous flight tests

Mauro Ricci and hangar technicians (Luciano and Ivano): fun club & support in helicopter mechanics

Stefano Lucchi: fantastic RC Helicopter pilot

Antonio Francia: enthusiastic support in helicopter engine set-up

Ing. Tiziano Bombardi: support in C++ program language and his "precious UDP.dll"

Prof. Alessandro Rivola: support in vibration tests and analysis

Prof. Gian Battista Garito for his interesting lessons on helicopter dynamics theory

Ing. Matteo Zanzi: support in flight control and navigation theory

Fodias Guys: experience exchange talks and test rig

Don Diacunu Pietro: helicopter flight test football field

Ing. Stefano Saputo and Ing. Fabio Antonini: advices in onboard electronics set up

Ing. Stefano Mazzoni: support in NI Hardware choice

Ing. Alessandro Boccalatte: visual system design

The CAPECON Rotary Team People: Stephen Mouritsen, Jan Floris Boer, Marzio Luigi Preatoni, Cyrille Sevin

TABLE OF CONTENTS

TABLE OF CONTENTS ---- *Continued*

LIST OF FIGURES

LIST OF FIGURES ---- *Continued*

LIST OF FIGURES ---- *Continued*

LIST OF FIGURES ---- *Continued*

LIST OF TABLES

ABSTRACT

This PhD thesis presents the results, achieved at the Aerospace Engineering Department Laboratories of the University of Bologna, concerning the development of a small scale Rotary wing UAVs (RUAVs).

In the first part of the work, a mission simulation environment for rotary wing UAVs was developed, as main outcome of the University of Bologna partnership in the CAPECON program (an EU funded research program aimed at studying the UAVs civil applications and economic effectiveness of the potential configuration solutions). The results achieved in cooperation with DLR (German Aerospace Centre) and with an helicopter industrial partners will be described.

In the second part of the work, the set-up of a real small scale rotary wing platform was performed. The work was carried out following a series of subsequent logical steps from hardware selection and set-up to final autonomous flight tests.

This thesis will focus mainly on the RUAV avionics package set-up, on the onboard software development and final experimental tests.

The setup of the electronic package allowed recording of helicopter responses to pilot commands and provided deep insight into the small scale rotorcraft dynamics, facilitating the development of helicopter models and control systems in a Hardware In the Loop (HIL) simulator. A neested PI velocity controller[1] was implemented on the onboard computer and autonomous flight tests were performed. Comparison between HIL simulation and experimental results showed good agreement.

[1] This PhD work was carried out in full cooperation with Roberto Pretolani, who was mainly responsible for the helicopter dynamic identification and the control system design. His PhD thesis [19] will report this work in details

NOMENCLATURE

u	helicopter longitudinal speed in body axes
v	helicopter lateral speed in body axes
w	helicopter vertical speed in body axes
p	roll rate in body axes
q	pitch rate in body axes
r	yaw rate in body axes
φ	Euler angle for helicopter roll
ϑ	Euler angle for helicopter pitch
ψ	Euler angle for helicopter heading
Ψ	blade azimuth angle
Θ	blade pitch angle
Q	the main rotor torque
T_{TR}	tail rotor thrust
F_{AF}	fuselage aerodynamic forces
F_{AT}	tail surface aerodynamic forces
Ω	rotor speed
β	blade flapping angle
Θ_0	average blade pitch angle
A_1	lateral cyclic blade pitch
B_1	longitudinal cyclic blade pitch
B_{lat}	lateral stick to cyclic pitch gearings (effective lateral control derivatives taking into account the effect of the stabilizer bar)
A_{long}	longitudinal stick to cyclic pitch gearings gearings (effective longitudinal control derivatives taking into account the effect of the stabilizer bar)
δ_{coll}	collective control input
δ_{lat}	cyclic lateral control input
δ_{long}	cyclic longitudinal control input
$\partial \dot{h} / \partial \delta_c$	rate of climb vs collective
$\Delta\theta_{level\_flight}$	delta pitch attitude in level flight relative to flight path
$\Delta\theta_{extended\_approach}$	delta pitch attitude in the energy extended approach relative to flight path
$\delta_{c\ max}$	maximum collective
$\delta_{c\ trim}$	trim collective
V	forward speed
E	total energy
M	mass
g	gravity acceleration
h	altitude
V_N, V_E, V_D	velocities in the north (N), east (E), down (D) reference frame
Vx, Vy, V_z	velocities along the trajectory
T_h	PWM high time
f_s	PWM frequency
T	PWM period
d	PWM duty cycle
ω_{nq}	natural frequencies of the longitudinal of the fuselage-rotor-bar modes
ω_{np}	natural frequencies of the lateral of the fuselage-rotor-bar modes
τ_e	flapping motion rotor time constant including the effect of the stabilizer bar.
X_{Vx}	longitudinal speed derivative
Y_{Vy}	lateral speed derivative
Z_{Vz}	vertical speed damping derivative

Z_{coll}	vertical control derivative

Acronyms

CAPECON	Civil uav APplications & Economic effectivity of potential CONfiguration solutions
DLR	Deutches Zentrum für Luft-und-Raumfahrt
RUAV	Rotorcraft Unmanned Aerial Vehicle
UAV	Unmanned Aerial Vehicle
UNIBO	UNIversity of BOlogna
PID	Proportional – Integral- Derivative
PI	Proportional – Integral
EU	European Union
AV	Air Vehicle
DL	Data Link
DD	Data Distribution
GCS	Ground Control Station
COTS	Commercial Off The Shelf
NGCS	Navigation Guidance Control System
LAN	Local Area Network
VLAB	Virtual reality LABoratory
SIT	Simulation Interface Toolkit
SDK	Software Development Kit
FF	Force Feedback
WP	WayPoint
USGS	US Geological Survey
SRTM	Shuttle Radar Topography Mission
DEM	Digital Elevation Map
PWM	Pulse Width Modulation
CRIO	Compact Reconfigurable Input Output
AHRS	Attitude Heading and Reference System
SITL	Software In The Loop
HIL	Hardware In the Loop
RF	Radio Frequency
RC	Radio Control
EMI	Electro-Magnetic Interference
EM	Electro-Magnetic
FPGA	Field Programmable Gate Array
IMU	Inertial Measurement Unit
GPS	Global Positioning System
AMSL	Above Mean Sea Level
AGL	Above Ground Level
SDA	Sensor Data line
SCL	Sensor CLock line
MSB	Most Significant Bit
LSB	Least Significant Bit
HL-AVCS	Heading Lock Angular Velocity Control System
DC	Direct Current
SISO	Single Input Single Output
SP	Set Point
PV	Process Variable
DI	Digital Input
DO	Digital Output
GUI	Graphical User Interface

Chapter 1

MOTIVATION AND BACKGROUND

1.1 OVERVIEW

The increasing interest in military Unmanned Air Vehicles (UAVs) is fuelling an equally ambitious build-up in the civil community. It is well known that UAVs may represent a promising and cost-effective alternative to manned aircraft for a large number of civil applications [1]. Compared to traditional air vehicles, UAVs may offer significant advantages in terms of human safety (especially in dull, dirty and dangerous missions), operational cost reduction and work rate efficiency. Nevertheless, while research activities in UAV or Rotary Wing UAV systems are very advanced in the United States, UAV interest in Europe has begun only in the last years. As a result, in year 2001, the European Union has sponsored the UAV development program CAPECON, to attempt to kick-start a civil UAV industry in Europe and try to fill the gap with the United States. Since 1999, the University of Bologna (UNIBO) has carried out several research projects concerning the development and manufacturing of fixed wing UAV systems for the civil aviation market. For that reason, when the EU decided to start the CAPECON program, UNIBO didn't hesitate to take part in. CAPECON (Civil UAV Application end Potential CONfiguration solution) was the first European wide program tying together the resources of eight countries, nine industrial organization, five aerospace centres and six universities. Its main goal was to provide European industry with detailed design and manufacture know-how on safe cost-effective and commercially viable civil UAVs. The program was structured to make a logical progression

from customer needs to final products [2]. The process (see figure 1) started with UAV applications analisys, braking them down into discrete missions which were then lumped into multirole missions [3,4].

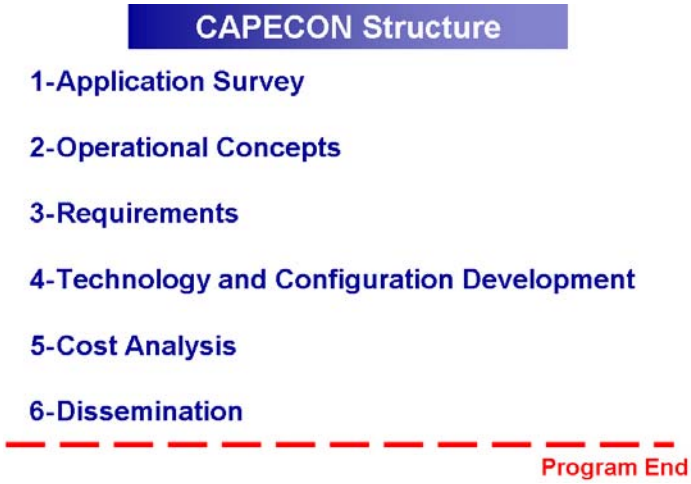


Figure 1: CAPECON Structure

The most promising and commercially viable multirole missions were selected and further translated into operational concepts, which lead to the definition of formal requirements [5]. The formal requirements were used to help defining five fixed wing and two rotary wing configuration which best fulfilled the requirements. The configuration were then designed and the related technology were also identified. The costs for the configurations were also estimated and final dissemination of results to the European industry was done. During the CAPECON project, interest in the Small/Mini size UAVs increased considerably, mainly due to the miniaturization of avionics and onboard systems. The project was therefore extended to study also the Small/Mini Configurations. UNIBO played an important role inside the rotary wing part of the CAPECON program both in the Small/Mini and big size Rotary UAVs (RUAV).

The main CAPECON outcomes for the RUAV systems concerned:

- the identification of the most promising applications [6,7]. Examples of the identified applications included: fire surveillance and fire fighting, civilian security, monitoring or close inspection of electrical powerline, pipeline or dam, search part of SAR (search and rescue) missions, agriculture spray etc..[8,9,10]

- as far as technology are concerned, one important aspect derived from the CAPECON program, was the real need to apply already existing, proven and cost effective technology to the UAV world. Therefore, many existing technologies were identified, ready for application in the short and mid term [11-18]
- the complete preliminary design of two rotary wing UAV configuration: one was a conventional main rotor-tail rotor configuration, while the other one was a coaxial rotor configuration

The culmination of the UNIBO CAPECON work was the design of a mission simulation environment for the rotary wing UAVs. The mission simulation environment was used inside the CAPECON for evaluating the two RUAV configuration operational capabilities. The work carried out by UNIBO during the CAPECON program will be summarized in chapter 2.

Based on the work performed in the CAPECON program, an independent RUAV research program was also started at UNIBO laboratories, since RUAV systems may represent an alternative to fixed wing UAVs (or even a more promising solution) for a wide number of civilian applications, due to their versatile flight modes, maneuverability and vertical take-off and landing capabilities.

The goal to be achieved with the UNIBO RUAV research program was to develop a helicopter, capable of autonomous flight, which could be used inside the university as platform for researches in control and navigation laws; meanwhile it should be proposed as technological prototype to industry interested in UAV development and manufacturing.

An UAV system is generally constituted by at least four main integrated sub-systems (see figure2): the air vehicle (AV), the ground support system, the data link and the data distribution [9].

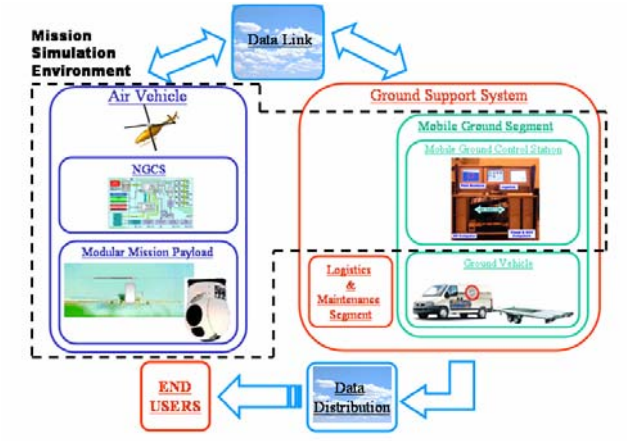


Figure 2: UAV System

- The AV includes all the airborne systems: the basic helicopter platform, the onboard computer and sensors, the mission payload and all the software necessary to guide, navigate and control the air vehicle.
- The ground support system includes all the ground infrastructures and equipments to enable the AV operations, such as a mobile ground control station (GCS), a logistic and maintenance segment and a ground vehicle.
- The data link supports video, data and telemetry communications between the AV and the ground support systems, while the data distribution is able to transmit annotated significant data, collected at the GCS, to potential users at remote locations.

The subsystems, both hardware and software equipments, can be much or less sophisticated, depending on the UAV system size and complexity.

For the purpose of the RUAV program, a small scale hobby model helicopter was used as flying platform, which was certainly a significant physical constraint for the RUAV subsystem equipment choice and development.

The work performed to develop the RUAV platform was carried out following a series of subsequent logical steps:

- first the RUAV hardware (including the onboard avionics, the air vehicle and the data link system) was selected and interfaced, placing attention to vibration isolation, electromagnetic interference and accessibility
- following the hardware set-up, sensor data acquisition software was developed and tested in flight in order to verify sensor measurement reliability. This step plays a crucial role in a RUAV development because, if the helicopter has to fly

autonomously, reliable information about its states is needed by the onboard control and navigation system.

- parallel to the hardware set-up, simulation plays also an important role in the development of an autonomous helicopter. A simulation model was developed, based on helicopter dynamics identification flight tests, to be used for the design of the onboard control and navigation algorithm
- once the previous task were completed, the onboard hardware and software were integrated into the simulation loop using a Hardware In the Loop (HIL) simulator. In this scenario, performance and possible errors of the onboard software can be detected during intensive ground safe and risk free tests
- in the end, autopilot flight tests were performed for final verification and tuning of the control and navigation system.

This thesis will focus mainly on the RUAV avionics package set-up and on the onboard software development, while the other steps will be covered in reference [19].

One important aspect to be taken into account in the development of a RUAV system is that it is, actually, an aerial robot. The set-up of a capable task-worthy aerial robot is essentially an integration effort and, always, requires knowledge of several different disciplines and experimentation on new system development. In the past years, most of the research efforts in miniature autonomous helicopter were lost for hardware integration and for obtaining reliable sensor measurements. For that reasons, taking also into account the outcomes of the CAPECON program, it was decided to evaluate the feasibility of using COTS sensors and electronics for the RUAV avionics package. Both the hardware and the software were integrated placing attention to modularity, growth potential, versatility and possibility for ease reconfiguration and software implementation. Results achieved in this work showed that the selected hardware and the onboard software were able to provide accurate flight data measurements and good helicopter control capabilities. Thanks to its modular architecture and accurate flight data measurement capabilities, the developed RUAV system may become a useful research test bench in several different field such as:

-aircraft /rotorcraft dynamic model identification

-researches in control and navigation laws (fast and ease software implementation could results also in a speed up of the research time)

-researches in man machine interface and air system integration which is addressed as one of the most critical technology aspect for the future development of the civil UAVs and their integration into the airspace [14,16].

1. 2 SUMMARY OF HELICOPTER PRINCIPLES

As well known, helicopters are air vehicles which are able to fly thanks to the lift force produced by lifting surfaces (the rotor blades) rotating about a vertical axis (the rotor shaft). In order to understand the contents of this work, some basic helicopter principles will be introduced. The standard helicopter notation , that will be used in the next chapters, is shown in figure 3.

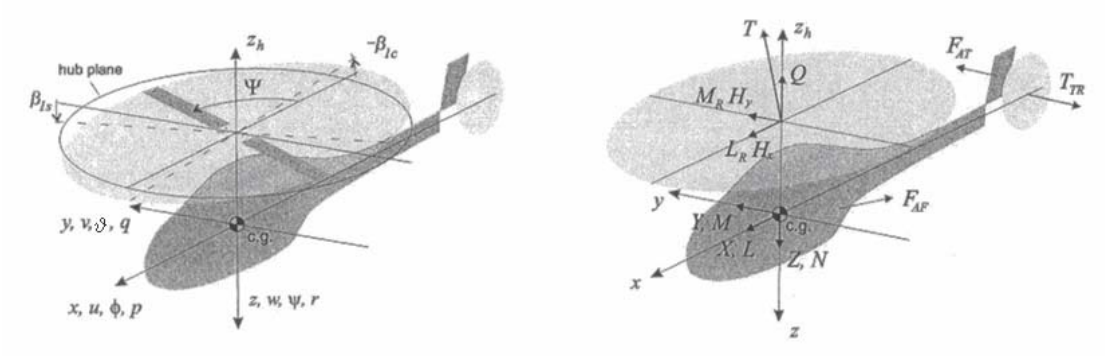


Figure 3: Basic Helicopter Notation [20]

Figure 3 shows the helicopter with its body reference frame; origin is at the helicopter centre of gravity. The principle variables are shown on the x, y and z body axes. They include: the body speeds u, *v*, w, the Euler angle φ, ϑ, *ψ*, the body angular rates p, q, r. The main rotor is represented as a disc that can tilt about the rotor hub in the longitudinal and lateral directions. This motion is describe through the angles β_{1c} and β_{1s} measured in reference to a plane perpendicular to the rotor shaft (hub plane). The actual rotor blade position is described by the angle Ψ measured from the tail (see fig.3). The components of the resultant forces and moment acting at the helicopter centre of gravity are X,Y,Z and L,M,N, respectively. The figure shows also the key forces that contribute to the helicopter motion, including: the rotor thrust T, the longitudinal and lateral rotor moments acting on the hub L_R and M_R, the main rotor torque Q, the in-plane rotor forces H_x and H_y, the tail rotor thrust T_{TR}, the fuselage aerodynamic forces F_{AF}, the aerodynamic forces from the tail surface F_{AT} [20].

Rotor blades are attached to the spinning shaft via a rotor head. Besides its rotation around the hub (speed Ω, position Ψ), the blade can also rotate about three hinges which are shown in figure 4. Feathering is the motion of the blade about its length and is described by the blade pitch angle Θ; flapping is the blade motion in a direction normal to the main rotor disc and is described by the flapping angle β; lead-lagging is the motion of the blade in the rotor disc plane and is described by the angle ξ. Significant variations in the design of the helicopter rotor head exist. Blade motion is enabled by mechanical hinges near the blade root (articulated rotor), by blade root compliance (hingless rotor), or by combination of both.

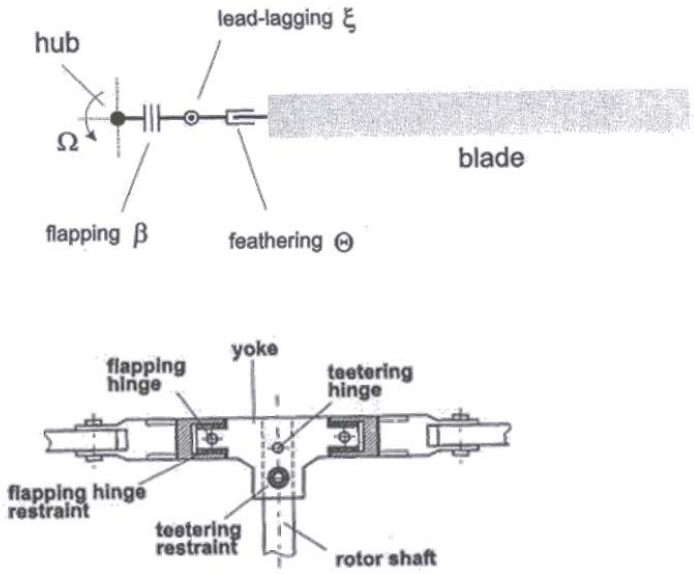


Figure 4: Blade Degrees of Freedom Schematic and Rotor Head Mechanization [20]

In the hobby helicopter used for the development of the RUAV system, there are no actual flapping hinges. The motion about the three hinges is restrained by elastomer fittings that act both as springs and dampers (see fig. 4, elastomeric flapping and teetering hinge). The spring effect is used to transform the blade flapping motion in a hub moment.

In most rotorcraft, the rotor speed is kept constant by an electronic engine governor. The thrust and rotor moments are produced by changing the blade pitch angle. The blade pitch control system is based on a swashplate mechanism. The purpose of this mechanism is to vary the blade pitch both in magnitude but also as a function of the blade angular position around the hub. Using the collective control input, the pilot controls the average blade pitch angle.

The blade pitch angle as a function of its angular position is controlled by the longitudinal and lateral cyclic controls.

The blade pitch angle as a function of its angular position Ψ around the hub (Ψ is zero when the blade is above the tail and it is assumed that the blades rotates counter clockwise; see figure 3) is described by:

$$\Theta(\Psi)= \Theta_0 - A_1\cos\Psi - B_1\sin\Psi \tag{1.1}$$

where

- Θ_0 is the average blade pitch angle, which is set by the collective control input δ_{coll} ,
- A_1 and B_1, the coefficients of the cosine and sine terms, are the amount of blade pitch the blade undergoes when it is located above the tail (-x body direction) and on the right-hand side (y-direction), respectively [20].

A_1 and B_1 are functions of the longitudinal and lateral cyclic controls δ_{lon} and δ_{lat}, respectively. It is possible to rewrite them as a function of linear gearing coefficients which transform the pilot stick input into angular blade-root pitch change:

$$A_1 = B_{lat}\,\delta_{lat} \qquad B_1 = A_{long}\,\delta_{long} \tag{1.2}$$

Usually, on small scale hobby helicopters, a stabilizer bar is also present. The stabilizer bar does not produce thrust (it has no collective blade pitch setting). Instead, the main blade pitch receives both the cyclic pitch servo command and a major component imposed by the stabilizer bar.

Hence, B_{lat} and A_{long} are the effective cyclic control derivatives taking into account the effect of the stabilizer bar.

The primary function of the four principal rotorcraft commands are the following: the main rotor lateral and longitudinal cyclic inputs control the roll and pitch moments produced by the main rotor; collective input controls the magnitude of the main rotor thrust; the tail rotor collective input controls the tail rotor thrust by varying the tail blade pitch. Hence, the commands have a direct effect on the rotorcraft roll and pitch attitude rate, vertical velocity

and heading rate, respectively. The pilot does not control the helicopter position or velocity directly, but via a chain of effects that can be summarized as follows. The cyclic control inputs result in control moments about the rotor hub via a tilting motion of the rotor disc (rotor disc refer to a simplified representation of the combined effect of individual blade motion). The rotor control moments produce a fuselage rolling or pitching motion. If the helicopter is hovering, changing the fuselage's roll and pitch angle will result in a tilting of the rotor thrust vector, producing horizontal thrust components that acts as propulsive force. For example, by holding a constant pitch angle, the helicopter will accelerate until the propulsive force is balanced by the aerodynamic drag force. Of course some cross axis effect are also present. For example, in the longitudinal velocity control, when the helicopter is pitched, the vertical thrust component will decrease, requiring an increase in the thrust magnitude to keep the vehicle at level altitude. This increase in thrust, however, will produce a reaction torque at the rotor shaft that in turn will result in a yawing moment, for which the pilot will need to adjust the tail rotor thrust. Other effects are more subtle, such as the roll responses following cyclic and collective control actions and the pitch responses following lateral control actions [20].

Chapter 2

MISSION SIMULATION ENVIRONMENT

This chapter will describe the mission simulation environment, developed inside the CAPECON program, to evaluate the operational capabilities of the two RUAV configurations designed by the industrial partners (AGUSTA and Eurocopter) [21]. The simulation environment was developed mainly as a tool for supporting the CAPECON industrial helicopter designers in the preliminary design phase of the RUAV systems. In this perspective, it was taken into account that the design of a RUAV system is quite different from the one of a classical manned rotorcraft, since it requires the concurrent definition of its main sub-systems: the air vehicle, the ground support system, the data link and the data distribution. For example, unlike manned helicopter, the helicopter control strategy is one of the most critical aspect in the design of a RUAV. Besides, the ground control station must be also brought into the design space, because it plays a crucial role for the RUAV platform operation. In order to develop such a complex system at industrial level, it was decided, inside CAPECON, that new design strategies are needed, which could be integrated into the design process normally used by the manned helicopter companies [22-28]. In this perspective, the idea was to create a simulation tool able to merge the contributions of different preliminary design working teams (the helicopter preliminary design, the NGCS preliminary design, the GCS preliminary design and payload integration) into a single environment and to test them in cooperative simulation. Such a simulation environment was developed at UNIBO Laboratories and is shown in figure 5.

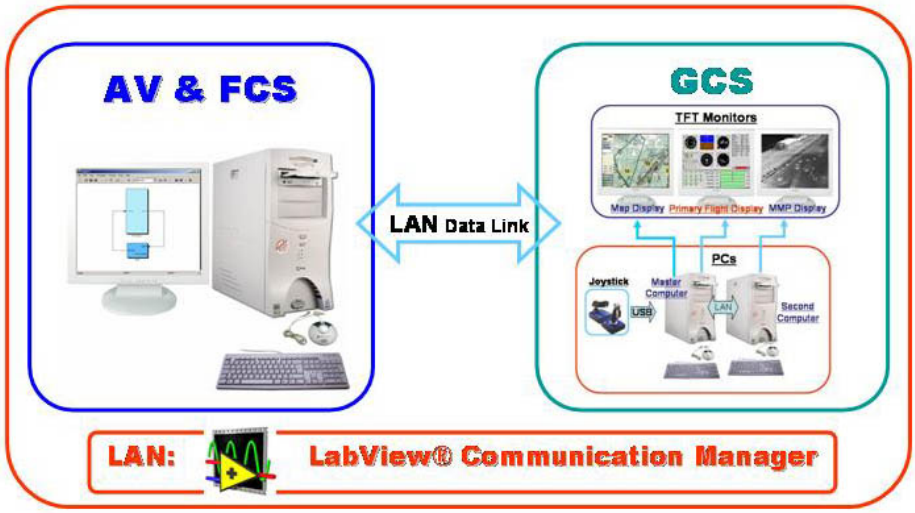


Figure 5: Mission Simulation Environment

The simulation environment is able to emulate the RUAV main sub-systems [10,21]. It is composed by three computers and three monitors (figure 5):

- The "Air Vehicle" computer: it represents the "airborne world" and contains a Simulink™ dynamic model of the helicopter and of the NGCS. In a preliminary design process, this part of the system may allow the designers to test different NGCS solutions.

- The "Ground Support System" computers: for the sake of simplicity the Ground Support System has been simulated as a unique control station, constituted by two computers:

- the master computer is used for real time mission planning, control and for managing communication among the mission simulation environment computers. IT is connected with two monitors displaying the mission planning window on the first screen and the flight instrumentations on the other one
- the second computer is used for providing a virtual view of the mission scenario and for mission payload data display. An Electro Optic camera payload was also simulated by means of a Visual system developed at the Faculty VLAB.

In a preliminary design process, this part of the system may be used to improve the GCS human interface design.

- Data Link: Communication between the AV and the GCS is simulated of course via local area network (LAN). Bidirectional communication between the AV and the GCS primary master computer is done by means of TCP/IP protocol managed by a LabView^{TMTM} software. Communication between the two computers of the GCS is done via UDP protocol and is always managed by the LabView^{TM} software.

The goals to be achieved with the development of the mission simulation environment were many [22]:

1- to create a simulation environment capable of prototyping a full RUAV system, including also the GCS operator into the design process since the preliminary design stage
2- to provide a modular and open environment easy to upgrade by changing a single or a set of components. The "open system approach" may allow quick tests of multiple design solutions, reducing design risk and time with the expectation of life cycle cost reduction
3- to create a tool that improves the RUAV performance estimation and evaluates the RUAV stability and controllability qualities by testing the full system into a realistic operational scenario.

Details of the mission simulation environment software implementation will be given in the next sections, while the most important achieved results will be reported in section 2.3

2.1 THE AV & NGCS SIMULINK MODEL

In order to simulate the behaviour of an autonomous RUAV, it is necessary to model the air vehicle dynamics and the navigation guidance and control system (NGCS).

The complete Simulink^{TM} model developed by UNIBO is shown in figure 6.

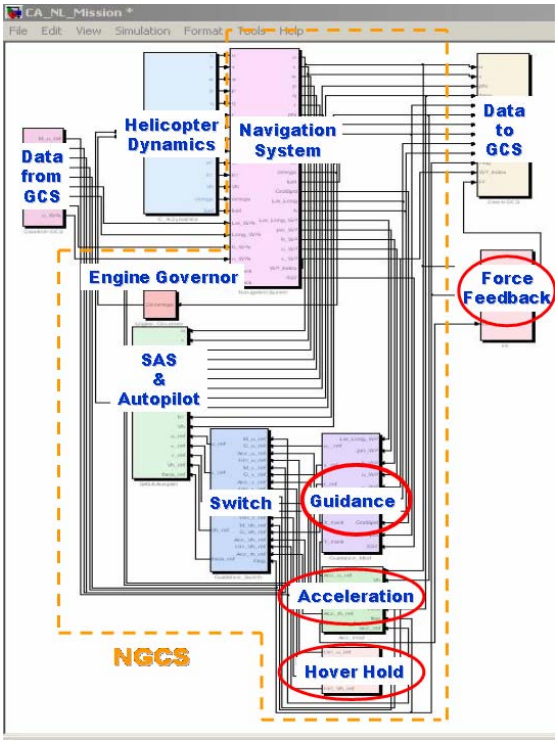


Figure 6: AV & NGCS Simulink™ model

The model is composed by several different parts:

- two "communication" blocks for exchanging data with the GCS computer,
- the "helicopter dynamics" block which is able to simulate the flight dynamics of a classic main & tail rotor helicopter. The model is a non-linear rigid blade dynamic model using main rotor first-order flapping dynamics, steady & uniform Inflow, and combined momentum & blade element theory. It was primarily derived from Mettler [20, 30, 31, 32] theory of small scale helicopters
- the Engine Governor block which changes the throttle settings in order to maintain constant rotor RPM,
- the Navigation, Guidance and Control System blocks which are able to provide controls for the air vehicle stabilization and enable the air vehicle to track a set of pre-planned flight segments, starting from any initial condition,
- the Switch block which is able to change the flight mode depending on a flag input joystick signal coming from the Ground Control Station. Four different flight mode were implemented which are detailed in section 2.1.1
- the Stability Augmentation System (SAS) & Autopilot block works both as stabilization and autopilot system. The autopilot gives controls to the helicopter flight dynamics block

for the Air Vehicle to maintain reference flight parameters, depending on the selected flight mode,

- the Force Feedback (FF) block gives back to the active joystick, at the GCS, different force feedback laws, according to the current flight mode. The active joystick features were used to backdrive the joystick, in order to eliminate transients between the various mission phases and to provide situational awareness to a potential ground control station pilot.

Details of the dynamic model and of the SAS and autopilot can be found in [19] (other useful references are in [33-39]. In the next section details of the navigation guidance system and the four control mode will be given.

2.1.1 CONTROL MODES DESCRIPTION AND FORCE FEEDBACK LAWS

Four control modes were defined for mission accomplishment: autonomous, manual, acceleration and hover hold.

They can be selected by means of the force feedback joystick buttons. Depending on the selected flight mode, the reference parameters for the autopilot are different as well as the force feedback on the joystick.

In autonomous mode, the reference flight parameters are generated by the guidance system (see section 2.3) according to the prescribed flight plan planned at the Ground Control Station, during which joystick stick inputs are ignored. The force feedback module backdrives the joystick so that it follows the current flight condition (see figure 7).

In manual mode, the reference flight parameters coming from the joystick are forward speed, lateral velocity, yaw rate and rate of climb/descent. The force feedback module helps the pilot to maintain the commanded reference speed (see figure 7).

In acceleration mode, the “acceleration” block calculates the reference flight parameters for the helicopter to achieve the maximum acceleration according to the energy law, described in section 2.1.2. The force feedback module gives back to the joystick the reference pitch angle to be maintained, calculated using the energy management equations. The reference flight parameters, commanded by the pilot, are the yaw rate and the rate of climb/descent. Moreover, also the pitch angle can be changed if necessary (see figure 7).

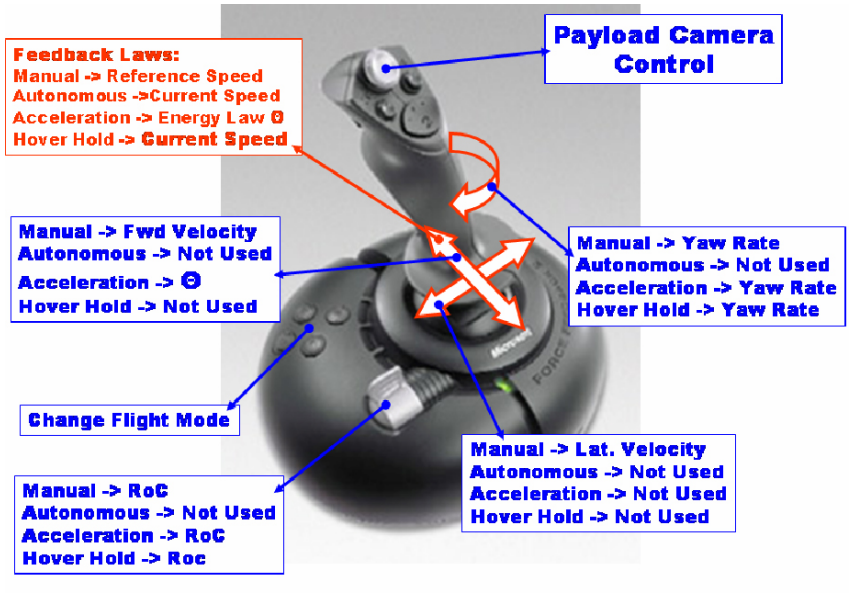


Figure 7: Control Modes and Force Feedback Laws

In hover hold flight mode, the "hover hold" block allows to quickly decelerate the helicopter in order to reach hover flight conditions and to maintain current spatial position. The force feedback module backdrives the joystick stick to follow the current speed so that it works as a speed indicator (helping situational awareness). The reference flight parameters commanded through the joystick are the yaw rate and the rate of climb/descent (see figure7).

2.1.2 ENERGY MANAGEMENT EQUATION

Energy management equations were implemented in order to define a maximum acceleration flight mode towards the object for task accomplishment of a potential search mission. The maximum acceleration was computed using an extension[2] to an approach documented in reference [40].

Reference [40] documents an approach to compute the pitch attitude required to perform a maximum acceleration at constant altitude while not exceeding limits in the vertical axis. For the purpose of this work, the Energy Management approach was used, but it needed a slight extension to take into account also for the rate of descent, which could be experienced by the helicopter during a mission descent phase .

The equation for the pitch attitude leading to maximum acceleration in forward flight and at constant altitude is shown below:

[2] The energy management extension approach was developed in cooperation with DLR (Ing. Stephen Mouritsen). Ref. 42 is the outcome of this work

$$\Delta\theta_{level\_flight} = \frac{1}{V \cdot \frac{\partial \dot{h}}{\partial \delta_{coll}} \cdot (\delta_{c\max} - \delta_{ctrim})} \tag{2.1}$$

For the original derivation, please refer to reference [40]. The terms in the equation are defined in the notation. This equation assumes a helicopter flying in forward flight so that the reciprocal of the forward velocity does not go to infinity. Each term of the equation is readily available: the forward velocity, the maximum collective, the trim collective and the stability derivative (rate of climb vs collective). This equation assumes level flight and computes the nose down pitch attitude such that there is sufficient collective margin to hold altitude (see the figure below).

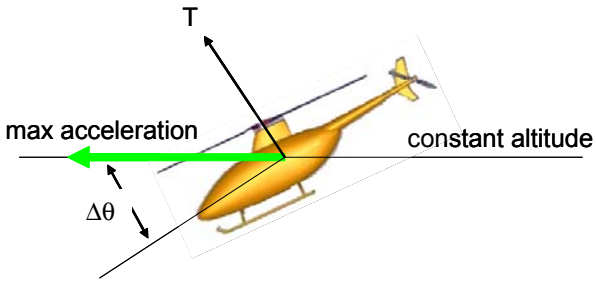


Figure 8: Max Acceleration at Constant Altitude; Δθ in figure is that of equation 2.1 [42]

During certain mission phases, however, the helicopter may need to perform a maximum acceleration, but not at constant altitude, rather in a descent. It follows, then, that the required pitch down attitude will not be as large relative to the flight path, because acceleration is also being aided by a component of gravity (see figure below).

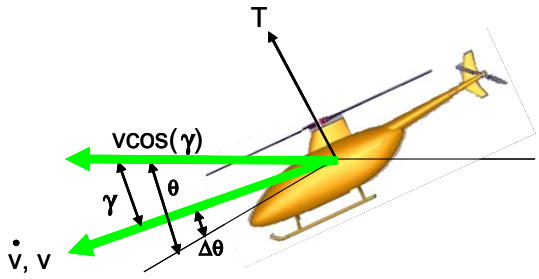


Figure 9: Max Acceleration along a descending flight path; Δθ is that of extended approach (eq. 2.4)

The Energy Management extension begins with the same basis equation used in Reference [40], the relation between potential energy in the vertical direction and kinetic energy in the flight path direction.

$$E=mgh + ½*mV^2 \quad (2.2)$$

To take acceleration in a descent into account the equation is modified by taking only the horizontal component of the flight path velocity.

$$E=mgh + ½*m[V*\cos(\gamma)]^2 \quad (2.3)$$

If we derive $\Delta\theta_{extended\_approach}$ from eq. 2.3 (following the procedure described in [40,41]), the results is an extra term multiplied against equation (2.1):

$$\Delta\theta_{extended\_approach} = \frac{1}{V \cdot \frac{\partial \dot{h}}{\partial \delta_{coll}}(\delta_{cmax} - \delta_{ctrim})} \cdot \underbrace{\frac{1}{1 - \frac{1}{V \cdot \frac{\partial \dot{h}}{\partial \delta_{coll}}(\delta_{cmax} - \delta_{ctrim}) \cdot \sin(\gamma)}}}_{\text{extra term}} \quad (2.4)$$

$\Delta\theta_{extended\_approach}$ is the pitch attitude required to accelerate in a maximal sense along a flight path, regardless if the flight path is horizontal or descending.

Checking the signs in the extra term in equation (2.4):

$1/V \quad > 0$

$$\frac{\partial \dot{h}}{\partial \delta_{coll}} < 0$$

$(\delta_{cmax} - \delta_{ctrim}) > 0$

$\sin(\gamma) > 0$

This means the entire extra term itself is always:

$0 <$ extra term ≤ 1.

This has the effect of reducing the needed $\Delta\theta$ in a descent flight path, which makes sense because acceleration is now aided by gravity and it is not necessary to command the same nose down pitch attitude as in horizontal flight [42].

Equation (2.4) was implemented in the acceleration block (see figure 6) and was used as input for the helicopter controller. The reference attitude provided to the controller was then:

$$\theta_{ref.} = \gamma + \Delta\theta_{extended\_approach} \qquad (2.5)$$

Simulation tests showed the rotorcraft UAV accelerations (along body x-axis) to be quite brisk, between 0.2 and 0.3 [g] at the beginning and reducing to about 0.1 [g] as the acceleration phase ended (see curve slope in figure 10).

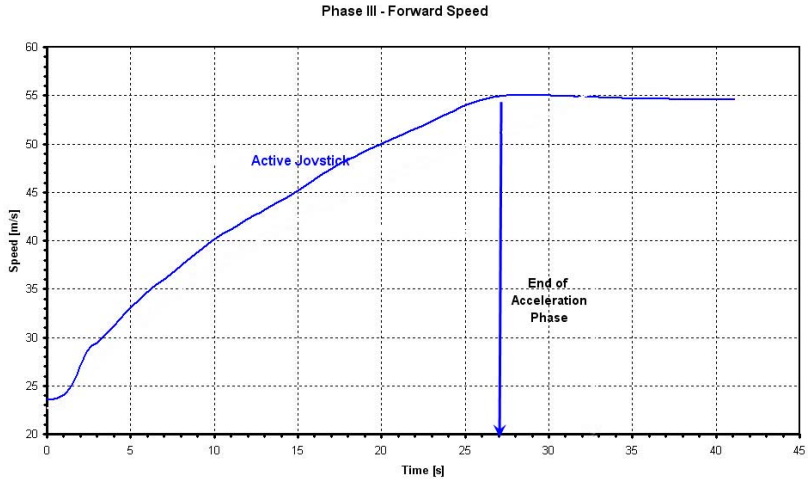


Figure 10: Forward speed vs Time during the max acceleration phase

The acceleration decreases with forward velocity due to the reciprocal term of the velocity in the very beginning of equation (2.4). The reciprocal term eventually drives $\Delta\theta_{extended\_approach}$ to zero with increasing airspeed, at least theoretically.

To aid the pilot, the force feedback joystick is backdriven to a forward position such that the pitch attitude follows the Energy Management law but with an extension which takes into account also for the extra term.

This feedback law is used when the helicopter flies in maximum acceleration mode as it is implemented in the force feedback block of the simulator together with the other feedback laws used for the three other control modes (see previous section).

2.1.3 NAVIGATION GUIDANCE AND CONTROL SYSTEM (NGCS)

The NGCS system used to guide the helicopter implements a proportional navigation strategy. The same strategy was also used for the onboard navigation software implementation of the rotary wing air vehicle developed in this PhD work (see enclosed CD).

The guidance system is composed by two main parts: the lateral track control and the "altitude hold" [43].

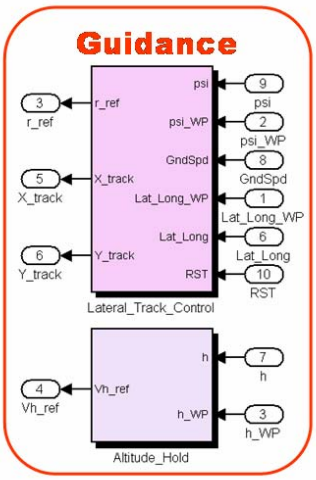


Figure 11: AV Guidance Simulink™ blocks

The altitude-hold is a simple Proportional Integral (PI) controller. It takes as input the destination waypoint altitude and the current vehicle altitude and gives as output the vertical velocity, required to maintain or reach the reference altitude.

The lateral control strategy guides the helicopter towards the destination waypoints (e.g. *WP 2*), along a track line defined by two consecutive waypoints (e.g. *WP1* and *WP2*) as depicted in Figure 12, by means of a yaw-rate command [43]. The guidance block first transforms the helicopter latitude and longitude coordinates into a (Xtrack ,Ytrack) local system (see figure 12). Knowing the AV current position in the local coordinates system (X*track*, Y*track*), the navigation strategy is to turn the ground speed vector ***V*** into the direction of the exact track, so that the helicopter intercept the track line at point C.

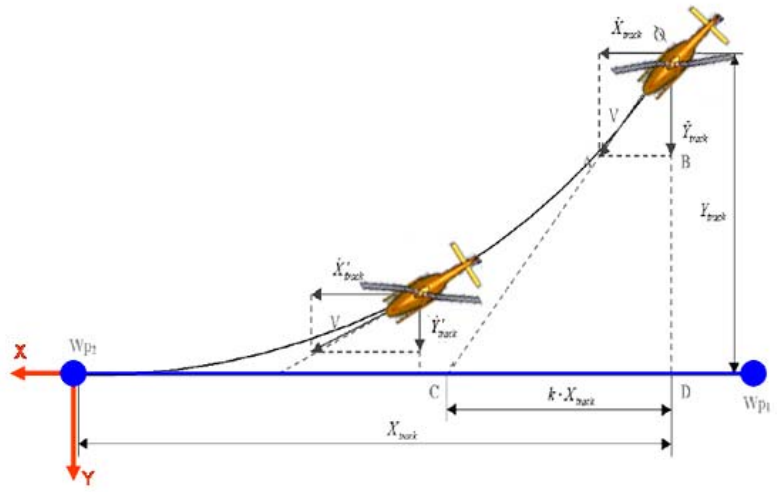


Figure 12: AV lateral track control strategy

The intercepting point C can be tuned by a design parameter *k*. The distance along the track line between the intercepting point C and the *Wp2* is, at any instant, equal to (1-*k*)*X*track*. From the geometry similitude of the triangles OAB and OCD, a control strategy can be defined following relationship [43]:

$$\frac{\dot{X}_{track}}{kX_{track}} = \frac{\dot{Y}_{track}}{Y_{track}} \tag{2.6}$$

To achieve this objective, the error E is computed by:

$$E = kX_{track}\dot{Y}_{track} - Y_{track}\dot{X}_{track} = 0 \tag{2.7}$$

The error can be driven to zero, using the proportional feedback control law that expresses yaw-rate commands as:

$$r_{CMD} = K_R E = K_R\left(kX_{track}\dot{Y}_{track} - Y_{track}\dot{X}_{track}\right) \tag{2.8}$$

The proportional gain K_r is determined iteratively through simulation, until good tracking is achieved with virtually no overshoot. The proposed lateral control handles also wind conditions in a simple manner, ensuring track stability over a wide set of initial conditions (see figure 13).

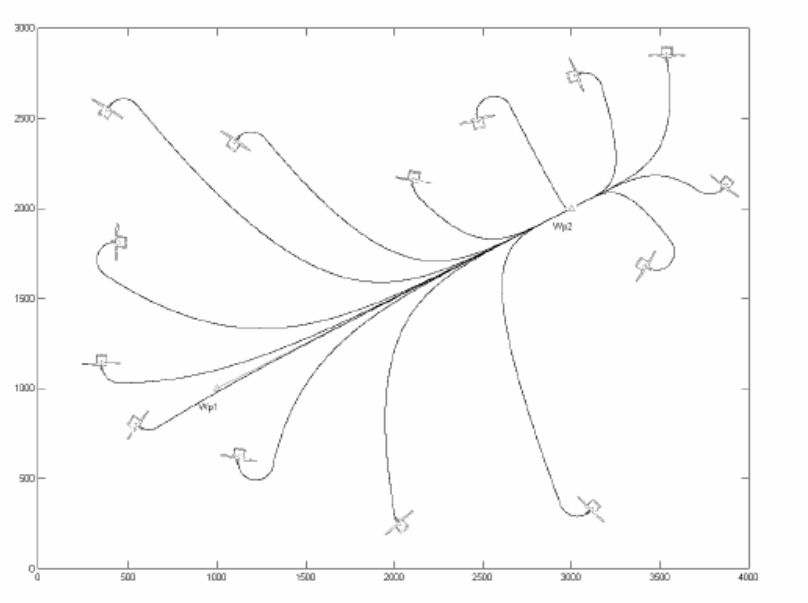


Figure 13: Simulation of lateral track control [43]

In manual flight mode, the guidance system is obviously disabled: the operator acts on the joystick in order to control directly body axis forward speed, vertical velocity, side sleep velocity and yaw rate. In this mode the joystick commands are directly sent to the autopilot.

2.2 GROUND CONTROL STATION

The work involved in this part of the project was the design and development of a ground control station for real time control and display of the simulated RUAV flight test data. The GCS is the hub of an unmanned air vehicle [44,45]. It processes the incoming data and sends control instructions to the air vehicle. Typically a GCS will envelope three main functions:

- mission planning
- mission control
- data processing.

The level of autonomy of the RUAV and the mission complexity dictate the GCS architecture. For the purpose of this work, a simplified standard GCS was developed, which is able to operate the RUAV in both autonomous or remote piloted flight. The GCS was designed to be easily modified for controlling and monitoring of a real RUAV; therefore, this part of the work was used as starting point to develop the GCS of the small scale helicopter UAV system.

The GCS includes a visual system which induces a sense of presence in the engagement area, provides a multi-modal input interface, including head tracker and joystick, which enables efficient interactions.

Key problems to be solved during the development of the GCS were (see next sections in details):

- the interfacing of the different hardware and software components,
- the development of the graphic interface for mission planning and control and flight data display,
- the development of a visual system for modular mission payload simulation and for a data-driven "virtual view" of the flight vehicle, displaying the helicopter current position.

The basic software was developed through the LabView™ data acquisition, control and visualization software. The LabView™ software has been chosen due to its quick and flexible applications. The source code, implemented on the primary master computer of the GCS, is able to manage:

- communication between the Simulink™ model of the air vehicle and the master computer of the GCS,
- communication between the visual system, developed in C++ language, and the primary master computer of the GCS,
- the graphic interface for mission control and flight data display.

Figure 14 shows the LabView code, which runs on the GCS master computer (please refer to enclosed CD for complete software implementation).

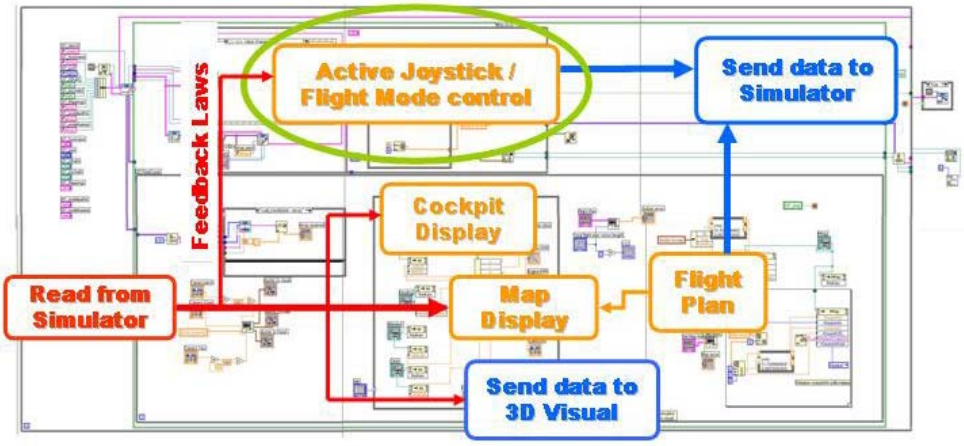


Figure 14: GCS LabView code

The LabView code is constituted by different blocks:

- a “read loop” which receives data from the RUAV simulator via TCP/IP using the LabView simulation interface toolkit 2 blockset;
- a “data selection block” which is able to split the data, received at the GCS primary master computer, into four main cluster of data to be displayed on the GCS graphic interface: a “cluster to visual” data, a “cluster to map” data, a “cluster to virtual cockpit” data and a feedback laws data to be sent to the active joystick;
- two graphic blocks have been created for generating real-time plots of various flight parameters, animated map display, flight plan window and virtual cockpit;
- a “ Force Feedback joystick manager” which manages input/output communication with the USB FF joystick;
- a “send loop” to the visual system for displaying data on a 3D graphical interface which uses an UDP communication protocol;
- a “send loop” to the air vehicle computer for sending real time control signal. In remote piloted flight mode the control signals comes from the joystick interface, while in autonomous flight mode the control-navigation signals depend on the flight plan, defined at the flight plan graphical interface.

2.1.1 COMMUNICATION MANAGER AND SOFTWARE INTERFACING

Communication between the Simulink model and the GCS primary master computer was done using the LabView simulation interface toolkit 2 (SIT V2.0). A vector of data is sent from/to the “air vehicle” computer to/from the GCS, via a TCP/IP communication protocol. Data received at the GCS primary master computer are then distributed to the graphical interface, to the active joystick and to the visual system; meanwhile control command data must be sent from the GCS to the air vehicle computer for the RUAV control and navigation. Since the visual system is developed in C++ and requires an UDP communication protocol, another communication interface between the visual PC and the master PC is needed and was therefore developed. Moreover, another block was developed for communication between LabView and the force feedback joystick.

Communication Simulink™ model-GCS

The LabView SIT V2.0 software was used to interface LabView with the MathWorks, Inc./Simulink environment. The SIT V2.0 provides a way to generate a LabView user interface which can be used to interact with the Simulink model without converting the

Simulink model into a dynamic link library. The component involved in the interaction between LabView and Simulink are shown in figure 15.

LabView exchanges data with MATLAB/Simulink using TCP/IP. For LabView to communicate successfully with Simulink, it is necessary to have MATLAB running on the "air vehicle" computer. When MATLAB is launched on the "air vehicle" PC, a Simulation Interface Toolkit (SIT) Server starts, which enables LabView and MATLAB to communicate with each other. On the master computer, the LabView front panel provides the user interface to the Simulink model. By configuring the SIT Connection Manager dialog box, it is possible to specify the relationship between LabView controls /indicators and the Simulink model input /output. LabView controls are the GCS data sent to the simulator (arranged into a 2D array); LabView indicators are the GCS data received from simulator (arranged into 1D array).

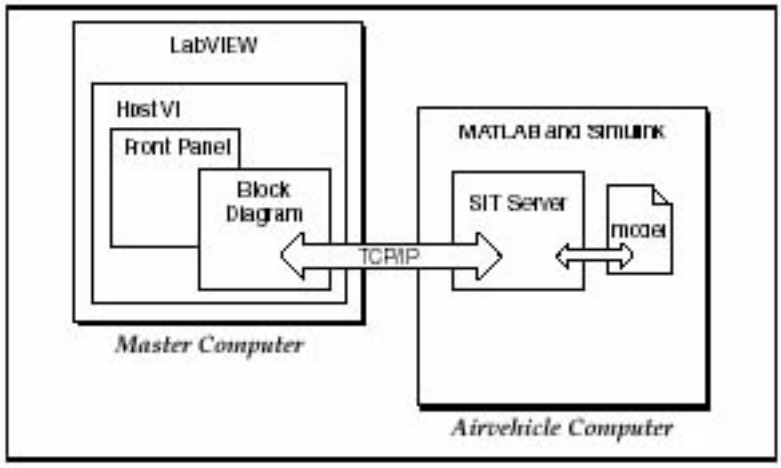


Figure 15: SIT Connection Manager Schematic

Communication visual system-GCS

The visualization software was developed in C++ using OPENGL library and OPENInventor software. It runs on the second computer of the GCS. The software was provided by the Faculty VLAB and was partially modified to be interfaced with the LabView environment. For LabView to communicate successfully with the visual system, an UDP communication protocol has been developed in C++ and integrated in the LabView environment using the LabView call library function node. The call library function node are LabView objects that link compiled source code, written in a conventional programming language as C++, to LabView. When the call library function executes, LabView loads the C++ code and passes input data (in that case the "cluster to visual" data) to the executable code. In that way, data are sent to the visual system which will display a 3D virtual view on a TFT monitor.

Communication force feedback joystick-GCS

The joystick chosen to be installed on the RUAV simulation environment was a Microsoft Sidewinder™ Force Feedback II Joystick as shown in figure 16. This decision was made rather arbitrarily and should not be construed as an endorsement. Other capable force feedback joysticks exist on the market. This joystick did have an advantage though. It is compatible with a joystick driver, written in C++, available from www.Microsoft.com. For those with extra interest, Microsoft offers a Software Development Kit (SDK) for free to support game developers. Visit the Microsoft website and download and install the latest version of the SDK (200 to 300 MB). The kit includes everything needed to write video games, including a joystick interface. Find the file called "joystick.dsw". This is a finished project file compatible with the Microsoft™ Visual C++ compiler. The C++ code is written using DirectX™ libraries which enable the generation of force-feedback effects for devices that have compatible drivers. The code was partially modified to be integrated with LabView so that the force feedback law inputs, coming from the simulator, can be passed to the active joystick. A compiled dynamic link library was generated and implemented in the force feedback manager module of the GCS source code.

This setup allowed testing of new active control ideas inside the mission simulation environment before turning to more sophisticated simulations.

Figure 16: Microsoft Sidewinder™ Force Feedback II Joystick

2.2.2 FORCE FEEDBACK JOYSTICK

The joystick has eight buttons and a 4-position "coolie hat". Button pushes can be recognized within Simulink or National Instruments LabView™, allowing pre-programmed control modes. For example, one button initiated autonomous waypoint flight, while another initiated maximum acceleration flight; the other two are left for the hover hold and manual mode. The 4-position coolie hat controlled the view of a slewable camera during autonomous flight.

For the joystick evaluations, the control range and maximum forces of the joystick can be measured. Based on an optical measurement using a protractor, the joystick can be moved ±35 deg forward/aft and ±35 deg left/right [42].

The maximum forces available from the joystick are ±5 N (not so large).

Nevertheless, the forces are sufficient to be felt by a GCS potential pilot, to backdrive the joystick and to simulate (at least partially) the behavior of a manned helicopter type sidestick.

The joystick forces were programmed in the following fashion, depicted in the picture below:

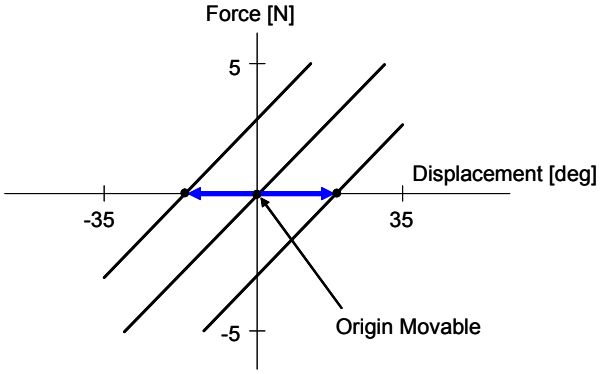


Figure 17: Spring Forces Programmed in Joystick

These forces have the effect of holding the joystick at a desired displacement in both the fore/aft and left/right directions. In other words, the effect is that the joystick can be backdriven. This is a very beneficial effect in that now a backdriven joystick can be used to, in turn, control the helicopter simulation.

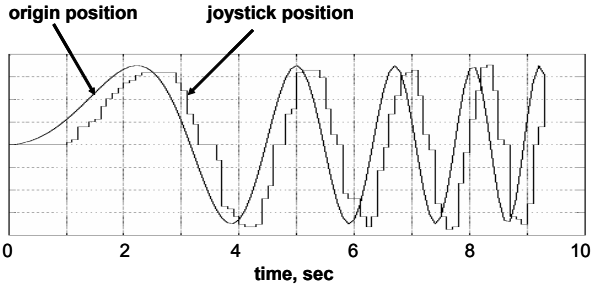


Figure 18: Comparison of Commanded and Actual Stick Position [ ]

The fidelity of the backdriven stick is shown in the above figure. Figure 18 shows a comparison between the position of the spring force origin (commanded position) and the actual position of the joystick, assuming, of course, the pilot does not have his hand on the joystick. It can be seen that the stick follows the commanded position with a small delay and chatter. The chatter is the result of some dead band existing at the spring force origin.

It might be possible (in a later study) to reshape the force curve to minimize the chatter. Evaluation trials were conducted to test if the commanded joystick could, in turn, control a helicopter simulation in real time. Despite the delay and chatter, the stick had no problem "keeping up" with the simulation during stabilized flight and aggressive maneuvers. During these trials it appeared the simulation was controlled by a "ghost pilot".
A stick shaker can be also programmed with variable frequency and amplitude.
This feature can be useful in warning the pilot of impending limits or danger but was not utilized in this study.

A popular feedback cue for manned helicopter sidesticks is softstops, depicted in the following figure.

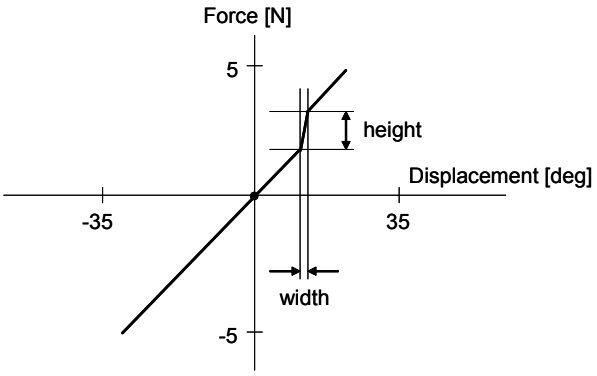


Figure 19: Description of Typical Softstop

A softstop provides a temporary resistance to stick movement at a given location [42]. The resistance is controlled through the height and width variables shown in the figure. It was attempted to simulate such a softstop with the active joystick. The result was found problematic in that joystick dead band make the softstop feel "jerky". A simple fix for this behavior was not found and therefore softstops were not used.

2.2.3 GCS RUAV USER INTERFACE

The GCS user interface is constituted by three video screens (shown in figure 20) which allow the GCS-operator to control flight data information (by means of the virtual cockpit view), to plan or re-plan the mission flight path (using the mission planning window) and to have a good situation awareness during all mission phases (by means of the 2D map view, of the mission vertical profile view, and of the 3D view).
The virtual cockpit and the 2D map view were developed through simple ActiveX controls, such as aircraft instrumentation available from Global Majic Software Inc

(GMS), which can be used as stand-alone applications. The activeX add-ons were chosen in order to simplify the software, since they can be easily interfaced with LabView. Other secondary displays were created using the indicators library of LabView.

The virtual cockpit contains six main displays, arranged according to the basic T layout, as shown in figure 20. They include the air speed indicator, attitude indicator, altimeter, turn coordinator, heading indicator, and vertical speed indicator. As well as the six main displays, other various flight control displays and warnings have been created such as main rotor and engine rpm indicators. For resource planning and monitoring, the Instruments Panel also shows the current fuel level and the mission elapsed time.

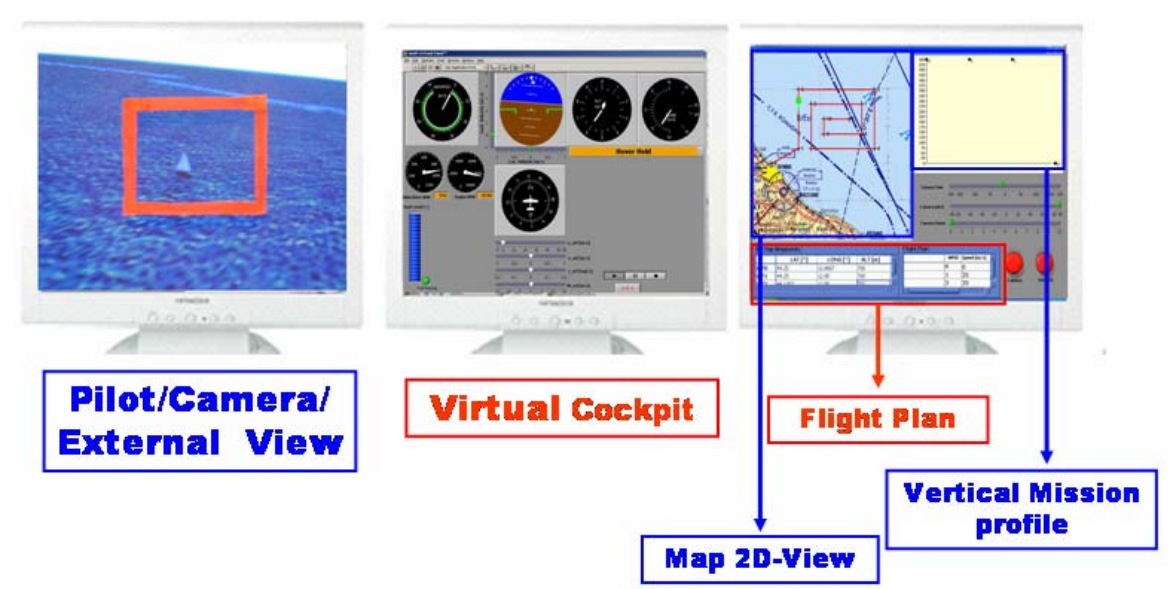


Figure 20: GCS Configuration

The Instruments Panel helps the operator to control the UAV flight.

The mission planning interface (right screen in figure 20) was created using also a LabView software. Mission planning is the first step to do at the beginning of the mission simulation process.

The first step is to decide the targets that represent the mission goals and constraints. After that, mission planning is accomplished mostly through interaction with the "Set Waypoints & Flight Plan" Menu (figure 20). The operator enters the waypoints coordinates (in terms of latitude, longitude and altitude) for the UAV to fly towards the targets. The user interface shows the waypoints and flight paths in 2D graphics. Waypoints can be also added or inserted into an existing flight path for mission re-programming while the simulation is running. Once the waypoints (WP) have been defined, the operator specifies the flight plan parameters into the flight plan menu in term

of waypoint identification number and velocity to be maintained by the helicopter, while overflying a specific WP. When running in autonomous flight mode, the LabView code automatically sends the waypoints coordinates and the flight plan parameters to the helicopter autopilot computer.

Situation Awareness

The visual system is used for viewing the engaging area in 3D and providing the operator with a good situation awareness. The virtual world window enables three different visualization modes, chosen via keyboard switch:

External view (key "1"): This button allows the operator to view the RUAV from outside (fig.21)

Figure 21: GCS UNIBO Visual (external view)

Pilot view (key"3"): This button allows the operator to view the world from the cockpit of the RUAV (fig.22).

Payload view (key "2"): This button allows the operator to display the RUAV EO camera viewpoint (fig.22). The onboard camera can be then moved by means of the POV joystick button. Zoom in and out can be done simply using the computer mouse.

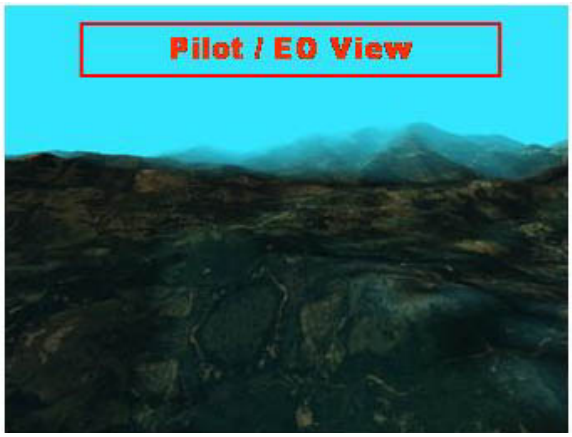


Figure 22: GCS UNIBO Visual (pilot/EO view)

The visualization software was developed in C++ using OpenGL library and OpenInventor at the University VLAB and further modified to be integrated into the simulation environment. It supports also 3D rotorcraft models in VRML format and other elements such as terrain model, clouds, airports and/or buildings in the mission area. The rotorcraft model is located in the virtual scenery based on the air vehicle position data, received through the GCS primary master computer. In order to create a realistic virtual scenery, it is necessary to generate a detailed terrain model of the mission area. Therefore, the visualization software uses a terrain regular grid quadmesh which covers an area of 300 km x 300 km [46] (see figure 23).

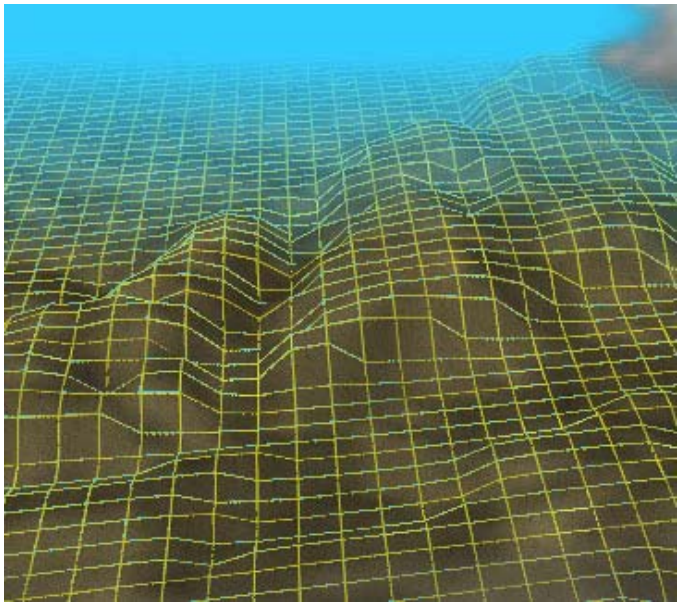


Figure 23: GCS UNIBO Visual terrain mesh

The terrain mesh is constituted by about four millions of polygons and nodes. The nodes elevations are based on free DEM (Digital Elevation Map) data, available at the USGS

(US Geological Survey) catalogue internet homepage [47]. The DEM data are SRTM (Shuttle Radar Topography Mission) data type with 3 arc-sec resolution (that is a pixel of 90m x 90m). The software incorporates also Gaussian data filtering routines to provide effective noise filtering of the SRTM data. Terrain rendering is done using view-dependant polygons rendering algorithms. This kind of algorithm is able to create quad-tree hierarchy structures of the terrain polygonal mesh, such providing quick and efficient terrain rendering even if the terrain mesh is constituted by several millions of polygons. Nevertheless, since the algorithm is "view-dependant", the number of polygons actually rendered every iteration is reduced to about four-ten thousands (4000-10000). One or more textures (such as satellite or aerial pictures) can be also applied to the terrain in order to produce a more realistic virtual environment.

2.3 SIMULATION ENVIRONMENT APPLICATIONS

In this section, the results of two interesting research activities performed using the developed mission simulation environment will be described. The first one concerns the evaluation of the rotary wing UAV configuration developed by AGUSTA inside the CAPECON program; the second is an investigation on how the active features could be used for successful RUAV mission task accomplishment. The last research activities was performed in cooperation with DLR (German Aerospace Centre).

2.3.1 CONFIGURATION EVALUATION

The first application of the developed mission simulation environment was the operational capabilities evaluation of the RUAV, designed by AGUSTA inside the CAPECON program [48,49]. The RUAV platform was tested over a wide spectrum of different mission scenarios, defined in CAPECON.

The results for a standard Fire Surveillance mission (figure 24) are reported in this section as an example of mission simulation and performance estimation. The intent was to evaluate the AV and NGCS performance using a realistic mission profile. Simulation results showed that the NGCS first step design is able to stabilize, control and guide the AGUSTA configuration. After this tests, the GCS layout was also improved to the current layout (described in section 2.2.3), taking also into account ground control station pilot suggestions.

The mission scenario and the related AV performances are described below.

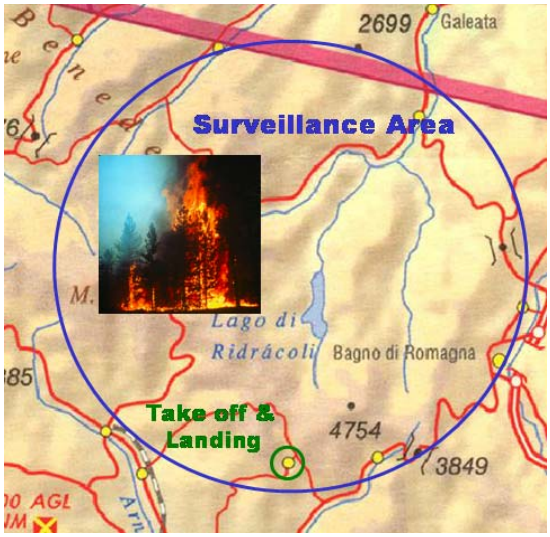


Figure 24: Mission Scenario

The air vehicle was supposed to take-off in manual mode and then follow in autonomous mode the flight path described in Figure 25 and in Table 1.

FLIGHT PLAN				
TAKE-OFF: Manual Mode				
WP	Latitude (North)	Longitude (Est)	Altitude [m]	Speed [m/s]
Wp0	43° 47'	11° 49'	1000	0
Wp1	43° 51'	11° 42'	2200	23
Wp2	43° 57'	11° 40'	1500	23
Wp3	43° 58'	11° 48'	1500	23
Wp4	43° 56'	11° 55'	1500	23
Wp5	43° 52'	11° 58'	2000	23
Wp6	43° 47'	11° 49'	2000	23
LAND: Manual Mode				

Table 1: Flight Plan Data

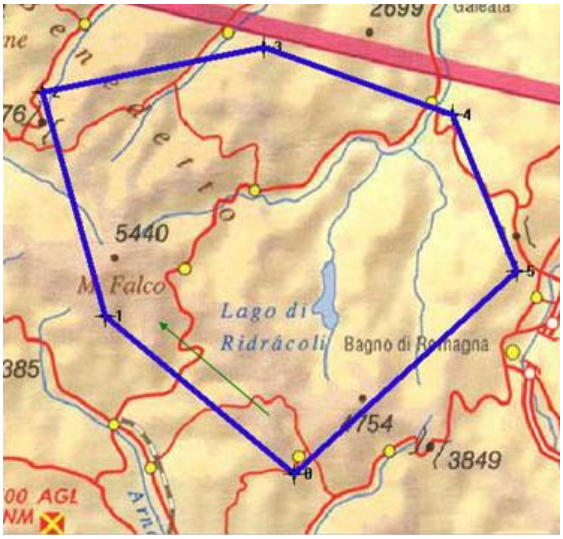


Figure 25: Fire Surveillance Mission flight path

The operator at the ground monitored the surveillance area by means of the simulated onboard slewable camera. For evaluating the mission operational capabilities of the RUAV, it was supposed to find the fire at a certain point of the mission path. In that case, the operator at the ground switched the air vehicle control to manual mode for monitoring the situation. The actual flight path and the mission vertical profile are shown in figures 26 and 27.

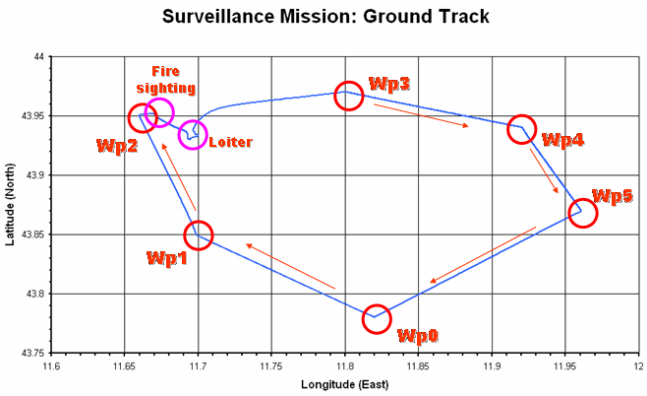


Figure 26: Fire Surveillance Mission actual flight path

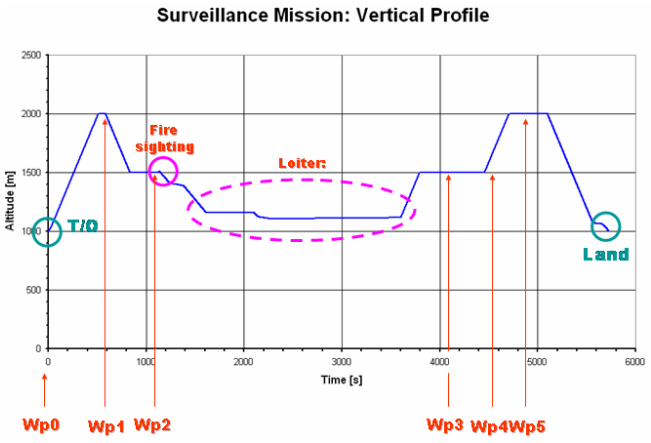


Figure 27: Fire Surveillance Mission vertical profile

Once the air vehicle was nearby the "fire" area, a manual descent and loiter were performed in order to have a better situation overview and send fire position and video images to the ground control station. Particularly, the AV performed a loiter in the area of interest for 40 minutes. After a detailed survey, the "autonomous" key on the joystick allowed the operator to redirect the AV to the original flight path.

If nothing is found the RUAV was supposed to cover the pre-planned path 3 times and then land at the base to refuel. The surveillance path was performed at the best endurance speed while the fire area was reached at higher speed.

The landing maneuver was always done in manual flight mode. Other post-processed data are reported in the following figures.

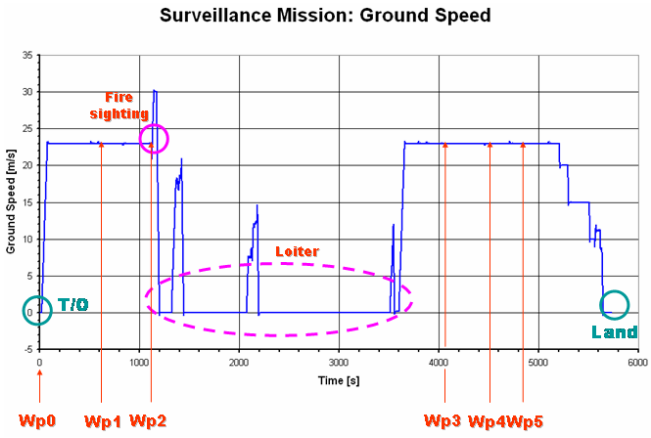


Figure 28: Fire Surveillance Mission ground speed

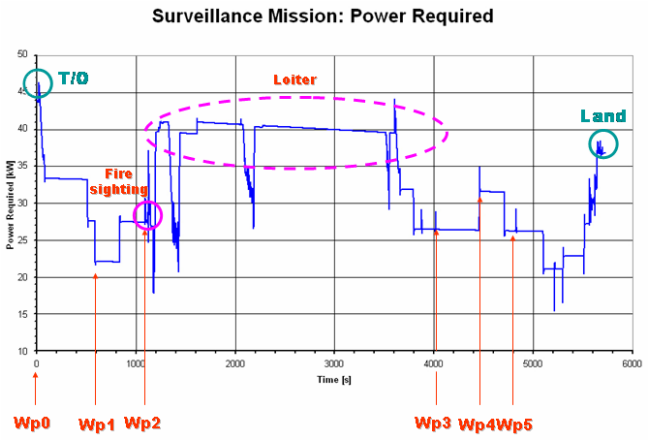


Figure 29: Fire Surveillance Mission power required

Figures 28 and 29 show respectively the ground speed and the power required during each mission phase.

The total fuel consumption was about 33 lts (figure 30). If the AV performs 3 times the flying path with no outrun the fuel consumption was 42 lts.

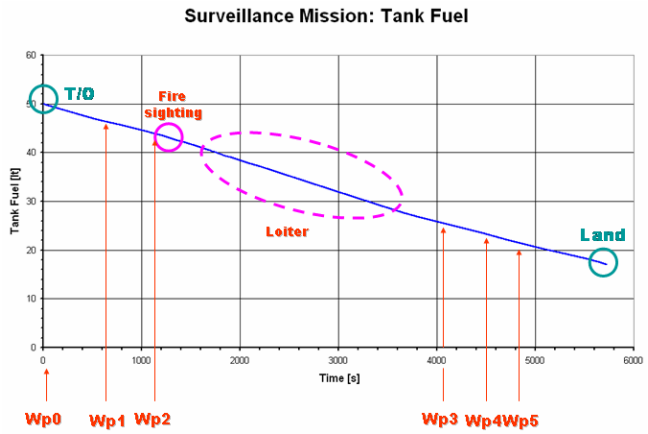


Figure 30: Search Mission fuel consumption

2.3.2 ACTIVE JOYSTICK APPLICATION

The task chosen for this study was a search/identification mission which originates from an autonomous waypoint flight mode. For the task accomplishment, it was supposed that, when the operator at the ground control station saw an object in the camera downlink, he engaged a maximum acceleration mode minimizing the time to the object. The pilot then engaged hover hold mode when he was near the object, all the while keeping the object in view for identification. The active features of the joystick were used to backdrive the joystick to eliminate transients between the various phases and to provide situational

awareness to the pilot during the phases. The pilots reported a marked improvement in mission effectiveness when the active features were turned on. With the active features disabled, some of the task requirements could not be met. Active features on the joystick helped to reduce the workload and total time while, at the same time, helped to increase situational awareness, absolutely needed in any ground control station environment.

To be consistent with the CAPECON program, the flight model chosen for the search/identification task was the UAV rotorcraft designed by Agusta [48]. The design is a 4 bladed, standard helicopter with a single main rotor and a tail rotor. The design is based on studies made to fulfil a number of rotorcraft UAV civil applications [9]. One of which was a search mission requiring 4-5 hours of endurance and a range in excess of 25+ [km]. This range requirement is what makes the use of a GCS compulsory. At 25 [km], the UAV is not in direct view and therefore a means must be found to control it remotely. During the first sizing of this UAV for CAPECON, the maximum take-off weight was 260 [kg]. This assumed only a 40 [kg] payload which reflected the requirement of only minimal components onboard. Therefore the search/identification task could be performed using only a datalink, videolink and a steerable camera, which are well simulated inside the mission simulation environment. No sophisticated object recognition software or fancy sensors were assumed. The idea was to evaluate the operational capabilities of a system capable to be trucked somewhere, set-up and flown with minimal personnel, equipment and operating cost, thus demonstrating also that the task can be performed by an RUAV at a fraction of the cost of a manned aircraft.

2.3.2.1 Search/Identification Task Description

The search/identification task had four phases:

- autonomous waypoint flight
- manual mode
- acceleration mode
- hover hold mode.

The four task phases are shown in Figures 31-34. The most demanding phase was phase 3, the acceleration task. A maximum acceleration was required to save valuable minutes taken from the overall search.

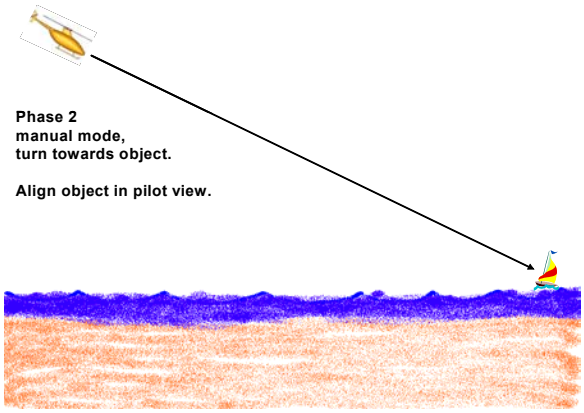


Figure 31: Phase 1, Autonomous, Waypoint Flight [42]

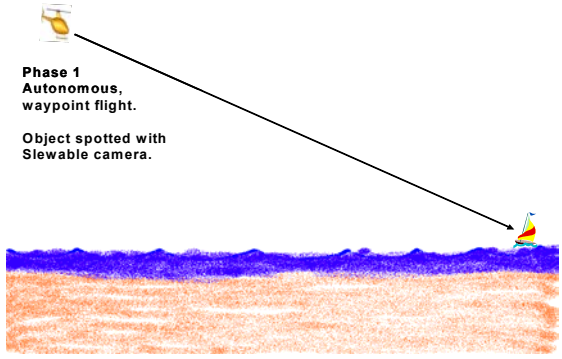


Figure 32: Phase 2, Manual Mode [42]

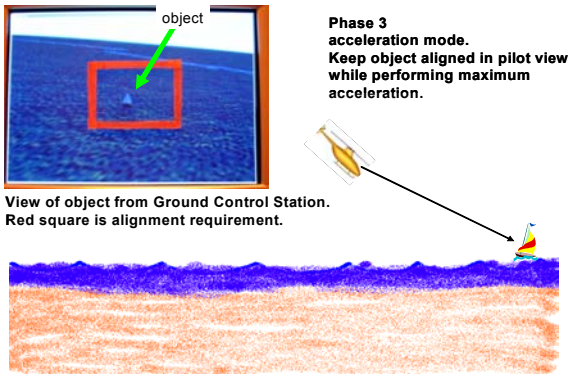


Figure 33: Phase 3, Acceleration Mode [42]

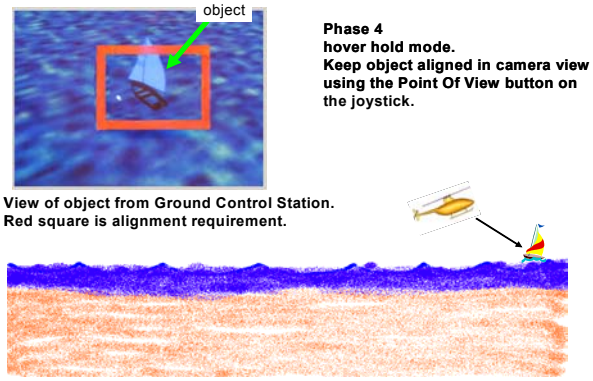


Figure 34: Phase 4, Hover Hold Mode [42]

2.3.2.2 Piloted Simulations

The piloted simulations were organized with the intent of using the Cooper-Harper handling qualities rating scale [50] to measure the impact of the joystick active features. Therefore the task was designed to be flown with or without the active joystick features to offer a direct comparison. It was decided to invite at least three pilots, each with UAV piloting experience, to do the evaluations. The final results were then averaged to show trends. The trends and pilot comments showed the active features made a marked improvement in task performance and situational awareness.

Following the Cooper-Harper rating procedures, so-called "desired performance" and "adequate performance" limits were defined. These limits were necessary to inform the pilots how much aggressiveness was required and which flight limits to be respected. Table 2 shows the limits for all the task modes.

The following definitions were used:

"Camera View" is the view from the onboard camera if it is being slewed around, and "Pilot View" is the view from the onboard camera if it is locked in a forward looking position.

	Phase 1 Autonomous,Waypoint Flight	Phase 2 Manual Mode	Phase 3 Acceleration Mode	Phase 4 Hover Hold Mode
Desired Performance	object found in camera view	- airspeed held $\pm$ 1 [m/s] - altitude held + 5 [m] - object laterally centered	- object centered within red square at all times - total time for Phase 2 + acceleration in Phase 3 $\leq$ 50 [sec]	- object centered within red square (drawn on pilot view) at all times
Adequate Performance	object found in camera view	-airspeed held $\pm$ 3 [m/s] - altitude held $\pm$ 10 [m] - object visible in pilot view at end of phase	- object visible in pilot view at all times - total time for Phase 2 + acceleration in Phase 3 $\leq$ 60 [sec]	- object visible within pilot view at all times

Table 2: Definition of Desired and Adequate Performance

The two most important performance parameters were the centering of the object and the total time for Phase 2 and acceleration in Phase 3. The centering of the object is a classic tracking problem and it is know that precise tracking increases pilot workload. But object tracking is critical for this task as it is an object identification task and the object must be held visible and recorded on camera at all times. Furthermore the tracking is vital so that the UAV does not drift into possible nearby obstacles. In this case tracking is used as a substitute for local navigation. This means the same task can be utilized not only to identify a boat on the ocean, as it is in this simulation, but the task can also be applied in a mountainous region.

The second important performance parameter is the total time for Phase 2 + acceleration in Phase 3. Because our onboard equipment is being held very simple, we assume no radar altitude or laser range sensors. This means that as the pilot is accelerating towards the object, he has a difficult time sensing depth from the 2D video monitor. In other words, the pilot needs a criteria at which to engage hover hold mode. In the absence of such a

criteria, it was decided to define a minimum time to complete Phase 2 and the acceleration to 55 [m/s] in Phase 3.

In practice, it may become necessary to give the pilot better depth information to prevent him from engaging hover hold mode too late, which could result in a collision with the object. Nevertheless, results from these piloted simulations showed that hover hold could be engaged at a safe distance while allowing a detailed view of the object.
Varying the desired and adequate performance for the total time of Phase 2+ acceleration in Phase 3 had the effect of varying the aggressiveness of the maneuver. The final values were selected to be consistent with the maximum accelerations achieved through the use of the Energy Management equations. This means that when the active joystick features were turned on, the desired performance for total time was easily achieved. When the active features were turned off, the pilot had a more difficult time optimizing his flight path and the total time increased. The following figure 35 illustrates this result.

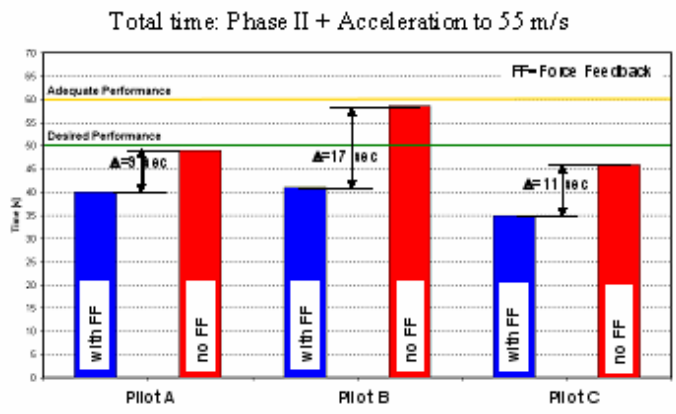


Figure 35: Total Time Comparisons for Pilots A,B and C.

Specifically, total time equals the time from the moment the pilot switches on manual mode (onset Phase 2) to the moment he reaches 55 m/s in Phase 3

It can be seen in figure 35 that there was a consistent improvement in the total time for all three pilots when the force feedback features are turned on. To understand this better, during the acceleration phase, the joystick is automatically backdriven to the forward position corresponding to Energy Management equations. The Energy Management equations compute the required $\Delta\theta$ for maximum acceleration. The control system, in attitude control mode, backdrives the joystick input until this attitude is matched. This relieves the pilot of the task of trying to optimize the acceleration while keeping the object

centered in the pilot view. But with no active joystick cues, the pilot needs more time to accelerate to the desired airspeed.

This result can be seen shown in the airspeed time history of Pilot B in the following figure:

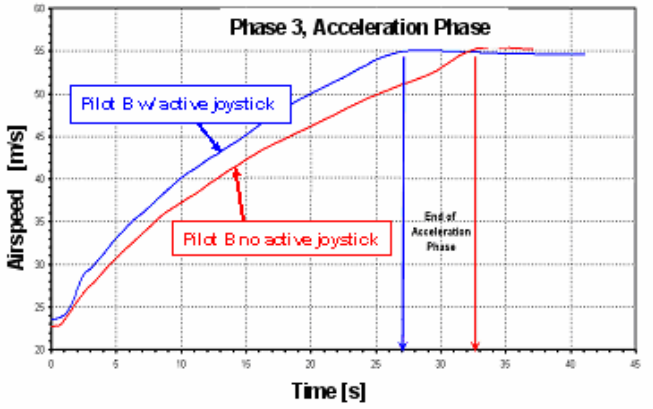


Figure 36: Airspeed Time History, Pilot B, Phase 3

It can be seen in figure 36 that Pilot B could accelerate more quickly when the joystick was automatically backdriven to its optimal forward position.

Other results for Pilot B include a time history of airspeed during manual mode in Phase 2. Here the pilot was tasked with taking over control from autonomous mode and turning the UAV towards the object to be identified. This task could not be automated because it inherently is dependent on the object cues in the pilot view and no object recognition software is assumed onboard.

As aid to the pilot, the joystick commands a speed hold mode so that moving the joystick forward commands a faster airspeed. The active features held the joystick at this position. Of course, with the active features turned off, the pilot had more difficulty hold the required airspeed and the pilot required more time to turn the UAV toward the object as shown below in figure 37.

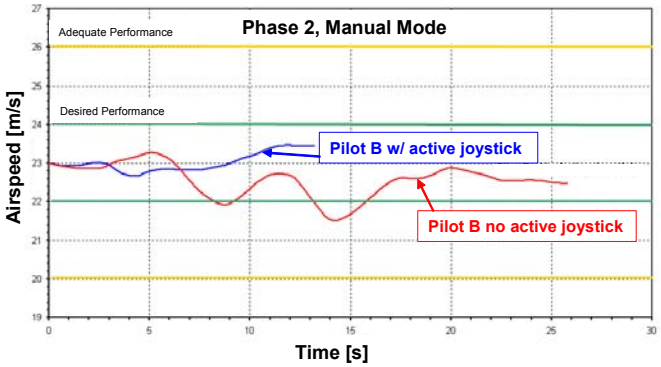


Figure 37: Airspeed time History, Pilot B, Phase 2

It can be seen in the figure that Pilot B could perform the manual mode task easier with a backdriven joystick with an airspeed hold command turned on. It should also be noted that during Phase 1, Autonomous mode, the joystick is continuously backdriven to match the airspeed hold command. Therefore when manual mode is turned on, there are no transient inputs.

The last phase for the pilot is Phase 4, hover hold. When the pilot determines that the object is "close enough" to the UAV, he pushes the joystick button for hover hold mode. This is an automatic hover stabilization which automatically backdrives the joystick to zero out velocities in all 3 axes. The heading is maintained to be the last active heading, that is with the nose pointed towards the object. During this phase the pilot does not enter any control inputs except that he has to operate the Point Of View switch in the joystick to keep the object centered within the red square on the monitor. During the switch from Phase 3 (acceleration) to Phase 4 (hover hold), the onboard camera switches from pilot view (camera fixed forward) to camera view (camera is slewable). The pilot does not have enough cues to do the hover hold himself, that's why it was automated. But during the stabilization, the pilot still needs to maintain visual contact with the object and this extra workload is reflected later in the Cooper-Harper ratings. When the aircraft is stable, then the pilot officially ends Phase 4 and completes the search/identification task.

2.3.2.3 Cooper-Harper Rating Evaluations

The procedures in support of the Cooper-Harper ratings involve a number of issues which, now, will be discussed.

- Step one, of course, is that the engineers need to design a task which is repeatable by different pilots and has performance criteria which can be measured. This was done and the results are documented in the previous sections.

- Next, pilots need to be chosen who fulfill minimum qualifications. Not just anyone should be invited to evaluate the task. The minimum requirements for the pilots were 1) they have an aviation background 2) they have time in either manned aircraft or UAVs and 3) they have a serious attitude when evaluating the task.

- When the task, the Ground Control Station, the active joystick and the pilots are ready, then each pilot takes time to train on the task until he is familiar with it. Each is allowed to fly the task several times until an "official run" is made. Official runs were made for both cases with the active features of the joystick turned on and with the features turned off. The official run time histories are recorded on computer and then the pilot fills out *immediately* a questionnaire and makes a rating. He is obliged also not to tell the other pilots his opinion or rating until all have finished their evaluations. This insures that the ratings are objective.

These procedures were followed with three pilots, A, B and C. Each was asked to fill out a questionnaire probing their knowledge and opinions about the task. The pilot's typically remarked that the joystick cues helped them achieve desired task performance.

Finally the Pilot's were asked to carefully go through the Cooper-Harper decision tree and make a rating. The rating procedure is shown below:

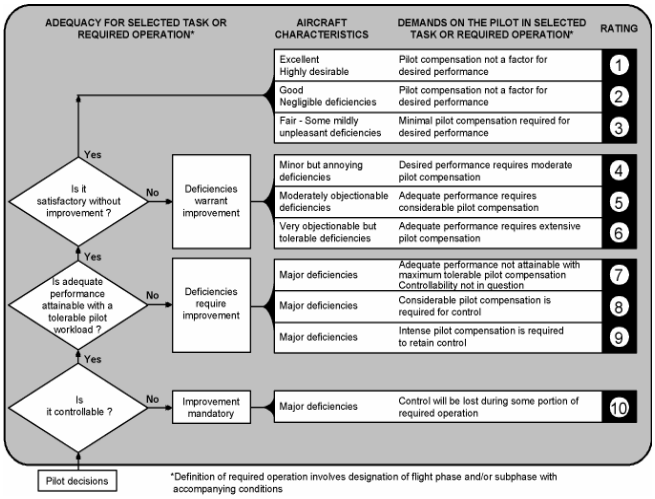


Figure 38: Cooper-Harper Decision Tree

The decision tree in figure 38 is well known and every attempt was made in these tests to respect its assumptions.

Cooper-Harper Ratings	Joystick Active	Joystick Not Active
Pilot A	3	5
Pilot B	4	6
Pilot C	3	6
average	3.3	5.7

Table 3: Cooper-Harper Ratings for the Rotorcraft UAV Search/Identification Task

Table 3 contains the final results for the handling qualities evaluation for the search/identification task. The results show quite clearly the impact active joystick features can have on handling qualities. With the features turned on, control of the rotorcraft UAV for this task is Level 1, normally accepted as quite good. But with the active features turned off, control of the UAV drops into the lower part of Level 2 which normally means the system requires improvement, especially true if it is to be certified under civilian airworthiness regulations.

Chapter 3

ROTARY WING UAV SYSTEM DEVELOPMENT

The goal of UNIBO RUAV project is to develop a helicopter platform capable of autonomous flight which could be used inside the University for researches in control and navigation laws, man-machine interfaces and system integration; meanwhile it should be proposed as a technological prototype for industries interested in UAV development and manufacturing. In order to develop such kind of platform, avionic systems are required that enable the helicopter to maintain a stable attitude and follow desired trajectories. This avionics package is comprised of sensors, computer and data link hardware as well as software to guide, navigate and control the air vehicle. These aspects are particularly critical for helicopters, which are well known to be inherently unstable systems, and place numerous requirements on the avionic system design.

The overall RUAV system architecture developed at UNIBO is show in figure 39.

It has the typical UAV system architecture, as defined in the CAPECON program [49], but simplified for a small RUAV. The modified Hirobo 60 helicopter mechanics was used as flying platform. The RUAV avionics is constituted by an onboard computer (the CRIO system from National Instruments) which acquires sensor data from an Attitude Heading and Reference System (AHRS) and the sonar altimeter and sends PWM commands to the helicopter servo actuator, based on the control and navigation laws implement on it. The data link between the onboard computer and the ground control station is performed by means of a simple WIFI access point. Details of the RUAV hardware set up will be given in chapter 4.

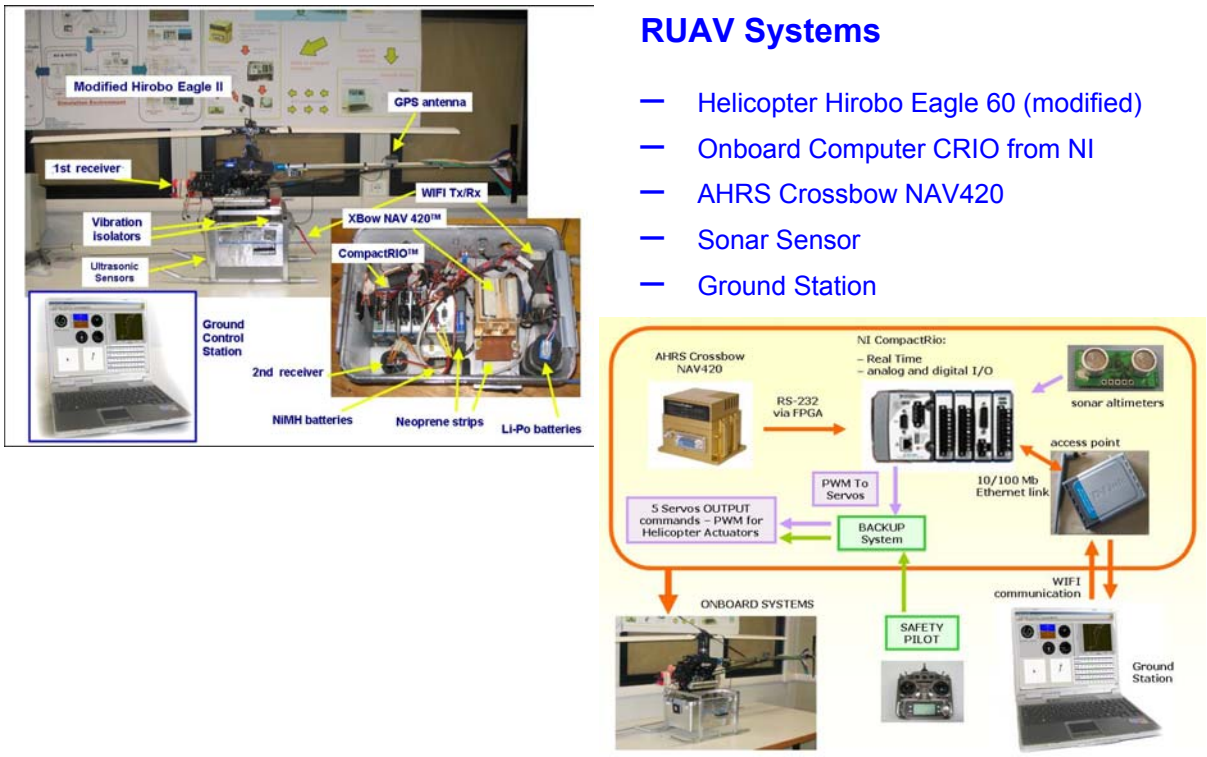


Figure 39: RUAV System set-up and Architecture

Next section introduces the applied design and integration methodology used to set-up the RUAV system. Details of the work performed in this thesis for the development of the RUAV system will be given in the next chapters following the design methodology described below.

3.1 DESIGN PROCESS

The design methodology followed for the RUAV system development is depicted in figure 40.

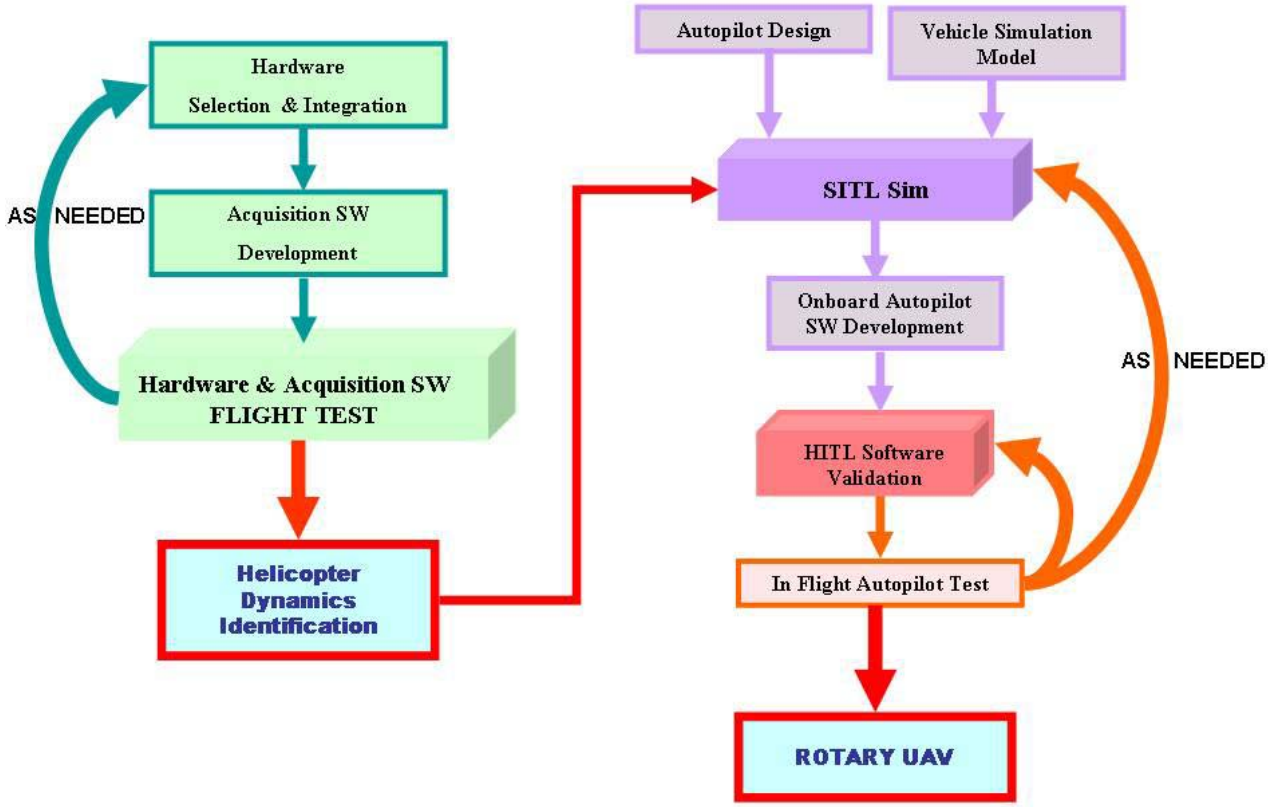


Figure 40: RUAV Avionics Design Flow

It is a multidisciplinary design process which includes six main steps:

1- Hardware Selection and Integration: this task include the selection and set-up of the rotorcraft airframe and of the onboard avionics. Once the hardware is selected, it must be packaged and interfaced placing attention to vibration isolation, electromagnetic interference and accessibility (see chapter 4)

2- Acquisition Software Development: if a RUAV has to fly autonomously, information about its states is needed which must be used by the control and navigation system. Therefore, following the hardware set-up, sensor data acquisition software must be developed and tested in flight in order to validate the acquisition software and ensure measurement reliability (see chapter 4).

3- Software In The Loop (SITL): parallel to the hardware set-up, simulation plays an important role in the development of an autonomous helicopter. At this aim, a series of flight tests must be also done in order to collect experimental data for identifying the helicopter dynamics characteristics and develop a reliable vehicle simulation model. After

that, before actual autonomous flight test can take place, control and navigation algorithm must be design using the identified model of the helicopter dynamics (see chapter 5).

4- Hardware In The Loop (HIL): once the previous tasks are completed, the onboard autopilot software must be developed. After that, the onboard hardware and software must be integrated into the simulation loop. For that, a Hardware In the Loop (HIL) simulator was developed in the NI LabView environment. In this scenario, performance and possible errors of the onboard software can be detected during intensive ground safe and risk free tests (see chapter 5-6).

5- In Flight Autopilot Test: autopilot flight test must be performed for final verification and tuning of the control and navigation system (see chapter 7).

At this point, improvement within the previous steps can and should be undertaken until a configuration is reached that promises satisfactory results for the final RUAV system set-up.

Chapter 4

HARDWARE SELECTION AND INTEGRATION

The synthesis of the RUAV hardware is a trade-off evaluation like any other design process. An optimal design solution is sought, by finding the best compromise to satisfy the design requirements.
The main requirements driving the hardware selection and integration process are outlined in section 4.1 while sections 4.2 to 4.6 describe the selected hardware and the overall hardware system set-up.

4.1 HARDWARE DESIGN REQUIREMENTS

The main requirements taken into account for the RUAV system design were both operational requirements and physical constraints. Numerous requirements were placed on the avionics system design while the air vehicle configuration was somehow freezed to the one already available at he UNIBO laboratories, which was modified only to increase performance and payload carrying capabilities.
The most important design criteria, followed for the RUAV testbed development, was maximum flexibility (i.e. easy and quick reconfiguration) while maintaining good air vehicle performance. Therefore, efforts were concentrated to ease hardware configuration and reconfiguration and allow for future system growth. From this point of view important requirements are :

- to provide accurate flight data acquisition for dynamic model development and validation
- to be versatile enough to enable fast and easy integration of different input/output sensors and to allow future system growth in term of payload, sensors and interfaces
- to be as light as possible in order to lower the total platform weight and maintain good helicopter maneuver capabilities. Preliminary flight test demonstrated that the helicopter still had good maneuverability with 6 kg payload mass.
- to be able to withstand the high vibration load typical of small scale helicopters. The primary sources of vibrations are the engine, the main rotor (spinning at roughly 22 Hz), the tail rotor and the tailboom bending resonance. These vibrations must be reduced to fit the operational vibration range of the onboard sensors and to provide accurate flight data measurements. Experimental tests performed with commercially manufactured elastomeric dampers showed that vibrations can be effectively reduced to the desired level
- to be protected against the electromagnetic and RF interference: common shielding precautions were used to isolate the onboard electronics from EM interference
- to allow onboard implementation of feedback control laws and demonstrate good control capability
- to be endowed with an onboard safety system in event of computer failure. Depending on the size and cost of the air vehicle this can include a completely redundant avionics system or simply a minimum safety system
- cost is of course a limiting factor for avionics and airframe selection and for achievable performance

4.2 FLIGHT TEST VEHICLE DESCRIPTION

The air vehicle chosen as RUAV platform is shown in figure 41. It is a Hirobo Eagle II 60 hobby helicopter which was modified to accommodate the avionics hardware. A more powerful engine, longer fiberglass blades, longer tail boom and tail blades were mounted in order to increase the helicopter payload carrying capabilities. The assembly also includes a Bell-Hiller stabilizer bar, which augments servo torque with aerodynamic moment to change the blades cyclic pitch and adds lagged rate feedback to improve the helicopter handling qualities.

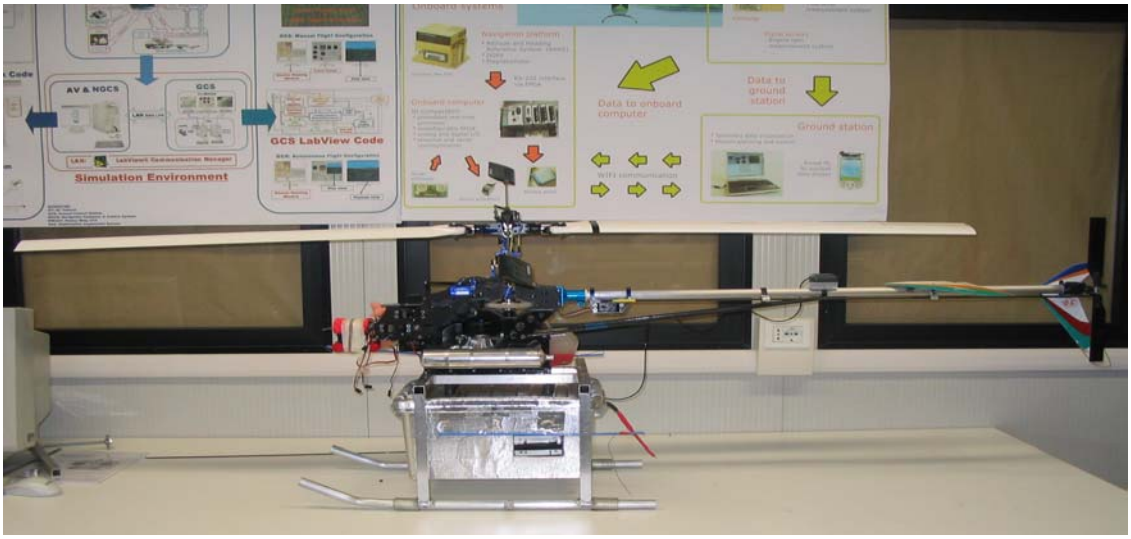


Figure 41: RUAV Air Vehicle

The main helicopter characteristics are:

- main rotor diameter: 1840 mm
- tail rotor diameter: 330 mm
- total helicopter mass: 11.2 kg
- engine: OS 91 Engine 15 cc; power 2.9 CV
- main rotor rpm: 1200-1300
- tail rotor rpm: 5000 -6000
- payload carrying capabilities: 5-6 kg

4.3 FLIGHT COMPUTER

The CRIO system from NI was selected as flight computer due to its ability to fulfill many among the stated design requirements. Particularly, the most important CRIO features that encouraged its usage as onboard computer for the UNIBO RUAV system were:

- modular and versatile architecture
- easily reconfigurable with minimal time investment
- ultrahigh performance and low power consumption
- relatively low cost system
- ease and open access to low level hardware resources
- rapid embedded control and acquisition system development that rival the performance and optimization of custom-designed circuitry
- possibility to use LabView graphical programming tool to develop a variety of different applications

- relatively small size and weight compared to its control and data acquisition capabilities

The CRIO platform includes the CRIO-9004 real time controller endowed with an industrial Penthium 200 MHz floating –point processor, a four slot reconfigurable chassis featuring three million gate FPGAs chipset and a wide variety of analog\digital I\O module types.
Figure 42 shows the CRIO configuration currently mounted on the UNIBO RUAV system.

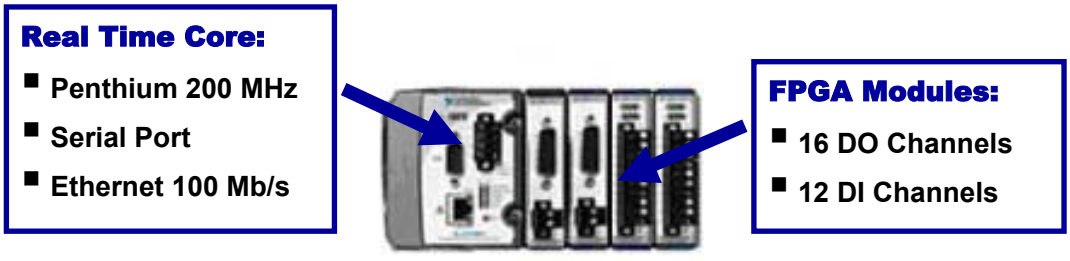


Figure 42: National Instruments CRIO Onboard Computer

The real time controller also features a 100 Mb/s Ethernet port for network communication with an host computer and a 9 PIN serial port.
The FPGA module currently used are:

- CRIO 9411 mounted in slot 1 having 6 digital input channels
- another CRIO 9411 mounted in slot 2 having 6 digital input channels
- CRIO 9474 mounted in slot 3 having 8 digital output channels
- another CRIO 9474 mounted in slot 4 having 8 digital output channels

Each CRIO module contains already build in signal conditioning.
FPGA devices are very useful and powerful since they combine the versatility of a reconfigurable digital architecture with a matrix of configurable-logic blocks surrounded by a periphery of I/O channels. This way, signal can be routed within the FPGA matrix in any arbitrary manner by programmable interconnected switches and wire routes (figure 43).

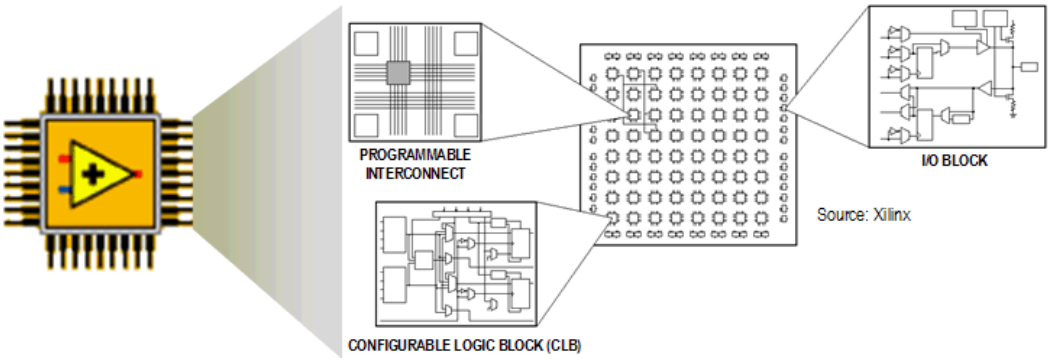


Figure 43: CRIO Field Programmable Gate Array (FPGA) Structure [51]

Control loops can be also implemented inside the FPGA environment using "while loops" up to 40 MHz (25 ns). Moreover, FPGA modules are ease programmable with NI LabView without need to know specialized hardware design languages such as VHDL (the LabView code is directly compiled in VHDL before being downloaded on the FPGA devices).

4.3.1 CRIO REAL TIME APPLICATION DESIGN

The real time control and acquisition system which is possible to develop with the CRIO system typically contains four main components(see figure 44):

- RIO FPGA core application for input, output, inter-thread communication and control
- Time critical loop for floating point control, signal processing, analysis and point-by-point decision making
- Normal priority loop for embedded data logging, remote panel interfaces and Ethernet/serial communication
- Networked host PC for remote graphical user interface, historical data logging and postprocessing

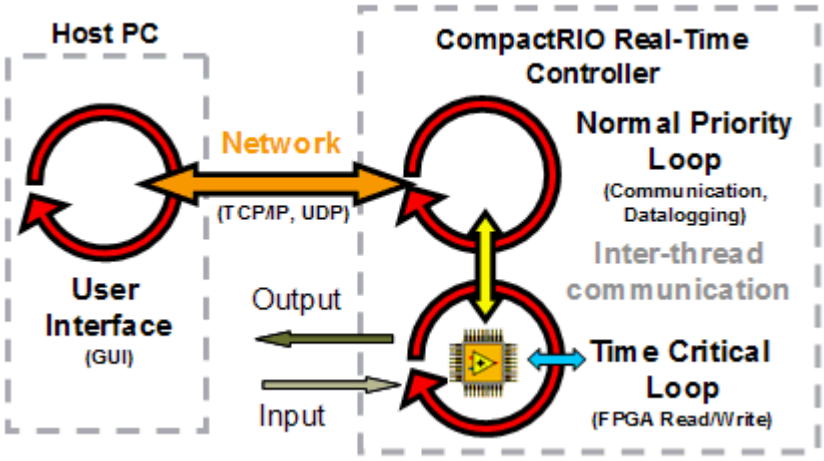


Figure 44: CRIO Programming Structure [51]

Depending on the application requirements, it's possible to implement one or all of these application components. The onboard software, currently implemented on the flight computer, follows this standard approach.

4.4 SENSORS

If an UAV is to fly autonomously or needs stability augmentation in remote controlled flight, its flight control algorithms need information about its state, which can be obtained by means of onboard sensors. Depending on the vehicle type and its mission, sensors can be different. For the purpose of this work, sensor types have been split into Attitude Heading and Reference System (AHRS) and altitude sensors.

4.4.1 ATTITUDE HEADING AND REFERENCE SYSTEM (AHRS)

Most common attitude sensors are based on gyros that can be either mechanical, piezoelectric or optical. A three axis gyro platform measures angular rates along all axes of the vehicle and is usually contained in an Inertial Measurement Unit (IMU), which also provides data from accelerometers. Magnetometers are also used to determine heading of the air vehicle by measuring the Earth magnetic field. Attitude and position can be then calculated in a state estimator by integrating IMU measurements. However the high accuracy, simplicity and availability of the Global Positioning System (GPS) makes it the emerging standard positioning system for UAVs as well as for general and commercial aviation. Depending on the quality of the GPS receiver, the achievable accuracy and the GPS update rate varies. Since common GPS update rate is usually once a second, this can

result in a limited bandwidth of the UAV controller. A common way of solving that problem is to fuse data from all the flight sensors into a navigation filter in a state estimator. In addition altitude data (coming from a radar or sonar altimeter) and magnetometers measurements can be also used to improve the navigation filter. Usually an extended Kalman filter approach is used to integrate data from all the navigation sensors [52].

An alternative solution to IMU, individual gyros and self-built navigation filter is to use a complete AHRS like the CrossBow NAV 420, which was chosen as navigation platform for the purpose of this work. This kind of unit is able to directly deliver vehicle attitude, GPS velocity and position data, acceleration and rates at a rate up to 100 Hz, thanks to a high performance Kalman filter algorithm implemented on an internal digital signal processing module. Velocity data includes aiding from the inertial instruments such reducing the latency associated with stand-alone GPS measurements.

Particularly, the NAV 420 uses the latest in solid-states sensor technology and consists of the following subsystems (see figure 45):

1) Inertial Sensor Array: This is an assembly of three accelerometers, three gyros (rate sensors) and four temperature sensors.

2) A three axis fluxgate magnetometer board used to compute heading.

3) A WAAS capable GPS receiver for position and velocity measurement.

4) A digital signal processing (DSP) module, which receives the signals from the inertial sensors and magnetometers. This unit converts the signals to digital data, filters the data, computes the attitude solution, monitors and processes all BIT data, and transmits the results to the user.

The NAV420 analog sensor signals are sampled and converted to digital data at 1 kHz. The sensor data is filtered and down-sampled by a DSP.

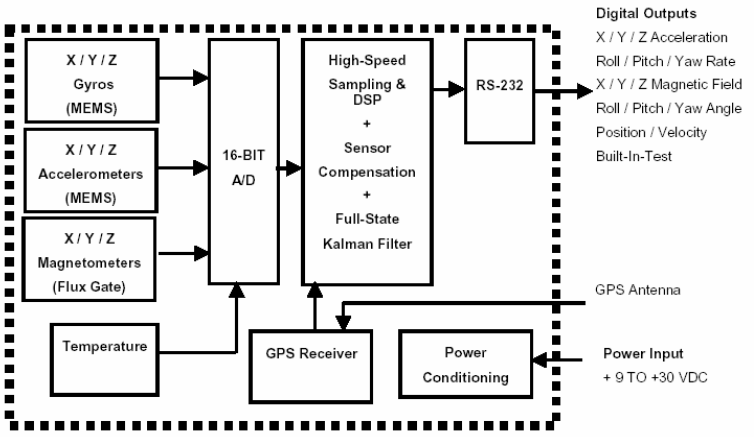


Figure 45: NAV420CA System Architecture

The choice of this kind of platform significantly reduced development time in signal processing and sensor fusion, greatly improved measurement reliability and guaranteed sensor stability and performance in a high vibration operating environment, like the one of a small rotary wing platform.

4.4.1.1 AHRS Set-Up

The NAV420 set-up procedures was done following four major steps (see figure 46):

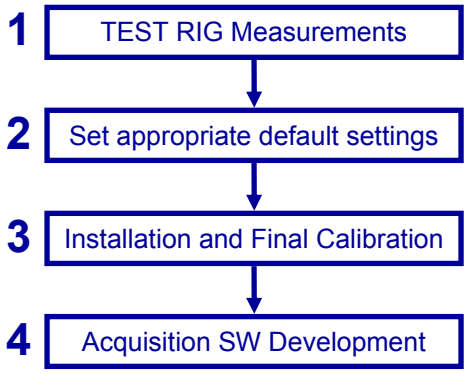


Figure 46: NAV 420 Set-up Procedures

- first some measurement tests were performed on a certified test rig in order to verify the navigation platform responses
- a LabView software was then developed in order to change the NAV 420 default settings: before using the NAV 420 data inside a control algorithm, the update rate, the baud rate and the output packet type must be set to appropriate values
- afterwards the navigation platform must be installed inside the avionics box: appropriate procedures must be followed in order to obtain correct states measurements
- in the end, NAV 420 data acquisition software must be developed in order to read sensor information, to be used inside the onboard control software

Test Rig Experiments

Figure 47 shows the NAV420 mounting on the UNIBO test rig during the first experiments performed to verify the platform responses.

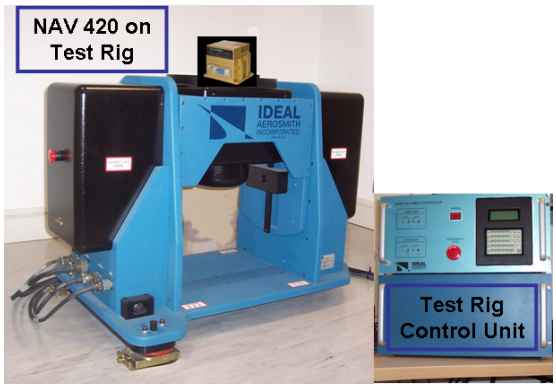


Figure 47: NAV420CA on Test Rig

Reference points in terms of angular rates and attitude were given by means of the test rig control unit; NAV420 responses were recorded by means of the NAVView software provided by CrossBow [53].

Diagrams of some experimental results reported in figure 48 confirmed the good quality of the NAV420 measurement capabilities.

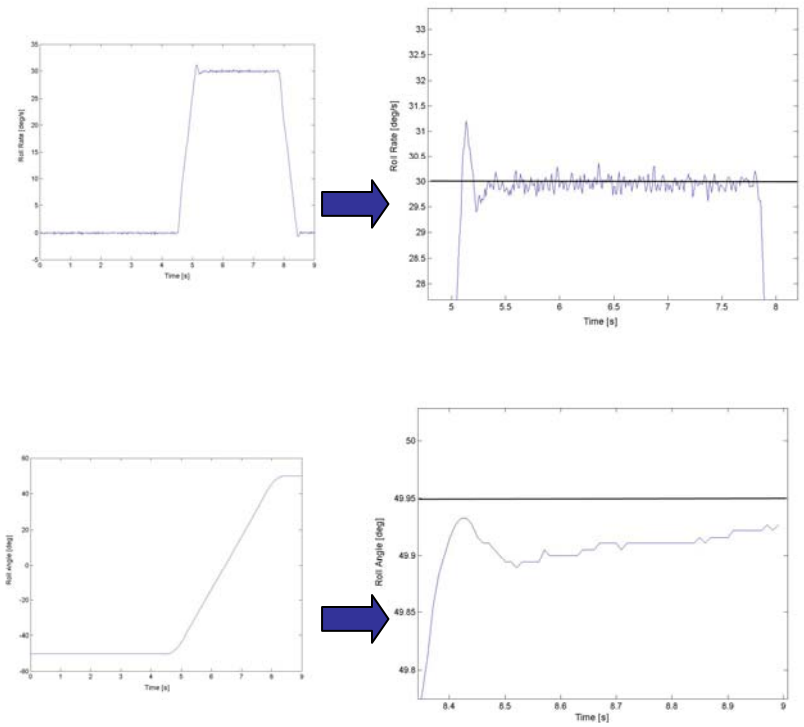


Figure 48: NAV420 Test Rig Measurements

NAV 420 Settings

The LabView software, designed to change the NAV420 default settings is reported in the enclosed CD.

The CrossBow NAV420 provides information to the user by means of a RS 232 protocol. Therefore, the developed software was used to set the transmission baud rate, the packet output rate and the output packet type.

This can be done by using an appropriate command list reported in the NAV420 user manual.

The command list was written directly to the NAV 420 EEPROM, using the power-up configuration field, so that the configuration settings are used always as default values by the system.

For the onboard software to work properly the NAV420 default settings must be as follows:

Baud Rate: 57600 bps

Packet Output Rate: 100 Hz

Packet Type: NAV mode

AHRS Mounting and Alignment

The CrossBow NAV 420 was installed inside the avionics box of the UNIBO RUAV (see figure 49). The GPS antenna is mounted on the tail boom and is connected to the GPS receiver inside the navigation platform with a SMA jack. The GPS antenna was changed with respect to the one provided by CrossBow: a Geohelix-S GPS Antenna was installed in order to improve GPS signal reception. The Geohelix characteristics can be found in the enclosed manual.

When mounting the NAV 420 some precautions must be taken in order to ensure proper functioning and measurement reliability.

The AHRS unit has its own coordinate system as shown in figure below.

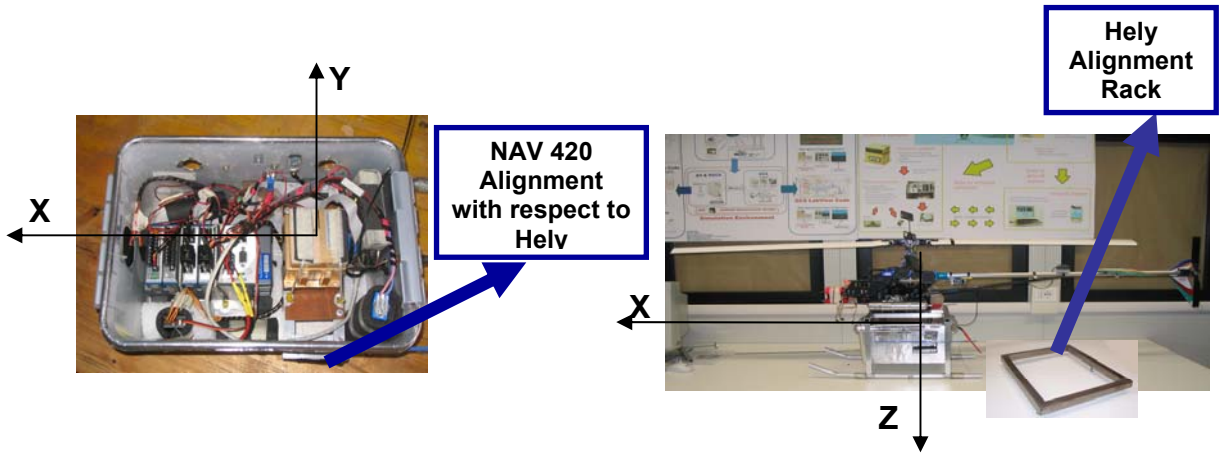


Figure 49: NAV420CA Mounting

The axes form an orthogonal right-handed coordinate system:
X-axis – from face with connector through the NAV 420
Y-axis – along the face with connector from left to right
Z-axis – along the face with the connector from top to bottom.

In this reference system, the direction of positive rotation for the rate is defined by the right-hand rule:
-Pitch is defined positive for helicopter nose up
-Roll is defined positive when the helicopter rolls to the right
-Yaw is defined positive for heading right turn.
-The position output form GPS is represented in Latitude, Longitude and altitude convention while the GPS velocity output is defined in the North, East and Down reference frame [53].

The NAV 420 mounting holes can be used as a reference for aligning the NAV420 sensor axes with the ones of the helicopter.
The NAV420 was installed along the x, y, z axis of the helicopter. Before any flight test can take place, it must be also ensured that the NAV420 is not rotated with respect to the helicopter, that would cause wrong helicopter attitude measurements. At this aim, the AHRS is mounted fixed in heading: alignment was compared to the one available from a small magnetic compass, fixed with the h/c, in order to have a rough feedback about the heading correct mounting.
As for the roll and pitch angle, the AHRS is mounted on a slew able flat plate around the x and y axis.

Before any flight, the helicopter and the NAV420 are both aligned with the g vector and the earth tangent plane so that they can be considered aligned between each other. The helicopter airframe is positioned on a special rack and is aligned by means of a spirit-level and of the rack screws (see figure 49). After that, the flat plate, on which the NAV 420 is mounted, is adjusted (using its own screws) so that the measured attitude is zero and the acceleration is parallel to the g vector.
Usually, once the NAV 420 has been aligned with respect to the helicopter, only attitude periodic checks are needed, if the navigation platform is not moved from its site.

Another important precaution to be taken is that the NAV 420 must be mounted as close as possible to the helicopter Center of Gravity (CG). If it is not mounted at the CG, then rotation around the CG can cause NAV 420 accelerometers to measure acceleration difference equal to the angular rate squared multiply by the distance between the NAV 420 and the helicopter CG. This, in turn, may also affect velocity and position measurements.
The helicopter CG can be easily determined experimentally. The NAV 420 was aligned with the CG in the x and y axes while there is an offset of about five centimeters along the z axis. This small offset, however, will not significantly affect the NAV 420 measurements for several reasons:

- the offset is very small
- the angular rates are usually small since the helicopter doesn't perform extreme maneuvers
- the NAV 420 internal Kalman filter updates velocity and position measurements using GPS information so that possible errors can be partially corrected

Finally, when installing the NAV 420 in a vehicle and the vehicle contains ferro-magnetic parts (as the helicopter for example), it is necessary to perform a magnetometers calibration procedure for hard and soft iron compensation before using it. The several steps to be followed for the calibration procedures are described in the NAV420 user manual and can be performed using the NAVView software, provided by CrossBow. Other calibration procedures are not necessary, since NAV420 internal sensors comes already factory calibrated for temperature bias, scale factor and misalignments.

AHRS Data Acquisition Software

The CrossBow NAV420 provides information to the user by means of a RS 232 protocol. The RS232 data acquisition was not performed using the real time serial port, since acquisition from serial port is well know to be not-deterministic and is, therefore, incompatible with real time critical control loops. For that reason, the RS232 data packet was acquired by means of a FPGA digital input channel (particularly the Slot2/cRIO-9411/DI 0) to guarantee deterministic data acquisition, needed for the control algorithms to work properly.

The full software developed for the NAV 420 is reported in the enclosed CD, together with a step by step explanation. The NAV420 string is acquired by reading directly the RS232 electrical signal coming to the CRIO digital input channel. In order to understand the program, a detail knowledge of the RS232 protocol and of the NAV string contents is needed as well as of the LabView software packet reconstruction methodology.

The program works following the flow chart reported below:

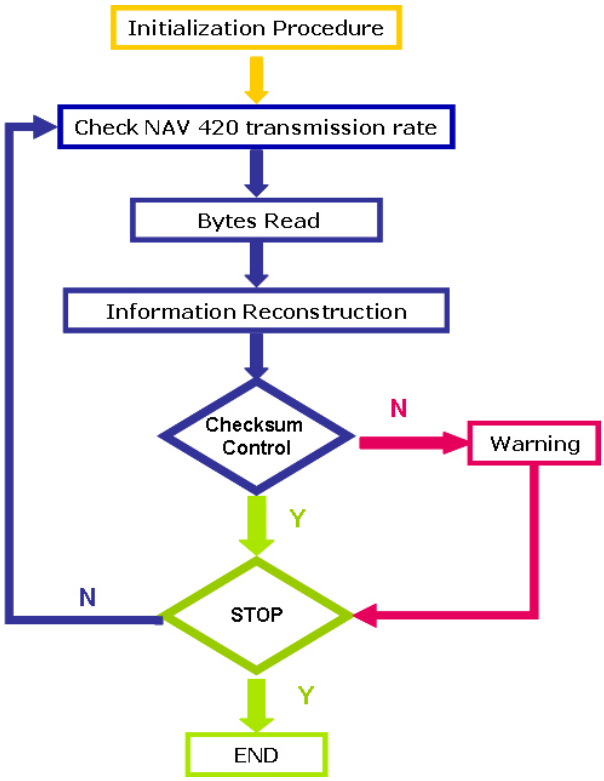


Figure 50: NAV420CA Acquisition Software Flow Chart

1) The program starts with an initialization procedure. This is constituted by a while loop that cicles till a low level signal time interval is found whose length is comparable with the time distance between two consecutive NAV 420 packets. This ensure that the

NAV 420 string is read starting from the first byte of the packet. The wait time value can be estimated from the NAV 420 packet rate and from the baud rate. During the initialization procedure, the time (measured in FPGA clock ticks) corresponding to one bit and half bit of information is also calculated. These two times will be used by the program to correctly read the bit sequence, which composes each byte of information in the NAV 420 packet.

Wait time and FPGA tick counts calculation

Since the NAV 420 output rate is 100 Hz and the baud rate is 57600 bps then:

microseconds corresponding to 1 bit of information = 17,36 μs
time needed for 1 packet transmission = 10 ms

Taking into account that the FPGA works at 40 MHz (which is 40.000.000 tick/s or 1 tick corresponds to 0.025 *μs*), then it is also:

Time in tick corresponding to 1 bit of information = 17,36 [μs] / 0.025[μs/tick] = 694 tick
Time in tick corresponding to 1/2 bit of information = 347 tick

Since the packet in NAV mode is constituted by 37 bytes of information and each byte in the RS232 protocol is constituted by 10 bit (1 start bit + 8 information bit + 1 stop bit), not all the 10 ms contains data information; there will be a certain time, during which the electrical signal will remain low, that can be used as wait time to identify the NAV packet first byte (see figure 51). Hence:

*Bit per packet = 37 byte * 10 bit/byte = 370 bit*
*Actual packet length in ms = 370 bit * 17,36 μs/bit = 6.4 ms*
Low level signal time= Wait time = 10 ms – 6.4 ms = 3.6 ms

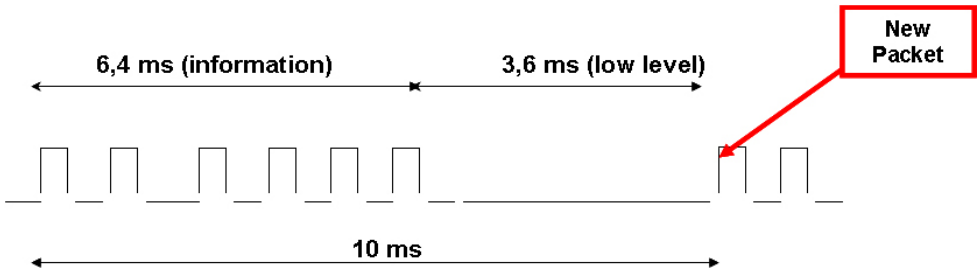


Figure 51: NAV420CA data Packet length time

By default the wait time is set equal to 3.2 ms to be sure that the program is ready to read the first signal rising edge corresponding to the start bit of the packet first byte.

2) During the second step the actual NAV 420 packet rate is read by calculating the inverse of the time difference between two consecutive received packets: actually this procedures requires two packets to be read so, at the program first call, this value is not used.

3-4) In the third and fourth steps, information are read each bit at one time, which are used first to build 1 byte and then respectively a 2 byte or a 4 byte data type.

This procedure is performed taking into account that:

- NAV420 data transmission is a standard RS-232 protocol with 8 data bits, 1 start bit, 1 stop bit, no parity and no flow control. The 8-bit data transmission starts from the least to the most significant bit and uses inverted logical levels (high signal level corresponds to "0", low signal level corresponds to "1"). On the contrary, the RS232 protocol byte transmission is done from the most significant byte to the least significant byte. So for example to transmit a 2 byte information, the RS232 protocol first sends the most significant byte and then the least significant. In turn, each bit, inside one byte is transmitted from the least to the most significant.
- the LabView program language uses boolean arrays to store bit/byte information which uses opposite logic levels with respect to the one of the RS232 protocol. Morevover, LabView associated the significant bits/bytes to the index position inside the array.
-the NAV 420 packet type (in NAV mode) is composed by 37 byte. Data are signed I16 (2 byte) or I32 (4 byte) format, depending on the information type as show in table below:

Bytes	Description	Range	Unit
0,1	Header 'UU'		
2	'N'		
3,4	Roll Angle	[-180 180]	Degrees
5,6	Pitch Angle	[-180 180]	Degrees
7,8	Heading	[-180 180]	Degrees
9,10	Roll Rate	[-630 630]	Degrees /second
11,12	Pitch Rate	[-630 630]	Degrees /second
13,14	Yaw Rate	[-630 630]	Degrees /second
15,16	V_N	[-256 256]	m/s
17,18	V_E	[-256 256]	m/s
19,20	V_D	[-256 256]	m/s
21,22,23,24	Longitude	[-180 180]	Degrees
25,26,27,28	Latitude	[-180 180]	Degrees
29,30	Altitude	[-100, 16284]	m
31,32	GPS ITOW	[0, 65536]	msec
33,34	BIT		
35,36	Checksum		

Table 4: NAV420CA Packet Details (NAV Mode) [53]

For example, to read a 2 byte information data type the following step are needed as shown in figure 52:

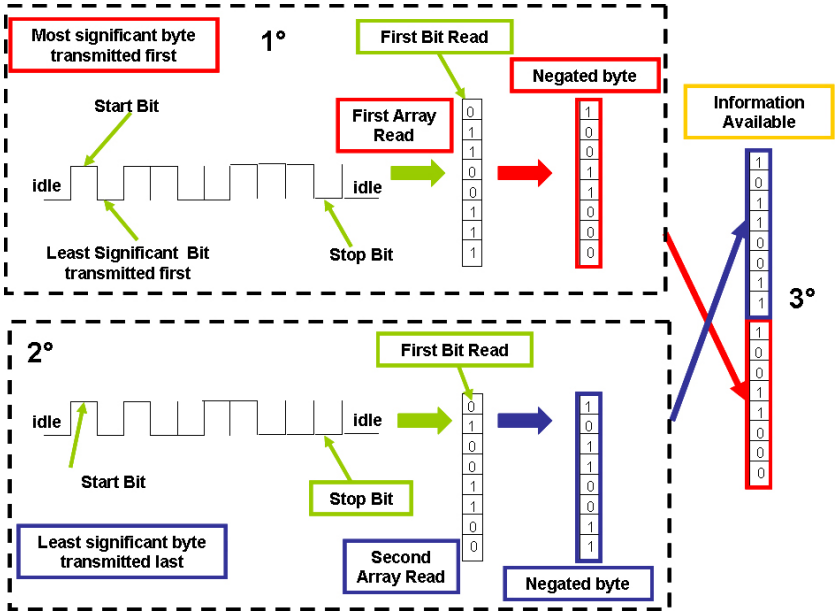


Figure 52: NAV420CA Packet Acquisition Sequence

1°) read the first eight information bit sequence belonging to the most significant byte, put it into a LabView boolean array and then negate it
2°) read the second eight information bit sequence belonging to the least significant byte, put it into a LabView boolean array and than negate it
3°) put these two boolean arrays into a 16 element boolean array with the first byte in the 8-15th position and the second byte in the 0-7th position. For the example reported in figure, the information will be interpreted correctly by LabView as follows:

$1011001110011000 = 2^0 + 2^2 + 2^3 + 2^6 + 2^7 + 2^8 + 2^{11} + 2^{12} = 6349$ (I16 format)

5- In the end a checksum control is performed in order to verify the correct data packet acquisition. The acquisition continues till the program is stopped.

4.4.2 ALTITUDE SENSORS

Altitude sensors measure with reference to sea level (AMSL, Above Mean Sea Level) or the local ground (AGL, above ground level). This kind of data is needed in order to control the altitude of the aircraft depending on the vehicle type and on the modes and location of operation. Operations within ground vicinity, such as landings, usually require absolute AGL measurements or a very accurate terrain database [52]. Available sensors include:

- Sonar (AGL)
- Radar (AGL)
- Laser/Lidar (AGL)
- GPS (AMSL)
- Barometric (AMSL)

Sonar sensors were chosen to measure the helicopter altitude with respect to the ground. Important issues taken into account for the selection of the altitude sensor type, besides the type of measurement, were its accuracy, range and cost, which was a very limiting issue driving the sensor choice. Radar altimeter would have higher AGL measurement capabilities at a comparable resolution, but at a price out of the project budget. Sonar sensors, however, worked very good to test the system operating capabilities at very low

altitude, which can be easily extended to higher altitude once future system improvements will be undertaken.

4.4.2.1 Description

The sonar sensor chosen for the UNIBO RUAV was the SRF08 (see fig. 53) which can deliver helicopter altitude till six meters with a resolution of 2 cm and has a minimum altitude measurement limit of 3 cm. It has a very low voltage and current consumption respectively of 5 V and 12 mA.

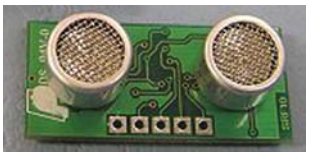


Figure 53: Sonar Sensor SRF08

Usually ultrasonic sensors use transducers to radiate sounds in many different types of patterns, from omnidirectional to very narrow beams. For a transducer with a circular radiating surface vibrating in phase, as is most commonly used in ultrasonic sensor applications, the narrowness of the beam pattern is a function of the ratio of the radiating surface diameter to the sound wavelength at the operating frequency. The larger the diameter of the transducer, as compared to a wavelength of sound, the narrower the sound beam.

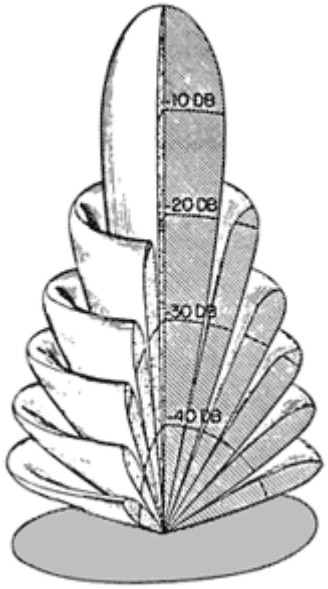


Figure 54: Example of Three-Dimensional Representation of the Sonar Beam Pattern

As can be seen, the sonar sensor produces a narrow conical beam and a number of secondary lobes of reduced amplitude separated by nulls. The beam angle is usually

defined as the measurement of the total angle where the sound pressure level of the main beam has been reduced by 3 dB on both sides of the on-axis peak. However, the transducer still has the sensitivity at greater angles, both in the main beam and in the secondary lobes.

For a symmetrical conical pattern, such as that shown in Figure 54 and typical of ultrasonic sensors, a simple two-dimensional plot known as beam pattern, can describe the entire three-dimensional pattern. The beam patterns of transducers are reciprocal, which means that the beam will be the same whether the transducer is used as a transmitter or as a receiver. Figure 55 shows the beam pattern for the SRF08 sonar sensor as a function of angle. The beam angle is enough narrow, approximately about +45° and -45° [54].
Of course, the presence of secondary lobes may produce unwanted echoes and cause false measurements. Therefore, the sonar sensor was mounted under the avionics box in order to avoid undesired reflection from the helicopter airframe (see fig 39).
Other aspects associated with sonar mounting and operation were:

- sonar measurements depend of course on the helicopter attitude: for the purpose of this work this effect was neglected, since, for typical flight conditions, the helicopter attitude is not very high
- sonar measurements can be affected from false echoes at the ground: since flight test were performed in open field with no presence of obstacle and building in the vicinity, this aspects won't be a real problem for helicopter operation. However, the SRF08 has the possibilities to choose altitude information among 16 different echoes starting from the nearest to the farthest, which could be used for future work improvements.

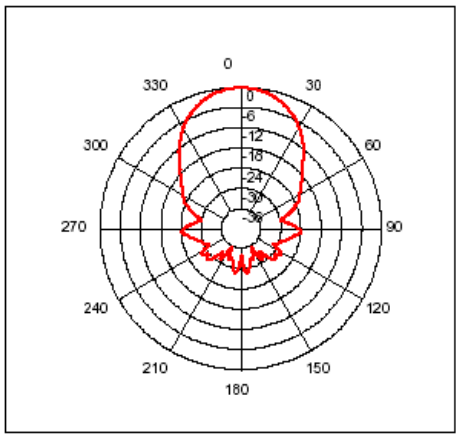


Figure 55:SRF08 beam pattern [54]

4.4.2.2 Sonar sensors data acquisition

Sonar sensor output is provided using an I2C protocol. In order to acquire altitude with the onboard computer the sonar sensor was interfaced with the CRIO using an in-house made interface card. After that the appropriate acquisition software was developed.

Sonar Card Design

The I2C protocol uses two lines (just two wires) to synchronize all data transfer over the I2C bus called SDA and SCL line. The first is the data line and the second is the clock line. The SCL and SDA lines must be connected to the CRIO digital output and input in order to write and read commands on the I2C bus. A third wire is used for the ground and a 5 Volt wire for distributing power to the devices (see figure 56). Particularly:

the Slot2/cRIO-9411/DI 1 is used for reading the clock line (virtual channel SDKR)

the Slot2/cRIO-9411/DI 2 is used for reading the data line (virtual channel SDAR)

the Slot3/cRIO-9411/DO 5 is used for writing the clock line (virtual channel SDKW)

the Slot3/cRIO-9411/DO 5 is used for writing the data line (virtual channel SDAW)

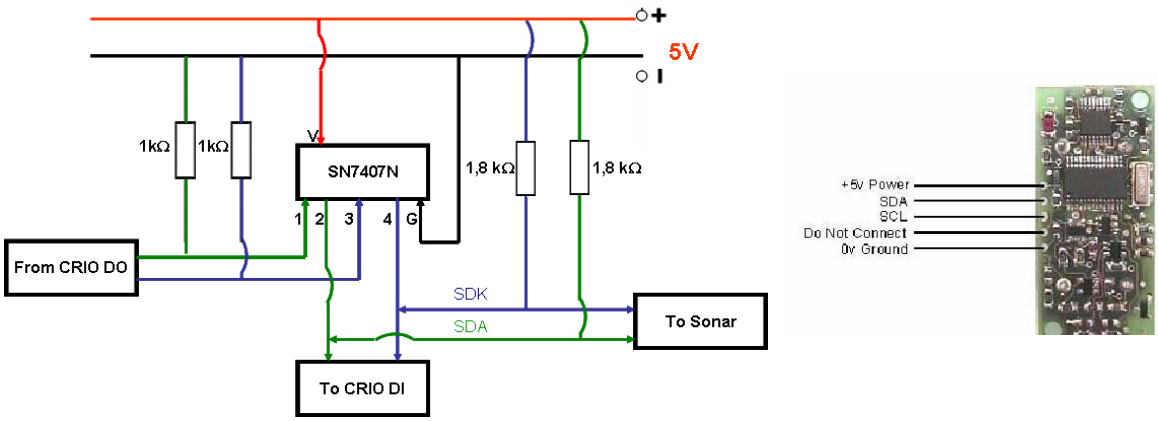


Figure 56: Sonar Acquisition Circuit

The core of the sonar acquisition circuit is shown in figure 56, while the final printed circuit is reported in the enclosed CD.

Particularly:

- a buffer open collector SN7407N was used to protect the devices from short cuts or other problems
- two “pull-up” resistors (1,8 kΩ) were used to pull up the SDA and SCL lines. This is necessary because both the SCL and SDA lines are "open drain" drivers. What this means is that the chip can drive its output low, but it cannot drive it high. For the line to be able to go high, pull-up resistors to the 5v supply must be provided. If

the resistors are missing, the SCL and SDA lines will always be low - nearly 0 volts - and the I2C bus will not work.

- two pull-down resistors (1 kΩ) were used to pull-down the line output from the CRIO DO. These were necessary to adjust the low logical level of the CRIO DO to be compatible with the ones of the SN7407N.
- in the final circuit other components were added to stabilize the voltage line and signals so that the sonar acquisition card can be powered with a maximum input voltage of 9 Volts. The card provides then 5 Volts terminals to distribute the correct power to the devices.

FPGA Sonar data acquisition

The sonar data acquisition software required deep knowledge of the I2C protocol in order to be developed. Therefore, some background is provided here below.

The I2C Protocol

If the CRIO has to talk to a slave (the sonar SRF08), it must begin by issuing a start sequence on the I2C bus. A start sequence is one of two special sequences defined for the I2C bus, the other being the stop sequence. The start sequence and stop sequence are special in that these are the only places where the SDA (data line) is allowed to change while the SCL (clock line) is high. When data is being transferred, SDA must remain stable and not change whilst SCL is high. The start and stop sequences mark the beginning and end of a transaction with the slave device (see figure 57).

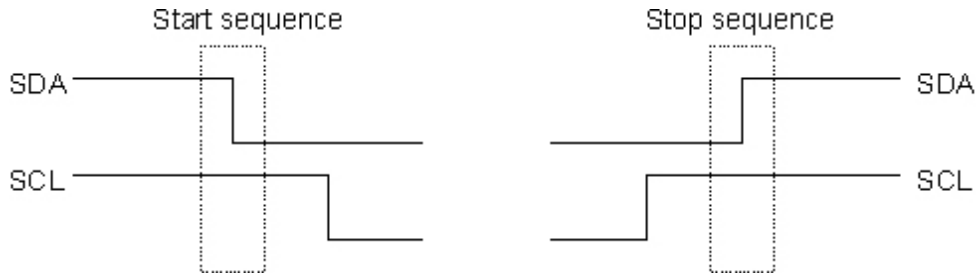


Figure 57: I2C Start and Stop Sequence [55]

Data is transferred in sequences of 8 bits. The bits are placed on the SDA line starting with the MSB (Most Significant Bit). The SCL line is then pulsed high, then low (actually the chip cannot really drive the line high, it simply "lets go" of it and the resistor actually pulls it high). For every 8 bits transferred, the device receiving the data sends back an acknowledge bit, so there are actually 9 SCL clock pulses to transfer each 8 bit byte of

data (see figure 58). If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte. If it sends back a high then it is indicating it cannot accept any further data and the master should terminate the transfer by sending a stop sequence [55].

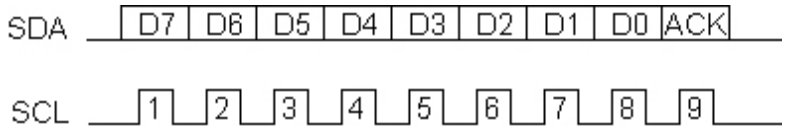


Figure 58: I2C bit transfer [55]

In order to manage the I2C protocol, appropriate command sequences were defined:

START Sequence: for the CRIO to start communication with the sonar
STOP Sequence: for the CRIO to stop communication with the sonar
TX Sequence: for the CRIO to transmit information to the sonar
Get-ACK Sequence: for the CRIO to get acknowledgement from sonar (that means the sonar has received the data). For the Get-ACK Sequence to work properly, it must contains also a Clock stretching wait routines (see software details). The clock stretching is necessary to be sure that the sonar has actually received the CRIO commands and is ready to send the data.
RX Sequence: for the CRIO to read information from the sonar
Give-ACK Sequence: for the CRIO to acknowledge the sonar (that means the CRIO has received data from sonar).

Moreover, the SRF08 sonar has a predefined device address using 7 bits + 1 R/W bit (Read/Write bit) and register addresses (whose values can be found in the SRF08 manual), which must be used for communication with the CRIO.
Particularly :
E0: it is the sonar device address + write bit (it is used when the CRIO wants to write the sonar)
E1: it is the sonar device address + read bit (it is used when the CRIO wants to read from the sonar)

Therefore, communication Sonar-CRIO can be performed on the I2C bus by using the above defined standard sequences, device addresses and register addresses.

Sonar acquisition software

The sonar acquisition software is shown in figure 59 while details of the implemented subVI are reported in the enclosed.

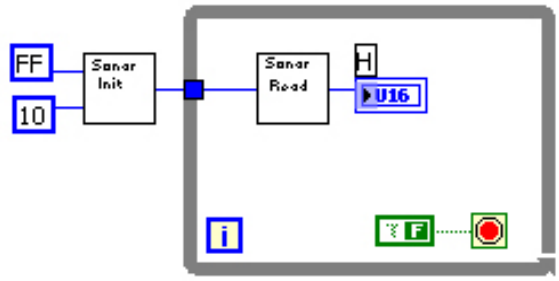


Figure 59: FPGA Sonar Data Acquisition Loop

The software is basically constituted by two main subVI:

- the first one performs an initialization procedure to set the sonar range and gain to appropriate values
- the second one read information when the sonar is commanded to range from the CRIO

For the SRF08 to start ranging in cm, the following instruction must be implemented in sequence:

Initialization procedures

```
i2c_start();          // send start sequence
i2c_tx(0xE0);         // SRF08 I2C address + W bit
i2_get-ack;           // get acknowledgment
i2c_tx(0x02);         // SRF08 range register address
i2_get-ack ;          // get acknowledgment
i2c_tx(0xFF);         // set range to appropriate level (determined experimentally). This
                         value must be set to Hex FF if we want the sonar to range till 6 m
i2_get-ack            // get acknowledgment
i2c_stop();           // send stop sequence
```

```
i2c_start();          // send start sequence
i2c_tx(0xE0);         // SRF08 I2C address + W bit
i2_get-ack;           // get acknowledgment
i2c_tx(0x01);         // SRF08 gain register address
i2_get-ack ;          // get acknowledgment
i2c_tx(0x10);         // set gain to appropriate level (determined experimentally). This value
                        must be set to Hex 10 if we want the sonar to range till 6 m
i2_get-ack            // get acknowledgment
i2c_stop();           // send stop sequence
```

Sonar data read

```
i2c_start();          // send start sequence
i2c_tx(0xE0);         // SRF08 I2C address + W bit
i2_get-ack;           // get acknowledgment
i2c_tx(0x00);         // SRF08 command register address
i2_get-ack;           // get acknowledgment
i2c_tx(0x51);         // command to start ranging in cm
i2_get-ack;           // get acknowledgment
i2c_stop();           // send stop sequence
```

Now after waiting 65mS for the ranging to complete, the following commands are sent:

```
i2c_start();          // send start sequence
i2c_tx(0xE0);         // SRF08 I2C address + W bit
i2_get-ack;           // get acknowledgment
i2c_tx(0x02);         // SRF08 range register address; before reading from the sonar, it is
                        necessary to tell the sonar which of its internal addresses we want to
                        read.
i2_get-ack;           // get acknowledgment
i2c_start();          // send again start sequence
i2c_tx(0xE1);         // SRF08 I2C address + R bit
i2_get-ack;           // get acknowledgment
```

```
i2_rx_high;          // read the most significant byte (sonar data are in the U16 format and
                        therefore requires two byte)
i2_give-ack;         // give acknowledgment
i2_rx_low;           // read the least significant byte (sonar data are in the U16 format and
                        therefore requires two byte)
i2_give-ack;         // give acknowledgment
i2c_stop();          // send stop sequence
data rebuilt;        // builds altitude information in a 16 element Boolean array (in a
                        similar way as is done for the NAV 420 I16 data)
```

4.5 ACTUATORS

Servo actuators allow accurate helicopter commands thanks to the on-board circuit. They are controlled by a Pulse-Width-Modulated (PWM) signal, where the desired servo motor angle is usually proportional to the pulse width (see further in Section 4.5.1).

Five servo actuators are currently mounted on the helicopter which must be powered at 5V:

- S9202 for throttle control
- two S9405 for lateral and longitudinal cyclics controls
- S9255 for collective control
- S 9252 for tail control which can provide a 6 kg cm torque

The servo actuators control circuit is show in figure 60.

During the helicopter flight, servo actuators are controlled either by the RC pilot via radio or by the onboard computer, when the RUAV flies autonomously.

The core of the onboard actuator circuit are 5 helymodel switches which are used to change the helicopter flight mode. They have 3 input cables and one output cable which brings signal commands, power and ground to the servo actuators.

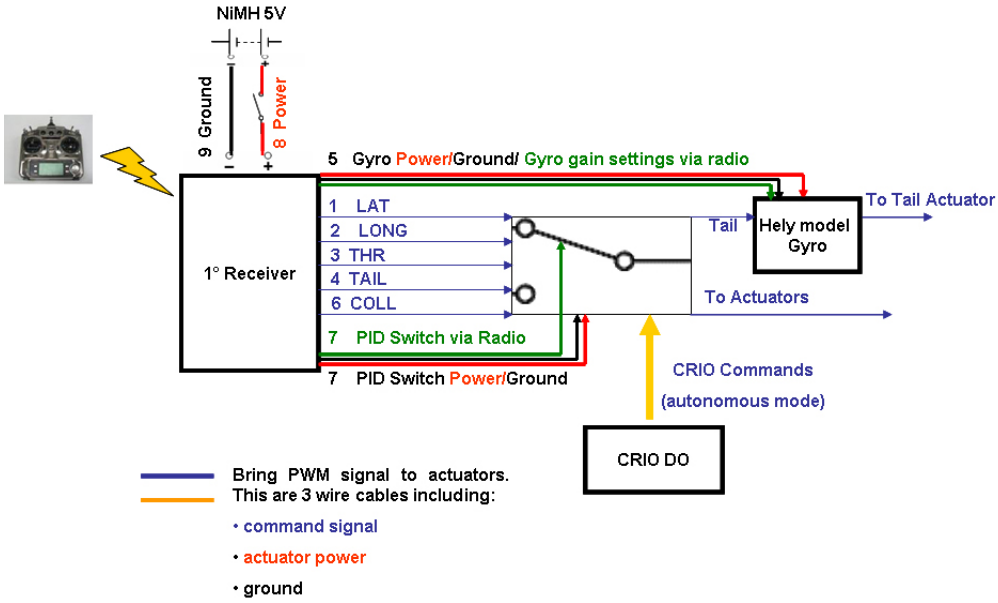


Figure 60: Servo Actuators Control Circuit

The needed input are:

1- The first input cables of each switch are connected together and plugged into channel 7 of the RC receiver so that the pilot can switch from/to manual-autonomous flight mode
2- The second input cable of each switch is connected to the radio receiver channel, which receive signal from the RC pilot via radio. Five switches were used since five commands are necessary to control the helicopter
3- The third cable of each switch is connected to the related CRIO digital output channel, to receive computer input when the helicopter flies in autonomous mode. The CRIO digital output channels are:
   the Slot4/cRIO-9411/DO 0 for the lateral cyclic
   the Slot4/cRIO-9411/DO 1 for the longitudinal cyclic
   the Slot4/cRIO-9411/DO 2 for throttle
   the Slot4/cRIO-9411/DO 3 for tail
   the Slot4/cRIO-9411/DO 4 for collective

The switch output signals are send directly to the servo actuators apart from the PWM tail commands. The tail command is sent to the tail actuator passing across the onboard hely gyro.

Actually, the onboard hely model gyro is coupled with a control unit, provided by the factory, and contains an Heading Lock Angular Velocity Control System (HL-AVCS),

which is used either to stabilize the helicopter in heading or to control the helicopter heading. From now on, the gyro sensor coupled with its control unit will be referred to as gyro system.

While all PWM signal outputs control directly the servo motor angle, this is not true for the PWM tail signal which, actually, is used as input for the gyro system. When the radio is switched on for the first time, the gyro control unit reads the tail PWM signal coming from the radio (which correspond to zero yaw rate since the helicopter stands at the ground and the gyro sensor measures zero yaw rate) and is initialized.

The initial PWM tail command (or the CRIO PWM tail signal) is, therefore, perceived by the gyro control unit as reference yaw rate to be maintained by the helicopter. If the PWM command is equal to the initialization value, the reference yaw rate to be maintained is zero, otherwise it is perceived as a reference of constant yaw rate, whose value depends on the commanded PWM (see fig. 61).

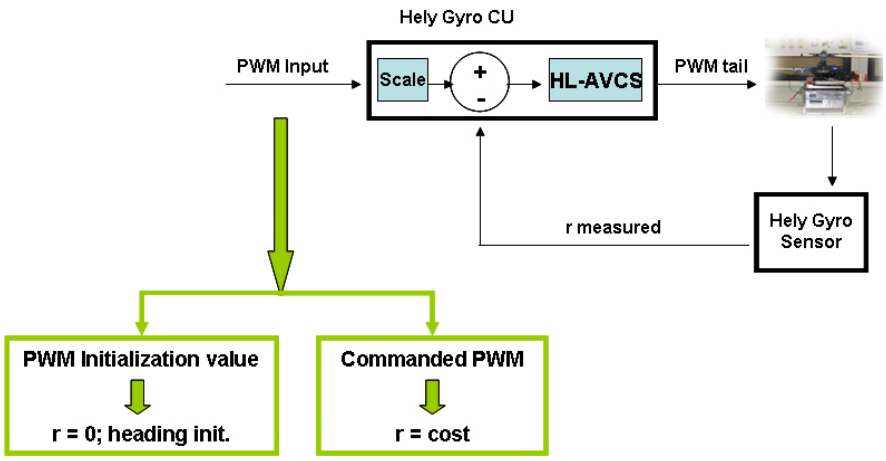


Figure 61: Tail & Helicopter Gyro System Interaction

-If during the flight the helicopter experiences a perturbation in yaw (not due to a pilot or computer command), the gyro feels a change in the helicopter yaw rate response, which is communicated to the gyro control unit. In turn, the gyro control unit will send a PWM signal to the tail actuator till the helicopter yaw rate is driven to zero; moreover, since an Heading Lock AVCS is implemented on the gyro control unit the helicopter returns also to the initial heading, while the yaw rate is smoothed to zero.

-If the RC pilot or the CRIO send to the gyro control unit a PWM signal, different from the one read during the initialization process, this is perceived by the control unit as a new reference yaw rate to be maintained; the control unit will send PWM signal to the tail servo in order to maintain the commanded yaw rate.

The commanded PWM, sent to the tail servo, is generated by the gyro control unit based on a gain settable by the user.

4.6.1 PULSE WIDTH MODULATION-SERVO ANGLE CURVE

Pulse Width Modulation (PWM) is a technique commonly used to represent an analogue signal using digital circuitry. It involves the switching on and off of a digital output at a fixed frequency (switching frequency f_s), but with varying times of either on or off. The ratio of on-time to the total period ($T = 1/ f_s$) is called the duty cycle (d):

$$d= T_h/T \tag{4.1}$$

where T_h denotes the PWM on-time.

RC equipments, such as servos, typically use PWM signals for their control input. As opposed to standard PWM signals where the signal value is dependant upon the duty-cycle, RC equipment use the actual pulse-width (in seconds) to represent the signal. Furthermore, the RC PWM signals usually has a standard frequency range between 20Hz and 200Hz. The servo actuators used on the helicopter operate at a PWM frequency of 50 Hz.

Depending on the PWM actual pulse-width, the servo actuators rotates of a certain angle (see figure 62), which can be easily identified experimentally.

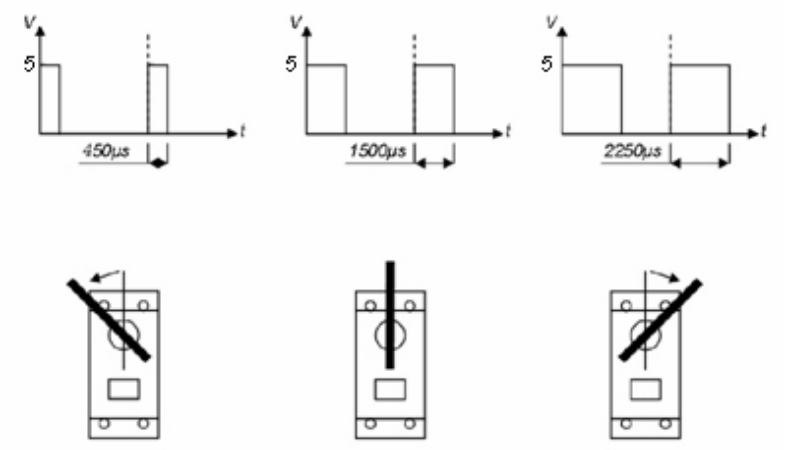


Figure 62: PWM pulse width and servo angle rotation

The servo Angle-PWM curve was determined by means of the experimental equipments illustrated in figure 63.

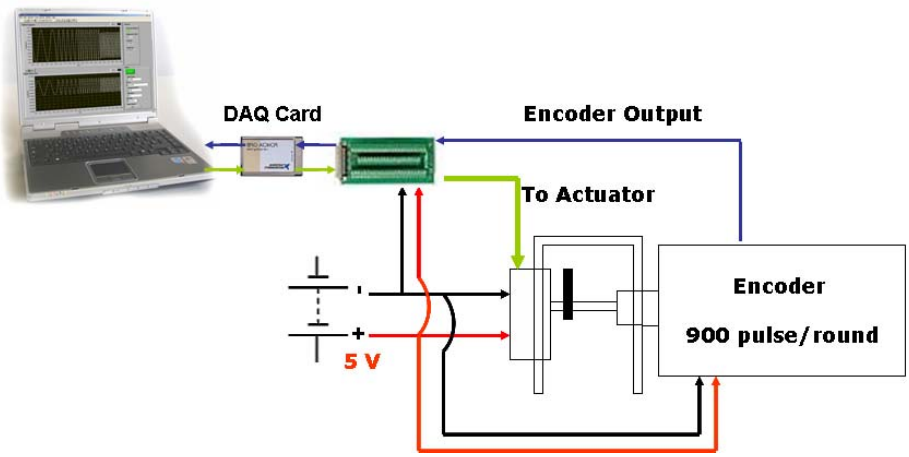


Figure 63: Experimental Set-up for Servo Angle-PWM curve determination

The actuator was coupled with an optical encoder in order to measure the angular displacement corresponding to a PWM servo command.

An encoder is a device that can converts a rotary displacement into digital or pulse signals. The most popular type of encoder is the optical encoder, which consists of a rotating disk, a light source, and a photodetector (light sensor). The disk, which is mounted on a rotating shaft, has patterns of opaque and transparent sectors coded into the disk (refer to figure 64). As the disk rotates, these patterns interrupt the light emitted onto the photodetector, generating a digital or pulse signal output. If the actuator arm is connected to the encoder shaft, the encoder disk rotates each time the actuator is commanded to rotate. Therefore, the encoder signal output will be broken when an opaque disk line is between the emitter-detector pair. It is the monitoring of this on-off pattern which allows the actuator angular displacement to be measured.

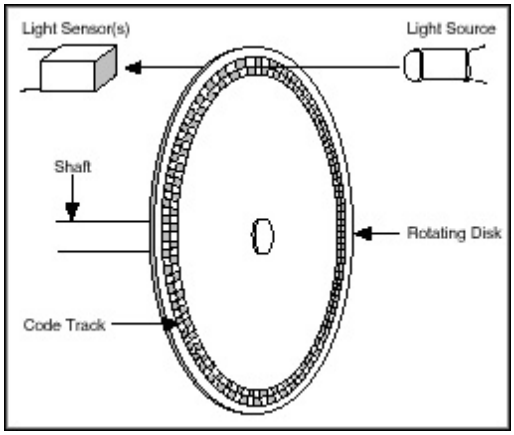


Figure 64: Optical Encoder Principle [56]

By counting the number of the encoder output pulses using a DAQ card, it is possible to know the rotation angle corresponding to a PWM actuator input [56]. At this aim, the phase A encoder cable was connected to a counter input channel of a NI PCMCIA shown in figure 63, while a PWM actuator signal was generated using the PCMCIA counter output.

The software developed for this experiment is reported in figure 65.

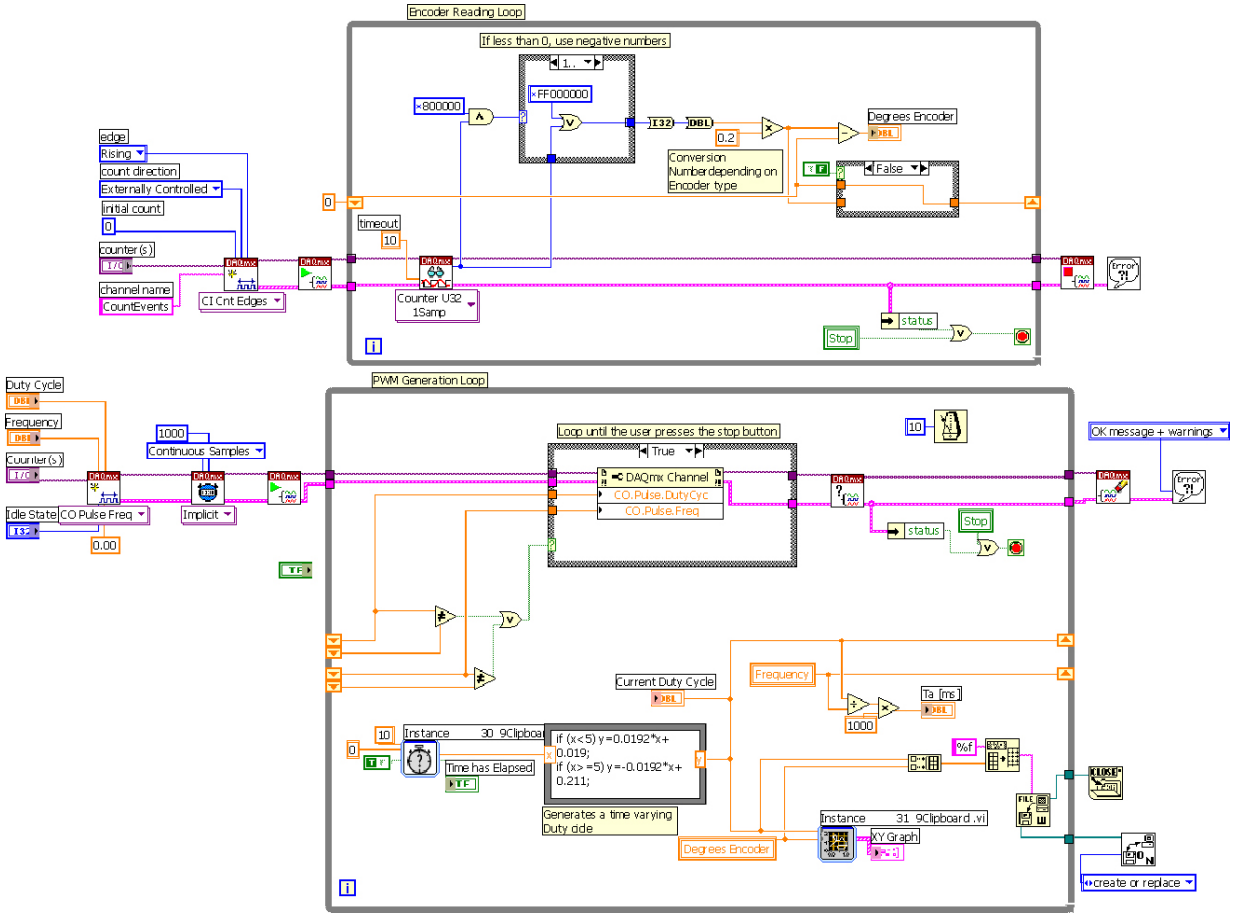


Figure 65: LabView™ Software for Encoder Signal Acquisition and PWM generation through DAQ Card

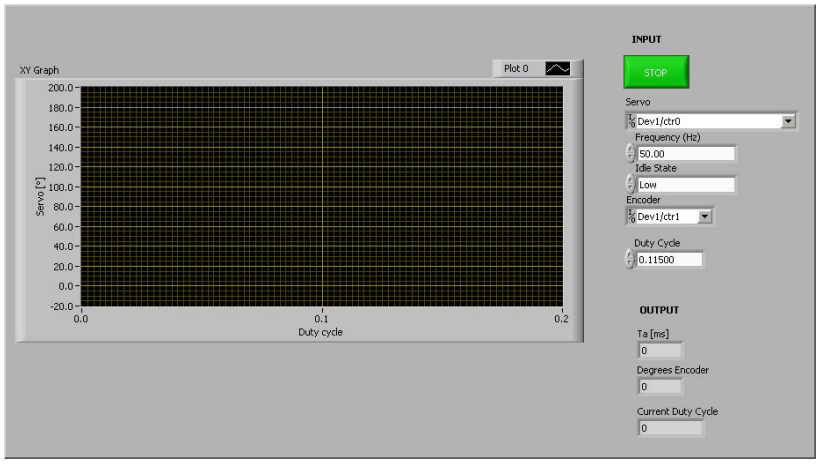


Figure 66: Front Panel of the software reported in figure 65

It is constituted by two parallel while loops:

-the first one is used for PWM signal generation and data logging: for the purpose of this work a time varying PWM pulse width was generated by changing the PWM duty cycle with time. The PWM frequency was left fixed and equal to 50 Hz

- the second one is used to count the encoder pulses and convert it into an equivalent angle measurement. At this aim, it must be taken into account that the encoder, available at the Hangar Laboratories, has different decoding possibilities depending on its usage. If only phase A cable is used, the encoder is not able to discriminate the sense of rotation (which, however, is not a stringent requirements for this work) and work in X2 mode (which means X2 resolution multiplication). Therefore, in order to calculate the angular servo displacement the following formula must be used:

$$Amount\ of\ rotation\ (°) = Counts * 360° / 2N \tag{4.2}$$

where N is the pulse/revolution. For the selected encoder, N is equal to 900 pulse/revolution; hence the scale factor is 0,2 °/Counts, which correspond also to the encoder angular resolution.

The acquired data were processed using Matlab curve fitting tools, which yielded to the following PWM width-Servo Angle curve:

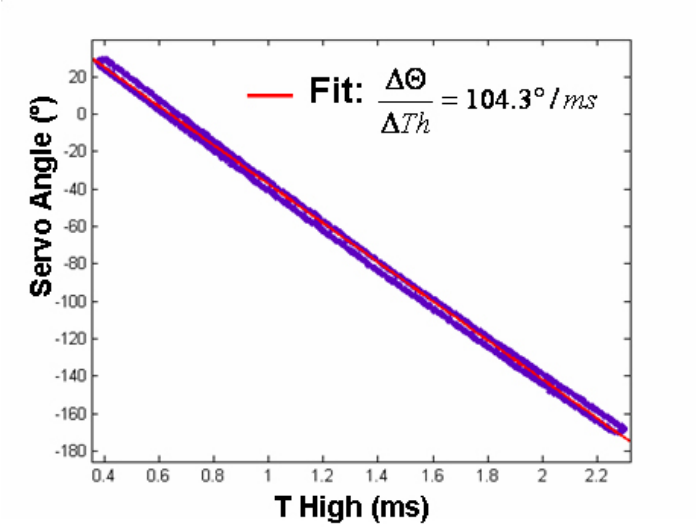


Figure 67: PWM on-time-Servo Angle curve

The servo angle is a linear function of the PWM on-time and the scale factor was found to be:

$$\frac{\Delta\Theta}{\Delta Th} = 104.3° / ms \tag{4.3}$$

which means that one degree servo rotation corresponds to 9,6 μs PWM on-time.

4.5.2 ACTUATORS SIGNAL ACQUISITION AND GENERATION SOFTWARE

For the helicopter to fly autonomously, PWM signal outputs must be generated by the onboard computer for the servo actuators. Furthermore, if a dynamic helicopter model must be identified for autopilot design, helicopter responses to pilot PWM inputs must be also recorded. For that, a FPGA software was developed both to generate PWM output signals and to acquire pilot PWM input commands.

PWM signal acquisition software

The PWM radio input signals have been acquired by measuring the corresponding PWM on-time in microseconds, using the FPGA digital input channels. The channels configuration is as follows:

Slot1/cRIO-9411/DI 0	lateral cyclic pitch
Slot1/cRIO-9411/DI 1	longitudinal cyclic pitch
Slot1/cRIO-9411/DI 2	throttle
Slot1/cRIO-9411/DI 3	tail
Slot1/cRIO-9411/DI 4	collective pitch
Slot1/cRIO-9411/DI 5	PID on/off (radio channel 7)

The software is reported in figure 68.

Basically, the software is constituted by a while loop running at 1 MHz (1 loop every one microsecond). The PWM on-time is measured by creating a virtual microsecond counter. Every microsecond, the digital input channels are read and a microsecond counter (each for one channel) is updated if the input signal logical level is high, otherwise the microseconds counter is re-initialized to zero. This way, the PWM pulse with is measured with a resolution equal to one microsecond.

Moreover, two further remarks must be taken into account:

- the counter variable, actually incremented every microsecond, is a value placed in a virtual memory (the LabView shift register). The corresponding PWM pulse width value is updated only when the first low bit is read. This way, the PWM commands

are all updated with minimum time latency and the software doesn't yield to false transient measurements.

- the program outputs are the 5 radio commands in μs (the measured pulse width for each channel) and a boolean value for the PID on/off channel. This boolean value will be used to enable or disable the autopilot in the control loop (TRUE means autopilot ON, while FALSE autopilot OFF)

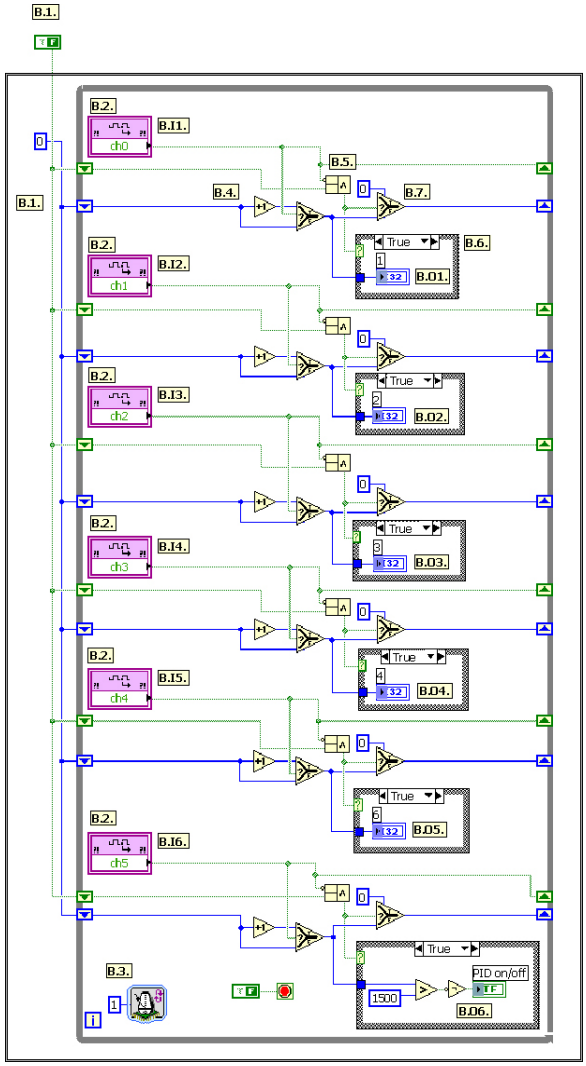


Figure 68: Actuators PWM Acquisition Software

PWM generation software

The PWM output signals have been generated using the FPGA digital output channels, configured as follows:

Slot4/cRIO-9474/DO 0	Lateral cyclic
Slot4/cRIO-9474/DO 1	Longitudinal cyclic
Slot4/cRIO-9474/D2	Throttle

Slot4/cRIO-9474/D3	Tail
Slot4/cRIO-9474/D4	Collective

The software is reported in figure 69.

The software core is the PWM generation subVI, which is able to generate a PWM signal based on the PWM period input (which is fixed to 20000 µs by the servo actuator frequency) and the PWM on-time input (which is provided by the control loop). This subVI is used five time (one for each helicopter command) inside a PWM generation while loop (see fig. 69).

Basically, the program creates a virtual clock whose time value ranges between 0-20000 µs. When the virtual clock time value reaches 20000 µs, it is reinitialized to zero. Each loop, the virtual clock value is compared with the PWM on-time input. If the virtual clock value is less then the PWM high time, a Boolean TRUE output is generated, otherwise the output is driven to FALSE.

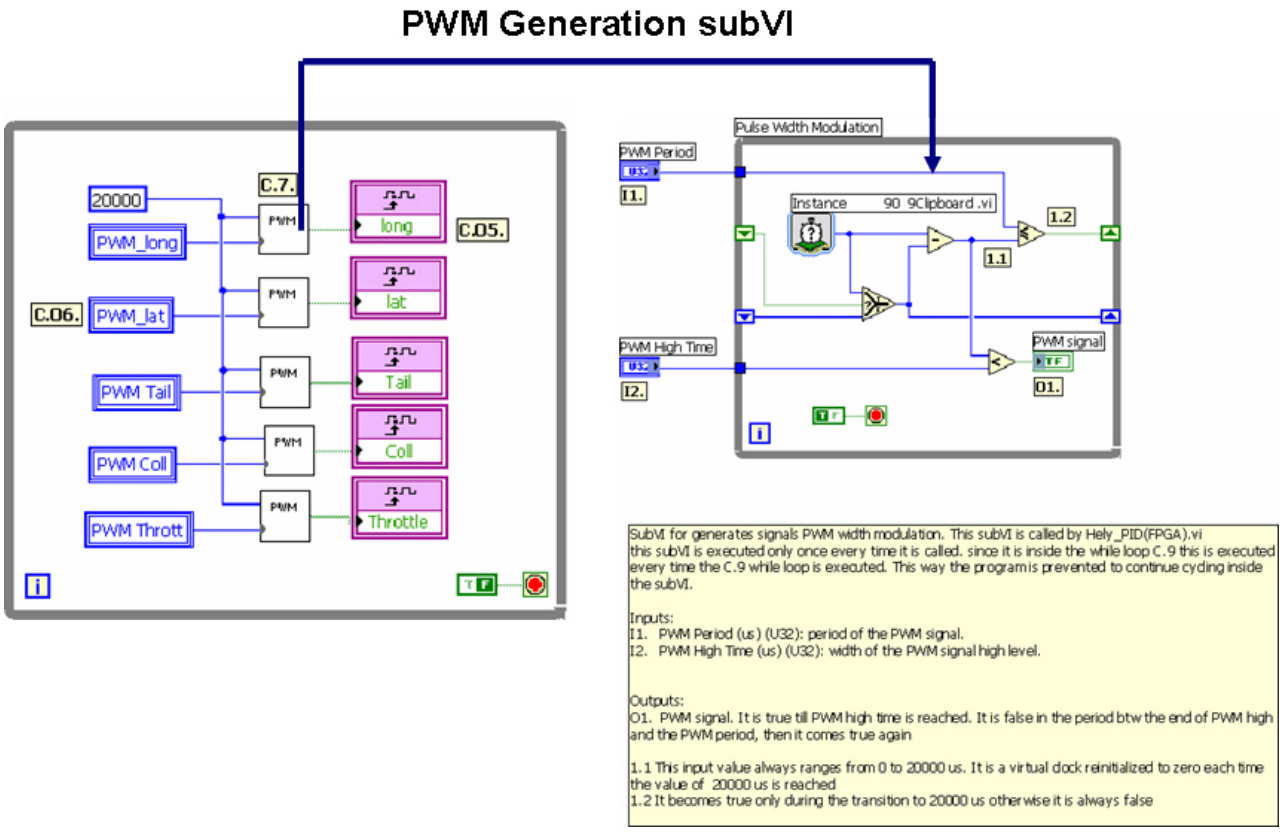


Figure 69: Actuators PWM Generation Software

4.6 DATA LINK

Usually, data link are used for unmanned vehicles to send commands and receive telemetry or payload data and can be divided into digital and analog links. An example for an analog link is a UHF video signal transmission. Digital links provide a way of communicating between ground and vehicle-mounted computers. The frequency band a data modem operates, affects its data rate. Typically, the higher the frequency, the higher the data rate. The frequency also affects the range of the data link. Lower frequencies typically offer a greater range than high frequencies. Furthermore, the higher the frequency, the greater the Line-of-Sight problem, i.e. the ability to penetrate obstacles like buildings. Common data links in the 2.4 Ghz band are more easily "blocked" than that in the VHF frequencies. Also, for unmanned vehicle operation a remote pilot data link is often used to steer the vehicle manually for some phases of the flight [52]. The radio link used for the UNIBO helicopter works at a frequency of 43.835 MHz. Instead, a common WIFI access point (fig. 70), in the frequency of 2.4GHz, is used to perform the data link between the onboard computer and the ground control station. Depending also on the operating environment, the data link range is about 200m-300m, which is anyway quite enough for the goals of the UNIBO RUAV project.

Figure 70: RUAV WIFI Access Point

4.7 HARDWARE INTERFACING, WIRING AND MOUNTING

The RUAV hardware was assembled together placing attention to accessibility, flexibility and modularity. A commercial off-the shelf plastic box was selected to house all avionics components, which was installed under the modified landing gear. The plastic box cover

was suspended to the landing gear by means of rubber shock mounts for vibration isolation (see fig. 72). The very light weight plastic box can be attached to the cover and easily removed, when maintenance or other works needs to be done on the avionics components. Moreover, this mounting system allows the structure of the plastic box to achieve its rigidity, which is of course necessary to perform good flight tests. The tail boom provides also installation points for the GPS antenna, enabling firm fit and leaving the boom structure unchanged. The sonar sensor was appended under and outside the avionics box (see section 4.7.1 for vibration isolation).

A schematic wiring diagram of the vehicle-mounted avionics is depicted in figure 71.

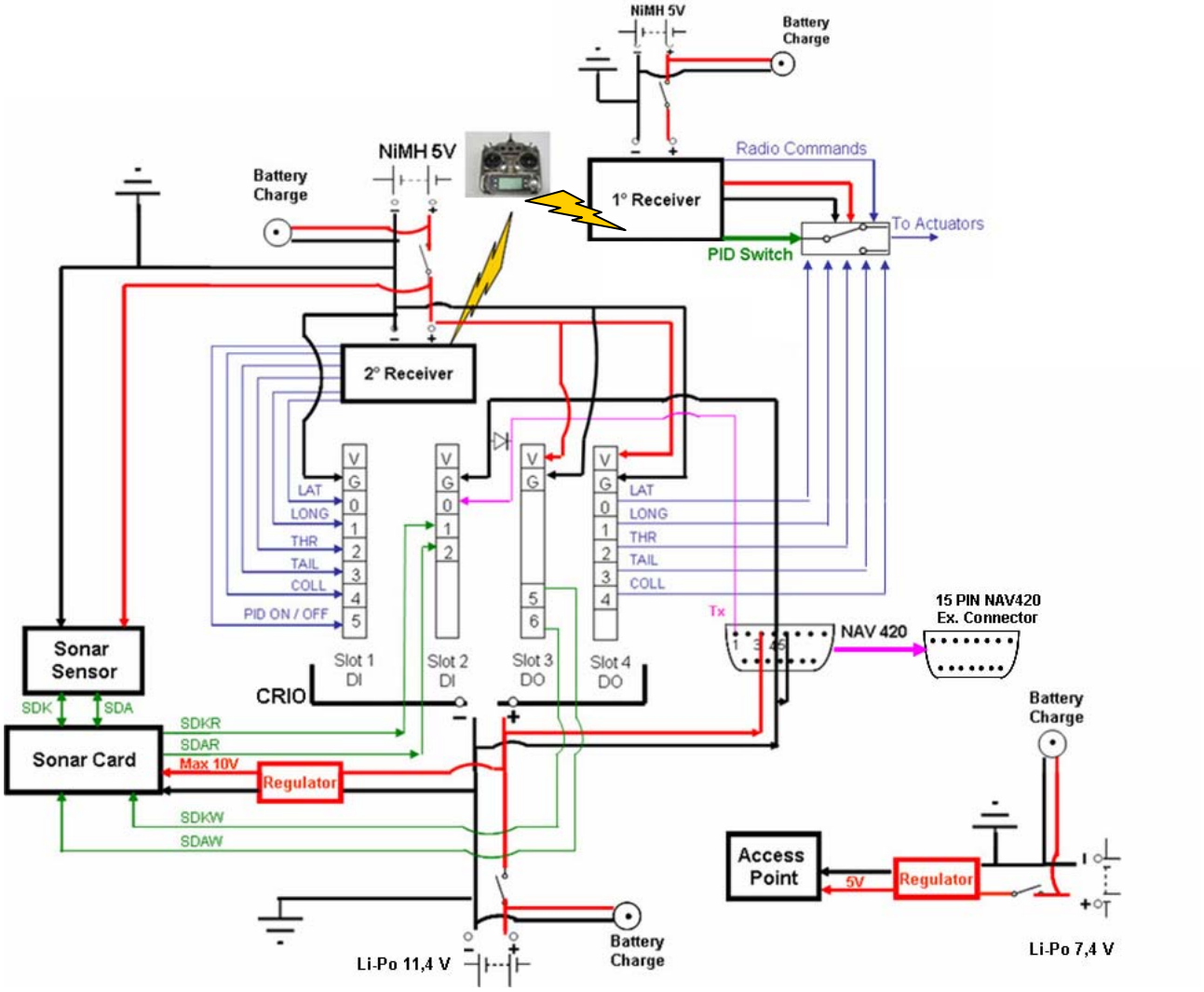


Figure 71: RUAV Schematic Wiring

Redundancy

The installation of a completely redundant avionics system was of course prevented from the small size of the UNIBO RUAV. Therefore, only a minimum safety system was installed, which was anyway enough for the purpose of this project. As discussed in section 4.5, the core of the RUAV minimum safety system is the electronic switch used for disabling the onboard computer in event of system failures. In this perspective, two separate radio receivers were mounted on the helicopter :

-one inside the avionics box (second receiver in figure 71), whose channels are connected to the CRIO digital input channels, for data acquisition

-one mounted on the helicopter airframe outside the avionics box (first receiver in figure 71), which is fully electrically separated from the other avionics and is used only by the RC pilot when the helicopter flies in manual mode. The first receiver power system is also fully independent from the one of the other avionics box equipments.

Power system & Electromagnetic Interference Shielding

All modules are powered by means of onboard batteries. The CRIO and the AHRS requires a 11-12V DC power connection: 3200 mAh Lithium Polimer Battery were used, which combines very light weight with long time power supplies (this battery package allows almost two hours autonomy at a "price" of 150 gr). For the same reason, a 7,4 V Litium Polimer battery package was used to power the data link access point. Since the access point requires 5 V power supply, this battery package is connected to a voltage regulator. The same was done for the 12 V battery package, which supplies also 9 V power to the sonar interface card. The two radio receivers are powered by two 5V NiMH separate batteries package to improve safety. The NiMH battery used for the second receiver, located inside the avionics box, is utilized also to power the sonar and to set the voltage level of the digital output channels of the CRIO.

The power panel is mounted on the side of the avionics box and includes power switch-on buttons, external interface ports for batteries recharging. Special attention was placed also to accessibility of the hardware interfaces: the panel comprises also a serial user interface port and easy access to the CRIO Ethernet port, for easy connection of the avionics to the planned hardware in the loop simulation system or to the ground control station computer (if necessary).

All battery packages are installed so that they can be easy accessible for package replacement with minimum efforts, if long flight tests have to be performed. The wires

inside the avionics box are tied to several mounting points and to each other, in order to prevent any shaving of the insulation.

The CRIO and the AHRS are already factory endowed with sufficient EMI shielding such that it cannot interfere with other equipments. Moreover, the full avionics box was coated with 2 layers of aluminium foil, in order to prevent any EM interference from the pilot radio transmitter or other external disturbances. All aluminum parts were also electrically connected and common grounded.

The two radio receiver antennas were left hanging under the helicopter, which was found to be their best position after several flight tests.

4.7.1 VIBRATION ISOLATION

Electronic circuits and sensors can be affected by harmful vibrations from the engine and rotors. Particularly, the AHRS, GPS antenna, the onboard computer and the sonar altimeter are likely to produce faulty readings with inadequate vibration isolation or may be subjected to damage, if their operational vibration range is overcome. Therefore, the avionics box was appended under the landing gear by means of four elastomeric silent blocks at its corners, which can be seen in figure 72. The dampers were mounted symmetrically with respect to the avionics box centre of gravity, in order to optimize the load distribution on the isolators. Moreover, the AHRS, the CRIO and the other electronics were isolated inside the avionics box by means of neoprene strips. As the GPS antenna and the helicopter gyro are mounted on the airframe structure, they needed separate protection from harmful vibrations. They were isolated by using short pieces of special hely-model rubber, that effectively attenuated vibrations. The sonar sensor was appended at the bottom of the avionics box, by means of a small plastic box isolated with rubber.

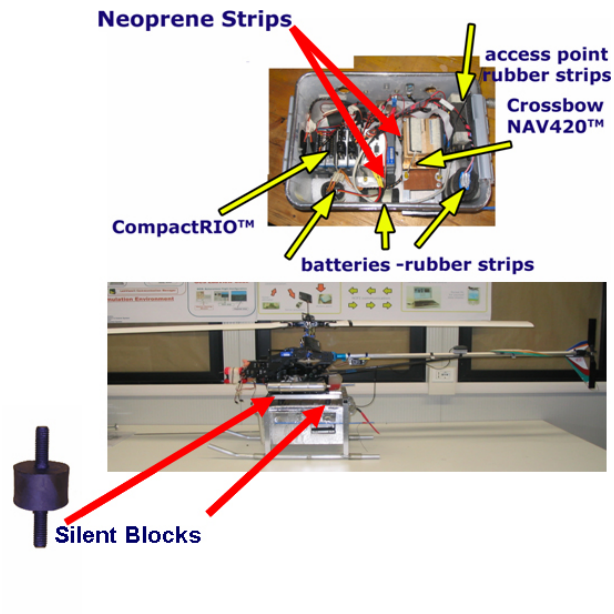


Figure 72: Avionics Vibration Isolation System

A good criteria for chosing elastomeric dampers is that the critical frequency of the shock mounts must not be close to any produced by the rotorcraft at its normal operation point. Figure 73 shows a typical diagram of resonant transmissibility versus damper frequency. If the damper work frequency is higher than its resonant frequency, than vibrations can be effectively attenuated [57].

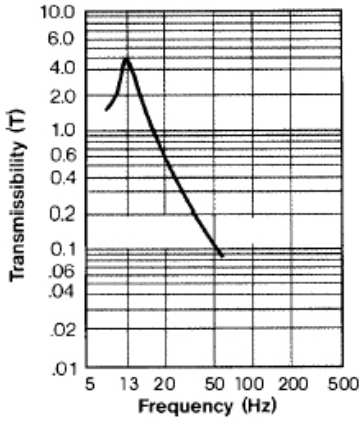


Figure 73: Typical diagram of resonant transmissibility versus damper frequency

With the selected shock mounts, and an avionics box weight of 5 kg, a natural frequency of 15 Hz can be expected. This frequency is far enough from the closest frequency of the system, the rotor-induced oscillations at about 22 Hz, to prevent any adverse effects. Moreover the effect of the neoprene strips should help increasing the damping effect as was demonstrated experimentally (see section 4.7.1.2).

4.7.1.1 Vibration Load Experimental Test

The vibration isolation system was tested by means of the experimental test bed illustrated in figure 74.

During experiments the onboard computer and AHRS were replaced with equivalent metal part to prevent any damage due to unknown vibration effects. Accelerometer were mounted on the landing gear, on the avionics structure after the dampers, on the CRIO and the AHRS to measure the damping effects at different points.

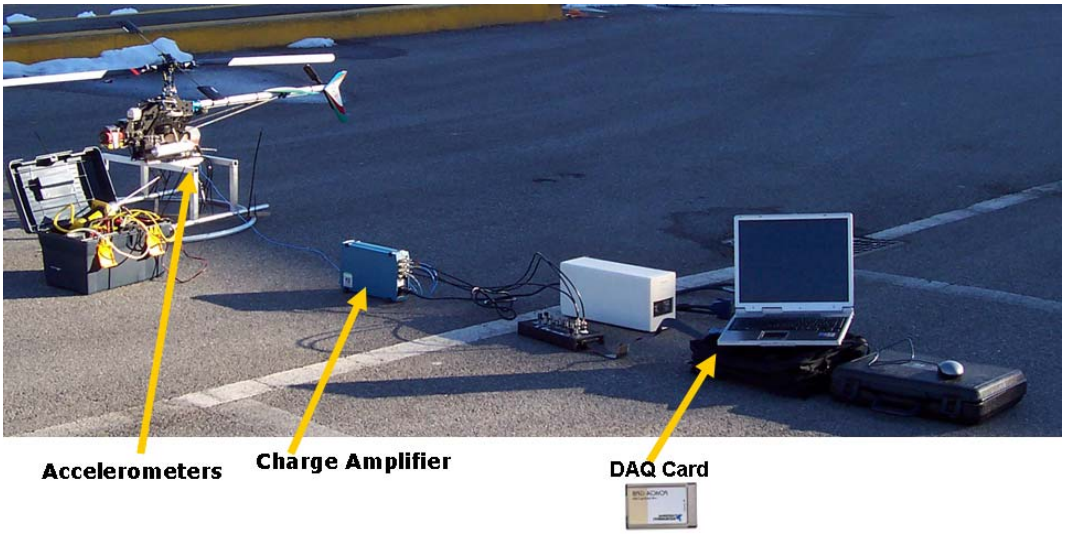


Figure 74: Experimental Data Acquisition System

The accelerometers were connected by means of BNC cables to a charge amplifier. The output signal from the amplifier was acquired by means of a data acquisition card installed on a laptop computer. An appropriate software was also developed to acquire accelerometer outputs, which is reported in figures 75-76.

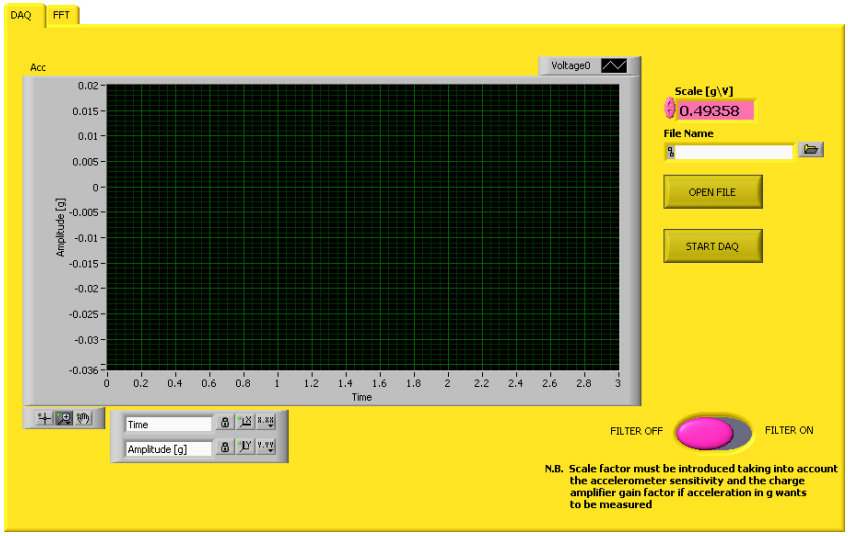


Figure 75: Acquisition Software Front Panel

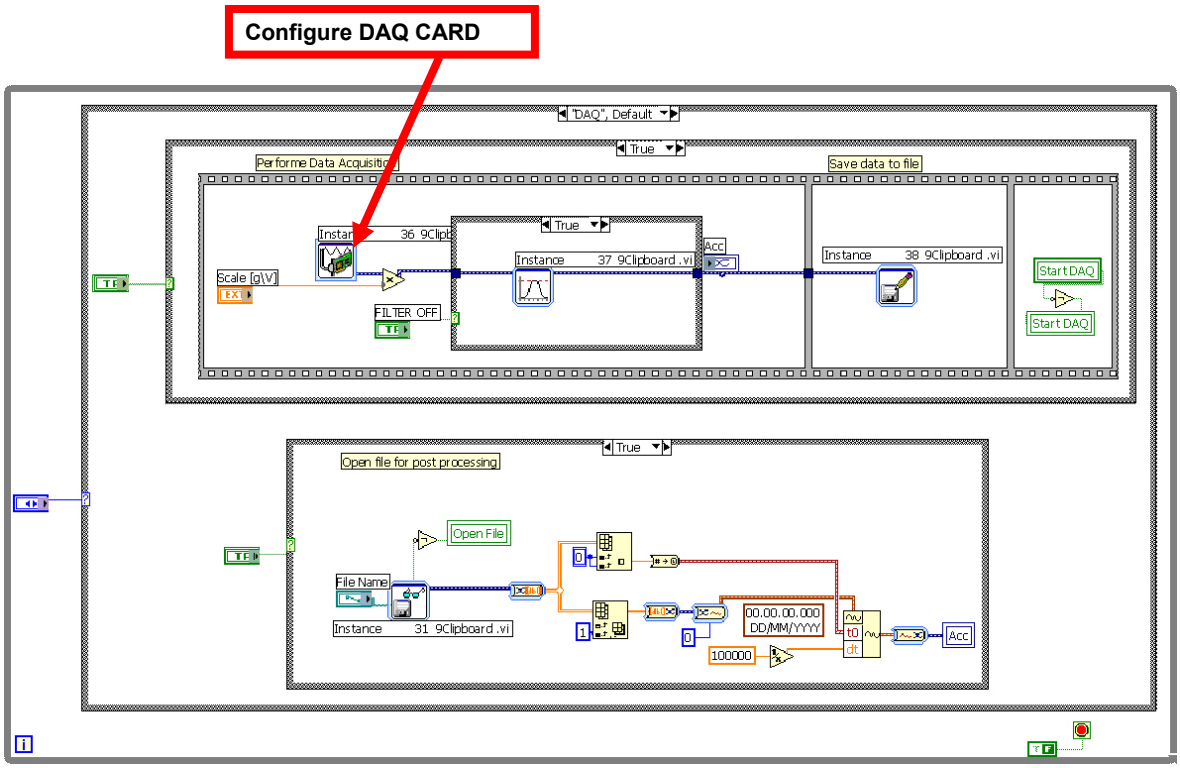


Figure 76: Accelerometers Data Acquisition Software

The program can be used both for acquiring accelerometer outputs form the DAQ card and for data post-processing by pressing the related button at the front panel. If it is used for post-processing, the data file must be selected at the prompt window and frequency analysis can be performed. If the program is used for data acquisition, the DAQ Card must be configured appropriately through the DAQ assistant indicated in figure above. The accelerometer scale in g/V must be given as input, which can be calculated knowing the charge amplifier gain and the accelerometer sensitivity (eq. 4.4). Once data are acquired, they are saved in a user defined file for post-processing.

In the software illustrated above, only one channel of the DAQ card is configured as example, but other channels can be easily added, depending on the experimental set-up.

Tables 5-6 report the experimental set-up configuration used for the purpose of the vibration tests.

Accelerometer Name	Sensitivity [mV/g]	G	Scale Factor [g/V]
89158	10.23	1	97.752
89160	10.91	1	91.659
89161	10.19	1	98.135
89162	10.40	1	96.154

Table 5: Accelerometers Characteristics

	DAQ Card Settings
Sample Frequency [Hz]	51200
N° of Samples per Channel	307200
N° of Channels	4
Channel Max-Min V	different according to test

Table 6: DAQ Card Settings

By denoting with:

x measured signal

G amplifier gain

S transducer sensitivity [mV/g]

the scale factor in g/V can be calculated through the following formulas:

$$x[g] = \frac{x[V]}{G\,S\left[\frac{mV}{g}\right]} 1000\left[\frac{mV}{V}\right] \qquad \text{or} \qquad x[V] = \frac{x[g]\,G\,S\left[\frac{mV}{g}\right]}{1000\left[\frac{mV}{V}\right]} \tag{4.4}$$

The DAQ Card settings were chosen as follows:

- the sample rate depends on the DAQ card and the channels number. The chosen acquisition rate is the maximum selectable for the DAQ card using four channels. Since the maximum for the DAQ Card is 210000 Samples/s, with four channels the maximum sample rate per channel is 52500 Samples/s (we choose a value a bit smaller)
- the number of samples per channel depend on the acquisition time. By fixing an acquisition time of 6 s, the needed number of samples per channel is:

  6s * 51200 Samples/s = 307200 samples (per channel)
- the Max - Min Voltage level per channel depends on the expected voltage measurements (to avoid overflow) and on the desired resolution. Since

measurements were quite different in the various points of the structure, we used different Max-Min V level values during the tests.

4.7.1.2 Experimental Results

A frequency analysis was performed on the experimental data using either LabView software (fig.75-76) or Matlab software. The most significant results are reported in pictures 78-79.

From the power spectral density (fig. 78-79), it is clear that the major vibration sources are at a frequency of about 200 Hz and come from the engine. A very small spectral component is present also at about 20 Hz and 80 Hz which are due to the main and tail rotor, but can be neglected if compared to the engine component.

By defining a g_{rms} value as [58,59]:

$$\text{g (rms)} = \sqrt{(std(x)^2 - mean(x)^2)} \tag{4.5}$$

where

x is the vector of the acquired data

n is the number of samples (the x row number)

std(x) is the x standard deviation defined as $std(x) = (\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^{1/2}$ (4.6)

mean (x) is the x average value over the n sample $mean(x) = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (4.7)

it is possible to check if the vibration load experienced by the electronic component, once they were damped, is within the operational vibration load advised by the factory, for the onboard avionics components (NAV 420 Operating Vibration Range: < 6 g rms 20Hz-2kHz ; CRIO Operating Vibration Range : < 5 g rms 10Hz-500Hz).

Experimental results confirmed that the elastomeric dampers efficiently attenuate vibration on the onboard avionics.

The high vibration load experienced by the landing gear (at the engine frequency of 200 Hz) may seem somehow surprising, above all because very poor literature (or better no literature at all) exists about that. Therefore, experimental tests were repeated several times (even with a different acquisition system) and were compared with the results provided by Boeing for a similar helicopter. This data were available at low engine rpm, but they seem to confirm the order of magnitude of the measured data.

Test "Eli09": Vibration level along the z axis

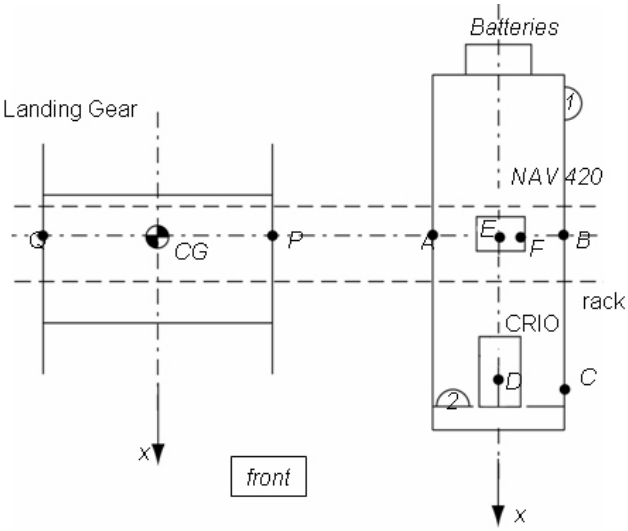


Figure 77: Accelerometers mounting points

As example acceleration experienced on the landing gear (position P in figure 77) and on the "emulated " NAV 420 (position E figure 77) will be reported.

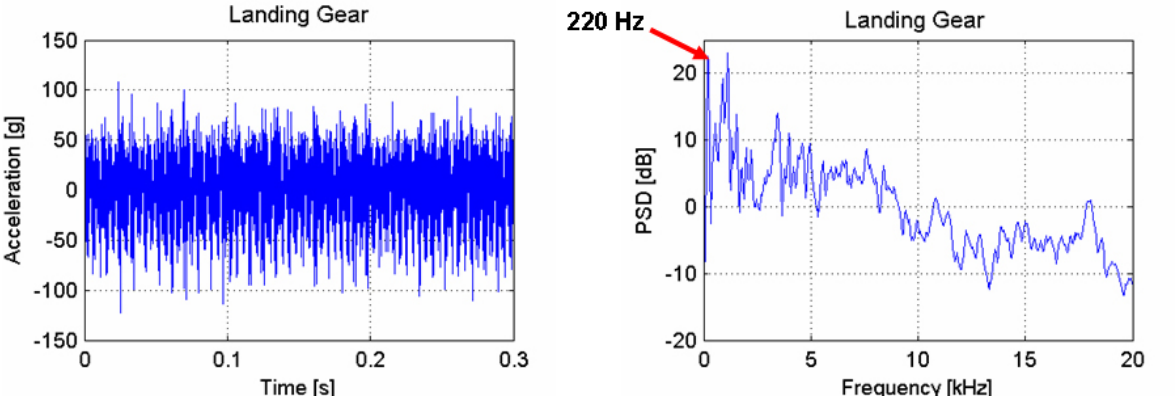


Figure 78: Acceleration experienced on landing gear

From the PSD [60,61] diagram it can be seen that the main vibration load is caused by the engine spinning at about 200 Hz. This is also confirmed by the fact that, filtering the signal in the band 0-1kHz the g_{rms} value, calculated with equation 4.5, is about 11.5. If the signal is filtered under 40 Hz, this value is reduce to 2.05. Using an high pass filter over 2kHz the rms value is also reduced to 2.22.

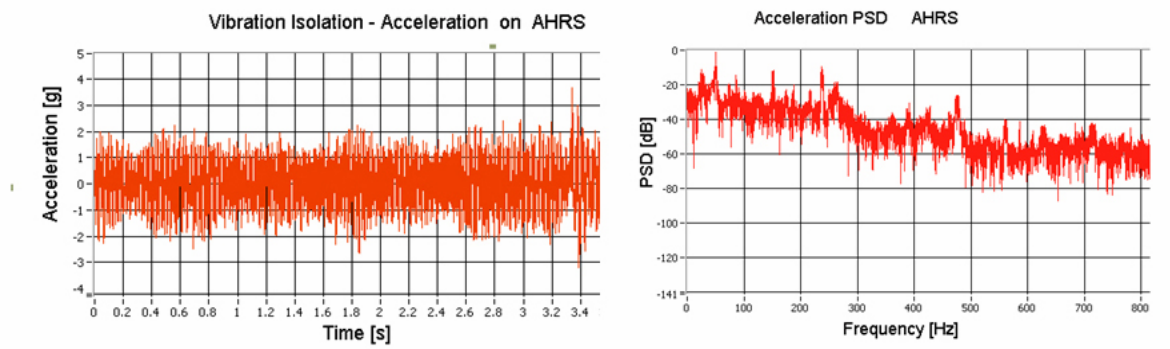


Figure 79: Acceleration Experienced on the isolated NAV 420

Acceleration are readily attenuated and the g_{rms} value calculated with equation (4.5) is reduced to 0.7281.

Figure 80 shows also the acceleration experienced at point B (see fig.77) after the first set of shock mounts; acceleration are attenuated, but using also the neoprene strips much better results were achieved (fig.79).

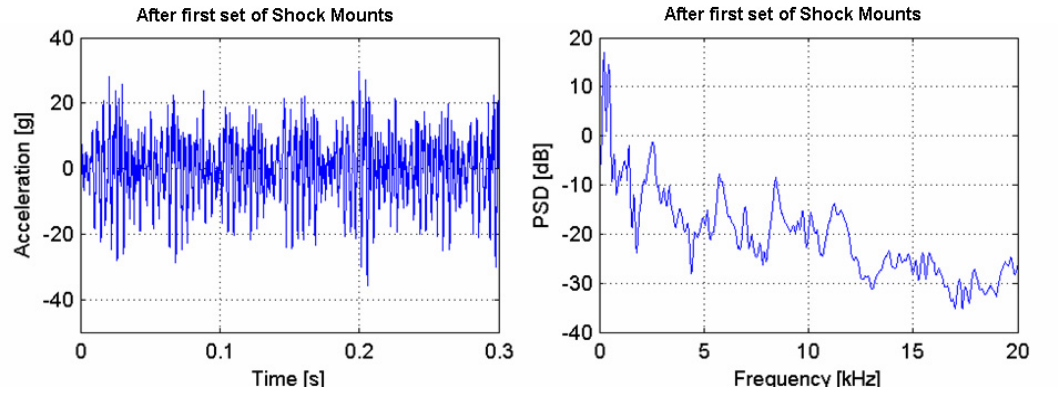


Figure 80: Acceleration experienced after the first shock mounts

For sake of comparison, Boeing results for Raptor 60 are reported in figure 81.

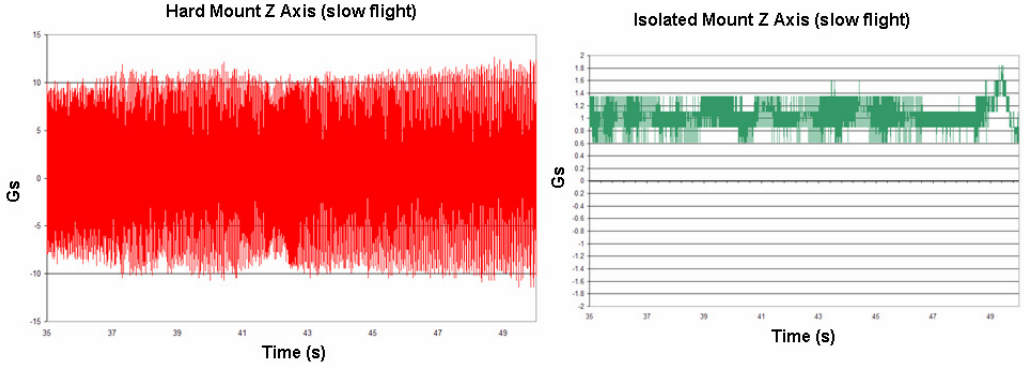


Figure 81: Boeing Results for Raptor 60 [62]

The hard mounted accelerometer showed accelerations of between 8 and 15 times the normal force of gravity, whereas, the isolated accelerometer only saw around a 0.3 to 0.8 increase in the force of gravity during the same flight test [62].

4. 8 HARDWARE AND SENSORS DAQ FLIGHT TESTS

The UNIBO RUAV avionics hardware was successfully tested in flight. Flight data were acquired by means of the data acquisition software described in the previous sections.

In order to conduct a flight tests, the vehicle avionics must be first powered using the avionics box power panel. Compiled flight code must be uploaded from the ground station onto the flight computer and started remotely. The ground control station, constituted by a simple laptop computer, connects to the air vehicle, displaying its status. When everything and everybody on the test team are set up for the flight, the engine is started allowing continuous flight for approximately 15 minutes, limited by the on-board fuel capacity (then helicopter re-fuelling must be made).

For the purpose of flight data acquisition tests, the helicopter was flying in RC mode, while onboard data logging was started and stopped by the ground control station operator.

High rate on-board data recording is independent from data communication and data display on the GCS. Data are recorded in a file on the volatile CRIO RAM. This file can be downloaded to the GCS using the WIFI data link during the flight tests, even without stopping the flight code, or after the flight.

During the flights, data were transferred from the air vehicle back to the Ground Control Station (GCS) via wireless data link and monitored by the GCS operator.

All onboard electronics worked properly while sensor data was recorded at 100 Hz. AHRS raw data (figure 82) show vibration disturbances.

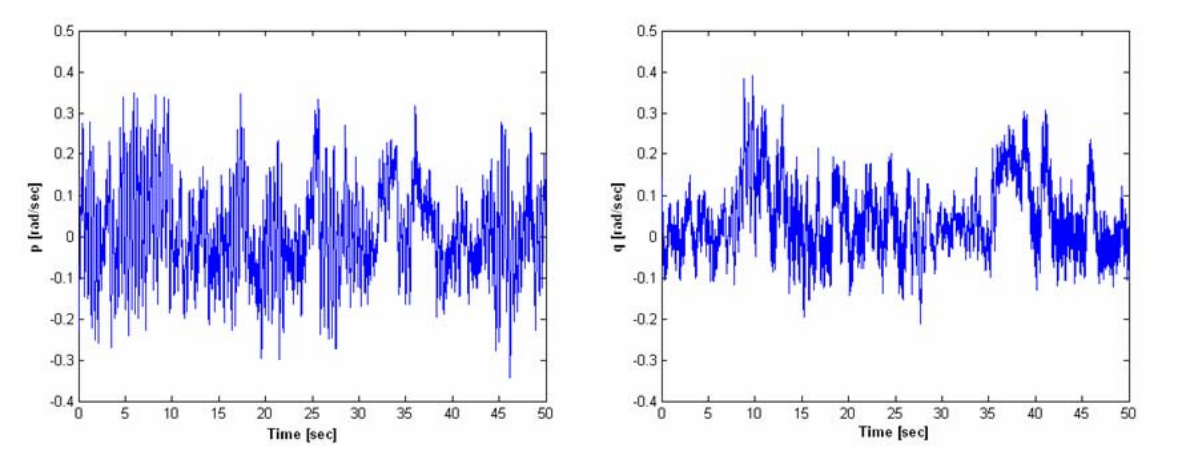


Figure 82: Example of pitch and roll rate AHRS raw data

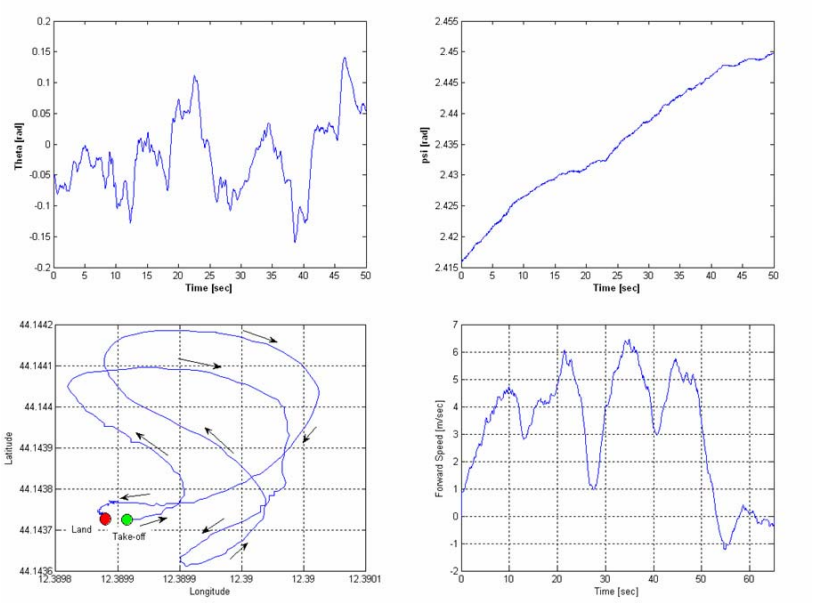


Figure 83: AHRS filtered flight data

However, thanks to the XBow NAV420 integrated Kalman filter, smooth and stable GPS position information, velocity and attitude measurements were available, which can be used for control and navigation system implementation. Figure 83 shows examples of sensor data measurements taken while the helicopter was overflying the test field at low speed conditions.

Ultrasonic sensors were also tested. Recorded flight tests showed good experimental results although they could provide reliable altitude measurements only up to 5.5-6 m (see fig.84).

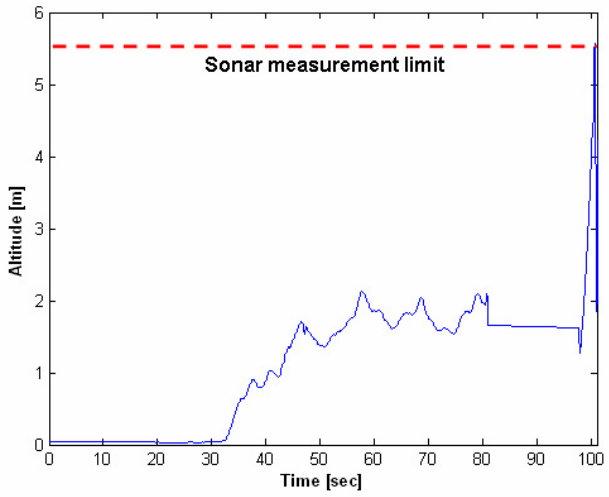


Figure 84: Sonar sensors measurements

4.8.1 FLIGHT DATA RECORD VIRTUAL RE-VIEW

Based on the work done in the CAPECON project for the ground control station, a "post-view station" was develop, in order to help data record analysis by reproducing the flight tests in a virtual scenery. The "post-view station" runs a LabView software which was derived from the one developed for the CAPECON mission simulation environment and is reported in the enclosed CD.

The program reads the data record file and displays flight information on a virtual cockpit (reproducing also a pilot virtual radio stick) and on helicopter states diagrams; meanwhile data are sent via TCP/IP to the visual system which is able to reproduce in real time an external view of the air vehicle.

The station architecture is based on two computer (see fig. 85): one is used for the visual system while the other one for the LabView code and user interface.

Such a system was very useful during post-processing analysis, since it's possible to have real time and immediate memory of the helicopter behaviour during experimental tests, thereby facilitating flight data record interpretation.

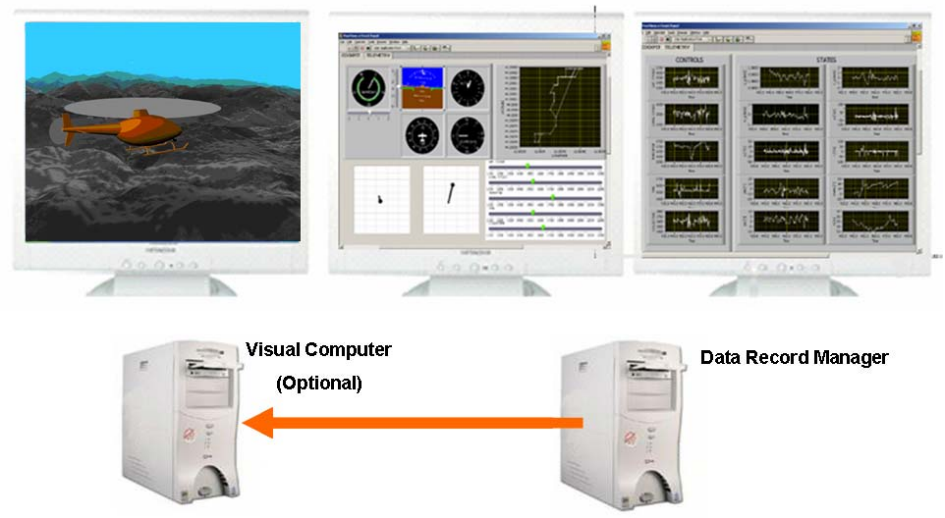


Figure 85: "Post-View" Station Architecture

Chapter 5

SITL SIMULATION

After the hardware was set-up, a series of flight tests were done in order to collect experimental data, for identifying the helicopter dynamics characteristics and develop a reliable vehicle simulation model. The helicopter dynamics was modelled in nearly hover flight conditions by Pretolani [19] using a transfer function approach [63-66]. Based on the helicopter dynamics transfer function identification, classical PID controllers were designed in the Matlab/Simulink enviroment, neglecting cross-coupling between the helicopter inputs. Results founded by Pretolani are summarized in section 5.1 and were used as starting point to implement the control system on the onboard computer.

The control system architecture used to control the helicopter is based on a nested PID approach as shown in figure 86.

The Vx and Vy track velocities control is implemented using the two level, nested loop structure shown in Figure 86. Lateral track velocity (Vy) errors are used to generate roll demands for the roll (ϕ) control module, while longitudinal track velocity (Vx) errors are used to generate pitch demands for the pitch (θ) control module. Integral contribution in the outer velocities loop compensate also external disturbances resulting from varying wind conditions. The inner attitude controllers generate servo rotation commands for the helicopter to maintain the desired reference condition.

The reference track velocities Vx and Vy can be generated either by an outer guidance and navigation control system or by a user pre-defined reference velocity profile.

The vertical velocity control uses a stand alone PI feedback control loop.

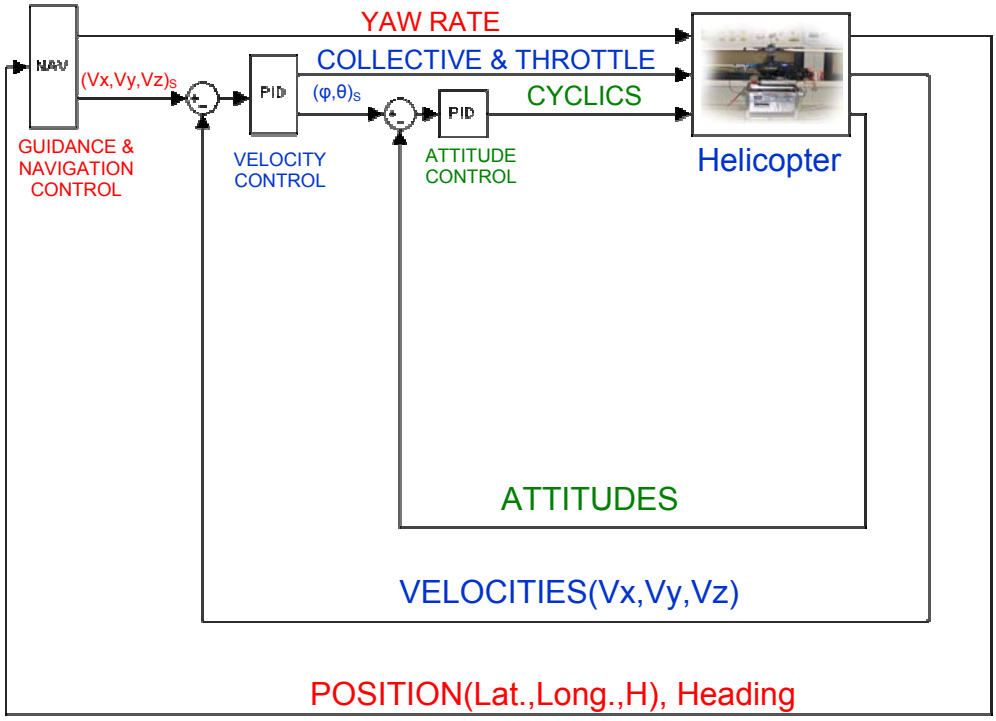


Figure 86: Onboard Control System Architecture

The vertical control either can take the form of vertical velocity regulation or height regulation: in the first case the vertical velocity profile to be maintained is given by the user, in the second case the reference vertical velocity is calculated by the altitude hold regulator in the guidance and navigation system.

The heading control is left to the HL-AVCS onboard gyro system. The HL-AVCS gyro input is actually a yaw rate, calculated on the basis of the helicopter heading error. Again, the helicopter heading error can be calculated either by the NGCS or can be user defined.

The control system architecture described above was implemented on the onboard computer. Currently, only the velocity control system was experimented in flight, while the navigation and guidance system must be still tested. Therefore, the reference velocities and heading profiles were defined by the operator at the ground control station, depending on the flight test to be performed.

The complete onboard software implementation is described in sections 5.2-5.3.

5.1 HELICOPTER DYNAMICS IDENTIFICATION AND SIMULINK PID DESIGN RESULTS

The identified helicopter dynamics transfer function are reported below [19]:

Attitude dynamics

$$H_{\vartheta} = \frac{1}{s}\frac{A_{long}}{\tau_e}\frac{\omega_{nq}^2}{s^2 + 1/\tau_e s + \omega_{nq}^2} \qquad H_{\varphi} = \frac{1}{s}\frac{B_{lat}}{\tau_e}\frac{\omega_{np}^2}{s^2 + 1/\tau_e s + \omega_{np}^2} \tag{5.1}$$

Identified Parameters				
A_{long} [rad/rad]	ω_q [rad/sec]	B_{lat} [rad/rad]	ω_q [rad/sec]	τ_e [sec]
0.30025	12.1	0.22078	18.1	0.132

Table 7: Attitude dynamics identified parameters

Velocity dynamics

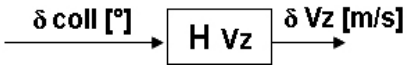
$$H_{Vx} = \frac{-g}{s - X_{Vx}} \qquad H_{Vy} = \frac{g}{s - Y_{Vy}} \tag{5.2}$$

Identified Parameters		
g [m/s^2]	X_{Vx} [1/s]	Y_{Vy} [1/s]
9.81	-0.39654	0.05

Table 8: Velocity dynamics identified parameters

Heave Dynamics

$$H_{Vz} = \frac{-Zcoll}{s + Z_{Vz}} \tag{5.3}$$



Identified Parameters	
$\mathbf{Z_{coll}}$ [(m/s^2)/rad]	$\boldsymbol{Z_{Vz}}$ [1/s]
-30	-1.14

Table 9: Heave Dynamics identified parameters

For Single Input Single Output (SISO) systems, the controller that is most commonly used in industrial process control is the PID controller. This controller has the following transfer function in the Laplace domain [67]:

$$G_c(s) = K_P + \frac{K_I}{s} + K_D \cdot s \tag{5.4}$$

where K_P , K_I and K_D the proportional, integral and derivative gains respectively.

Equation 5.4 is often rewritten in terms of time constants:

$$G_c(s) = K_c + \frac{K_c}{T_I \cdot s} + K_c \cdot T_D \cdot s \qquad \text{with} \quad K_C = K_P \tag{5.5}$$

This controller is termed a PID controller because Equation 5.4 has a proportional, integral and derivative term. Although these controllers are simple, they are quite robust, simple to tune and often provide sufficient control [68-72]. PID controllers have well known tuning methods such as the Ziegler-Nichols Step and Ultimate Gain Methods. Whilst these tuning methods are unlikely to produce an optimally tuned controller, they do provide a good starting point for further optimisation. Therefore, the PID gains were tuned in the Matlab/Simulink environment using the transfer function reported above. Even if decoupled PID loops were considered, this was enough to control the helicopter, since such loops should view the coupling between axes merely as a disturbance and should be able to compensate this effect in a robust manner.

Simulation results showed also that a PI controller (derivative term set to zero) will provide sufficient control capabilities [19]. Therefore, a simple PI was implemented on the onboard computer. The calculated PI gains are reported below, together with the final gain value, calibrated experimentally and currently used for the onboard control system.

Attitude PI Gains				
	$K_C\theta$ [°servo/°θ]	$K_I\theta$ [°servo/°θ * s]	$K_C\varphi$ [°servo/°φ]	$K_I\varphi$ [°servo/(°φ * s)]
Calculated	-0.77366	-0.08	1.0418	0.11346
Experimental	-1	-1	1	1

Table 30: Attitude Controllers PI Gains

Velocity PI Gains						
	K_CV_x [°θ/(m/s)]	K_IV_x [°θ /m]	K_CV_y [°φ/ m/s]	K_IV_y [°φ/ m]	K_CVz [°coll /m/s]	K_IVz [°coll /m]
Calculated	-13.2	-4.03	11.43	3.55	-3.622	-4.96
Experimental	-10	-1	10	1	-10	-10

Table 41: Velocity controllers PI Gains

5.2 ONBOARD CONTROL SYSTEM

For the implementation of the onboard velocity – attitude control system a lot of things must be taken into account:

- the control loop is not in a continuous time domain but cycles at a discrete time interval. Therefore a discretise controller must be implemented
- the FPGA environment allows programming only by using integer values and the sensor data output are all I16 values with their own scaled data. Therefore, controller output and input values must be adjusted with some scale factors in order to provide the correct servo commands value in PWM microseconds high-time.

5.2.1 DISCRETE PID IMPLEMENTATION

Consider the ideal PID controller written in the continuous time domain form [67]:

$$u(t) = K_c e(t) + \frac{K_c}{T_i}\int_0^t e(t)dt + K_c T_d \frac{de(t)}{dt} + u_0 \tag{5.6}$$

where e(t) is the process variable error defined as:

$e(t) = SP - PV$ (SP being the Set Point and PV the Process Variable) (5.7)

and u(t) is the PID output.

To discretise the controller, we need to approximate the integral and the derivative terms to forms suitable for computation by a computer. From a purely numerical point of view, if Ts is the loop cycle time, we can use:

$$\frac{de(t)}{dt} \approx \frac{e(t)-e(t-1)}{T_s} \qquad \int_0^t e(t)dt \approx T_s \sum_0^t e(i) \tag{5.8}$$

The general discrete PID algorithm can be therefore [73]:

$$u(t) = K_c e(t) + \frac{K_c T_s}{T_i} \sum_{i=0}^{t} e(i) + \frac{K_c T_d (e(t)-e(t-1))}{T_s} + u_0 \tag{5.9}$$

which is now in the form of a difference equation, suitable for coding in an appropriate programming language. This particular form of the PID algorithm is known as the **'positional'** PID controller, because the control signal is calculated with reference to a base level, u_o (which can be known experimentally and must be set up correctly inside the algorithm).

Actually, the PID integral action is calculated by using a trapezoid integration to avoid sharp changes in integral action, when there is a sudden change in *PV* or *SP*. The integral contribution is therefore express as:

$$u_I(t) = \frac{K_c T_s}{T_i} \sum_{i=0}^{t} \frac{e(i)+e(i-1)}{2} \tag{5.10}$$

As for the derivative contribution, the derivative action is applied only to the PV in order to avoid effects due to abrupt changes in SP. Therefore the following formula represents the Derivative Action:

$$u_D(t) = -\frac{K_C T_d}{T_s}(PV(t) - PV(t-1)) \tag{5.11}$$

So, finally it is possible to implement the following formula for the discrete PID controller:

$$u(t) = K_c e(t) + \frac{K_c T_s}{T_i} \sum_{i=0}^{t} \frac{e(i) + e(i-1)}{2} + \frac{K_c T_d (PV(t) - PV(t-1))}{T_s} + u_0 \qquad (5.12)$$

Another important aspect is that the use of a summation to calculate the contribution of the integral term can lead to problems causing long periods of overshoots in the controlled response. This phenomenon is known as integral windup. The algorithm implemented on board provides code for integrator anti- wind up.

5.2.2 ONBOARD NESTED PI SOFTWARE

The Complete nested PI software is shown in figure 87.

The control loop runs at 50 Hz taking into account the bandwidth of the helicopter servo actuators.

The loop performs a series of instructions in three subsequent frames:

-in the first frame the control loop rate is set

-the second frame is used to read all the input parameters necessary to the controller

-the third frame contains the PID implementation and calculation of PWM high time to be sent to the PWM generation loop (see section 4.5.2).

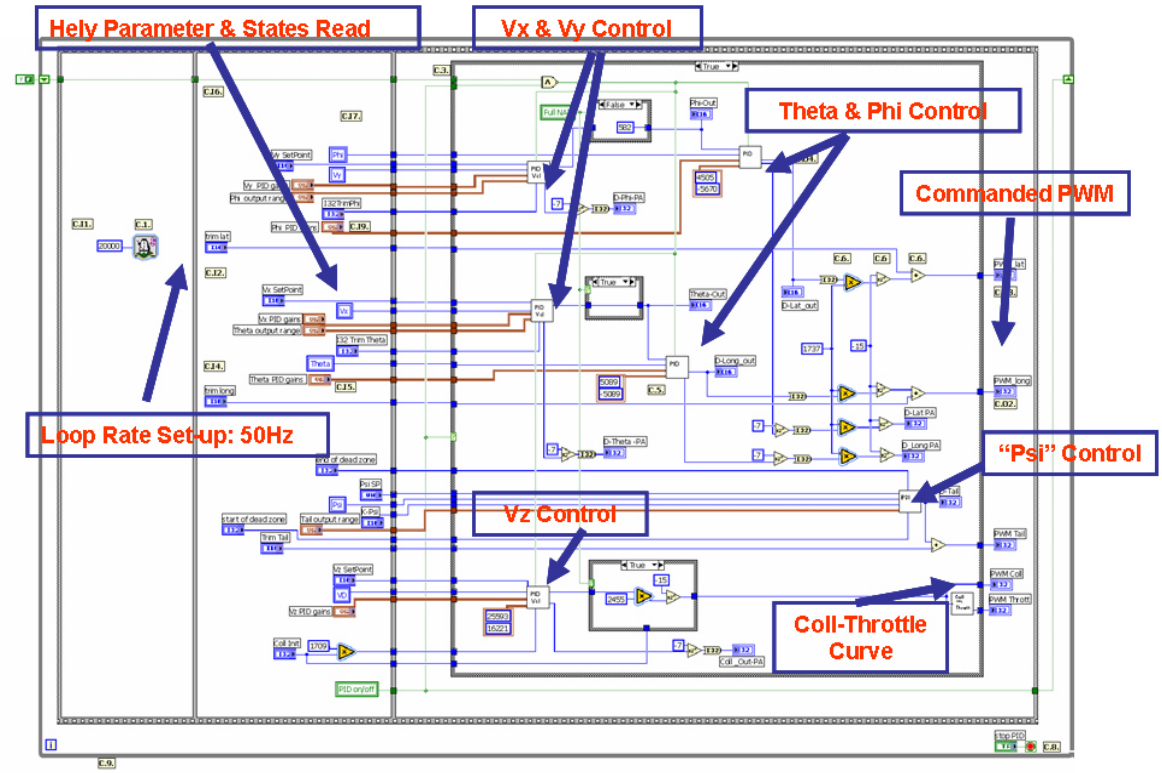


Figure 87: Onboard Control Loop

The controller structure is the one shown in figure 86. In order to understand the PID FPGA implementation, it is necessary to define the following unity of measure and scale factors:

Name	Symbol	Scale factor
AHRS Euler Angle Output	***[aI16]***	***[°]=[aI16]*180*2^{-15}***
AHRS NED Velocity Output	***[vI16]***	***[m/s]=[vI16]*256*2^{-15}***
AHRS Latitude, Longitude Output	***[LI32]***	***[°]=[LI32]*180*2^{-31}***
PWM High time [µs]	**[µs]**	**[° servo] = [µs]*104.3*10^{-3}**
PID Attitude servo angle out (I16)	***[sI16]***	***[°servo]=[sI16]*180*2^{-15}*** **And** **[µs] = *[sI16]*180*2^{-15}* *10^3/104.3 = *[sI16]*1737*2^{-15}***
PID Vz output	**[VzO]=[°*vI16/(m/s)]**	**[µs] = [VzO]*256*2^{-15}* *10^3/104.3 = [VzO]*2455*2^{-15}*** **And** **[VzO]= [µs] *13.3504***
Initial collective [I32]	**[cI32]**	**[cI32]= [VzO]*2^7=[µs]*1709**
K_Vx and K_Vy Gains	**[°Attitude/(m/s)]**	**[°Attitude/m/s]=[aI16]/([vI16]*1.42)**

Table 52: Unity of Measures and scale factors used in the control code

Each PID implementation will be described in the next sub-sections.

Vx-theta PI

A schematic of the Vx-Theta nested PI software is reported in figure 88.

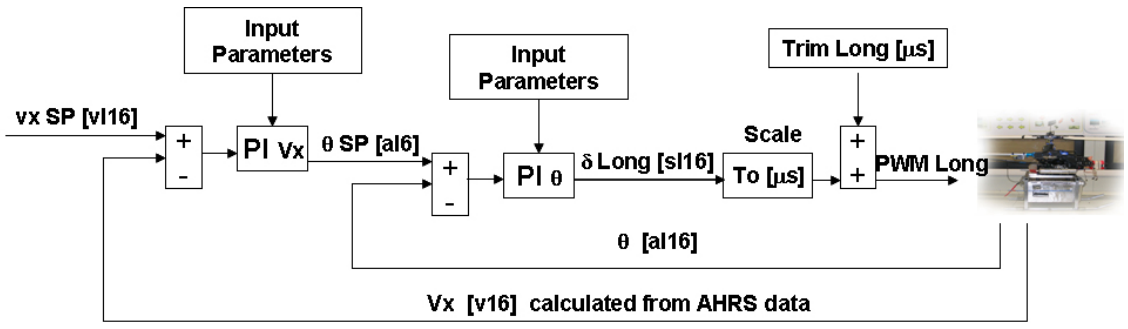


Figure 88: Schematic of the FPGA Vx-Theta nested PI

The first PI implement the forward velocity control along the trajectory. This PI calculate the reference theta attitude to be passed to the theta attitude controller. The attitude PI calculates the servo rotation angle variation to be added to the trim value in order to maintain the desired set point.

This commands are then scaled in microseconds of PWM high time which is used to generate the PWM signal for the servo actuator (see PWM generation algorithm section 4.5.2).

Unity of measure must be scaled as shown in figure taking into account the scale factors defined in table 12.

Tables 13-16 summarize the input and output parameters needed for the algorithm to work properly.

Vx PI Input Parameters	
Vx SP [vI16]	User defined profile or from Navigation System
Current Vx [vI16]	Calculated from AHRS data
$K_C Vx * 2^8$ [aI16/*vI16*]	-2560
$K_I Vx * T_s * 2^8$ [aI16/*vI16*]	-5
Output High [aI16]	3640
Output Low [aI16]	-3640
Initial theta [aI16*2^7]	0
PI Reset	TRUE first PI call otherwise FALSE (performed automatically by the program)

Table 63: Vx PI Input Parameters

Vx PI Outputs
Proportional action theta [aI16*2^7]
Total theta action [aI16]

Table 74: Vx PI Outputs

theta PI Input Parameters	
theta SP [aI16]	User defined profile or from Navigation System
Current Theta [aI16]	from AHRS data
$K_C \theta * 2^8$ [°servo/°θ]	-256
$K_I \theta * T_s * 2^8$ [°servo/°θ]	-5
Output High [sI16]	5089
Output Low [sI16]	-5089
PI Reset	TRUE first PI call otherwise FALSE (performed automatically by the program)

Table 85: theta PI Input Parameters

theta PI Outputs
Proportional action Long cyclic with repect to trim [sI16*2^7]
Total Longitudinal cyclic action with respect to trim[sI16]
N.B.Long Trim value to be added after PI ouput scale: 1544 μs

Table 96: theta PI Outputs

Vy-phi PI

A schematic of the Vy-phi nested PI software is reported in figure 89.

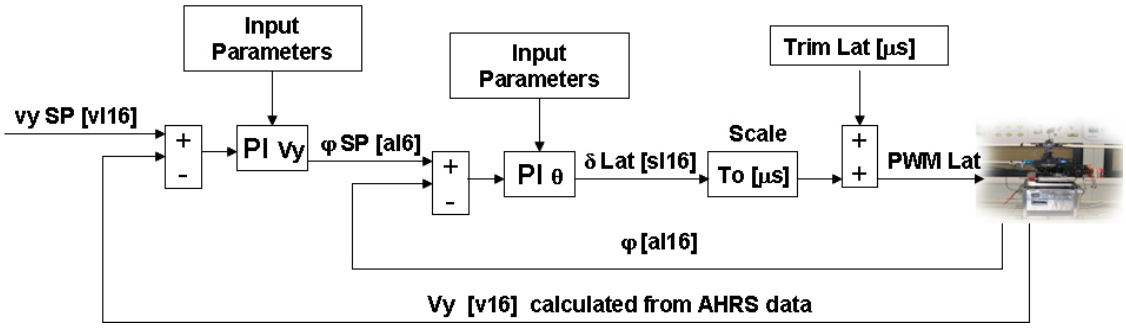


Figure 89: Schematic of the Vy-phi nested PI

The first PI implement the lateral velocity control along the trajectory. This PI calculate the reference phi attitude to be passed to the roll attitude controller. The attitude PI calculates the servo rotation angle variation, to be added to the trim value, in order to maintain the desired set point.

This commands are then scaled in microseconds of PWM high time which is used to generate the PWM signal for the servo actuator (see PWM generation algorithm section 4.5.2).

Unity of measure must be scaled as shown in figure taking into account the scale factor defined in table 12.

Tables 17-20 summarize the input and output parameters needed for the algorithm to work properly.

Vy PI Input Parameters	
Vy SP [vI16]	User defined profile or from Navigation System
Current Vy [vI16]	Calculated from AHRS data
$K_C Vy * 2^8$ [°aI16/*vI16s*]	2560
$K_I Vy * T_s * 2^8$ [°*aI16 /vI16*]	5
Output High [aI16]	3640
Output Low [aI16]	-3640
Initial phi [aI16*2^7]	74565
PI Reset	TRUE first PI call otherwise FALSE (performed automatically by the program)

Table 107: Vy PI Input Parameters

Vx PI Outputs
Proportional action phi [aI16*2^7]
Total phi action [aI16]

Table 118: Vx PI Outputs

phi PI Input Parameters	
phi SP [aI16]	User defined profile or from Navigation System
Current phi [aI16]	from AHRS data
$K_C\varphi^* 2^8$ [°servo/°θ]	256
$K_I\varphi^* T_s^* 2^8$ [°servo/°θ]	5
Output High [sI16]	4505
Output Low [sI16]	-5670
PI Reset	TRUE first PI call otherwise FALSE (performed automatically by the program)

Table19: phi PI Input Parameters

phi PI Outputs
Proportional action Lateral cyclic with repect to trim [sI16*2^7]
Total Lateral cyclic action with respect to trim[sI16]
N.B.Lat Trim value to be added after PI ouput scale: 1564 µs

Table 120: phi PI Outputs

Vz PI

A schematic of the stand alone Vz PI software is reported in figure 90.

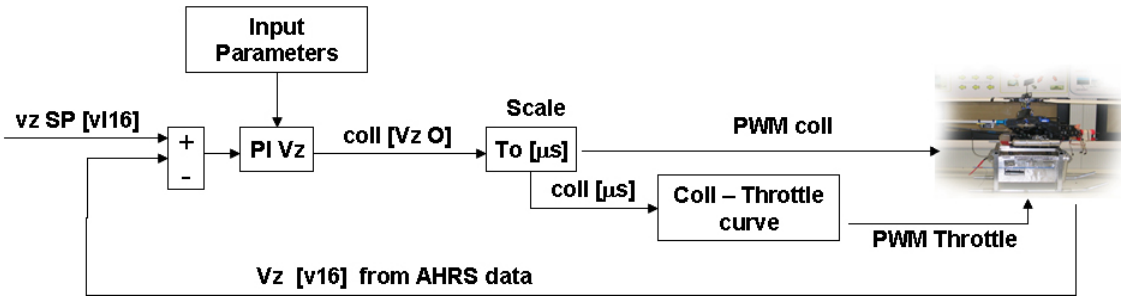


Figure 90: Schematic of the stand alone Vz PI

The PI implement the vertical velocity control along the trajectory. This PI calculates the collective servo rotation angle in order to maintain the desired set point. The trim condition (servo rotation corresponding to hover condition) must not be added, since it was taken into account in the PI integrator initialization.

This commands are then scaled in microseconds of PWM high time, which is used to generate the PWM signal for the servo actuator (see PWM generation algorithm section

4.5.2). The collective PWM high time is used also to find the corresponding PWM throttle high time, in order to send commands to the throttle servo actuator. At this aim, a calibration curve was derived from the one defined inside the radio settings, which was implemented on the FPGA by means of a look up table. The PWM collective-throttle curve is defined so that the rotor maintain constant rpm. The values used for the collective- throttle look up table are reported in table 23.

Unity of measure must be scaled, taking into account the scale factor defined in table 13.

Tables 21-22 summarize the input and output parameters needed for the algorithm to work properly.

Vz PI Input Parameters	
Vz SP [vI16]	User defined profile or from Navigation System
Current Vz [vI16]	AHRS
$K_C Vz * 2^8$ [°*coll /m/s*]	-2560
$K_I Vz * T_s * 2^8$ [°*coll / (m/s)*]	-51
Output High [Vz O]	25593
Output Low [Vz O]	16221
Initial Collective [cI32]	Coll init [μs]*1790 (coll init not less than 1420[μs])
PI Reset	TRUE first PI call otherwise FALSE (performed automatically by the program)

Table 131: Vz PI Input Parameters

Vz PI Outputs
Proportional action collective [cI32]
Total Collective action [Vz O]

Table 142: Vz PI Outputs

Collective [µs]	Throttle [µs]	Collective [µs]	Throttle [µs]
1218	1893	1578	1414
1238	1857	1598	1404
1258	1817	1618	1392
1278	1771	1638	1378
1298	1721	1658	1369
1318	1669	1678	1356
1338	1634	1698	1343
1358	1598	1718	1328
1378	1583	1738	1309
1398	1564	1758	1292
1418	1549	1778	1269
1438	1529	1798	1249
1458	1510	1818	1224
1478	1494	1838	1203
1498	1474	1858	1178
1518	1457	1878	1151
1538	1444	1898	1130
1558	1433		

Table 153: Collective-Throttle Curve Look-up table

Heading

A schematic of the heading control software is reported in figure 91.

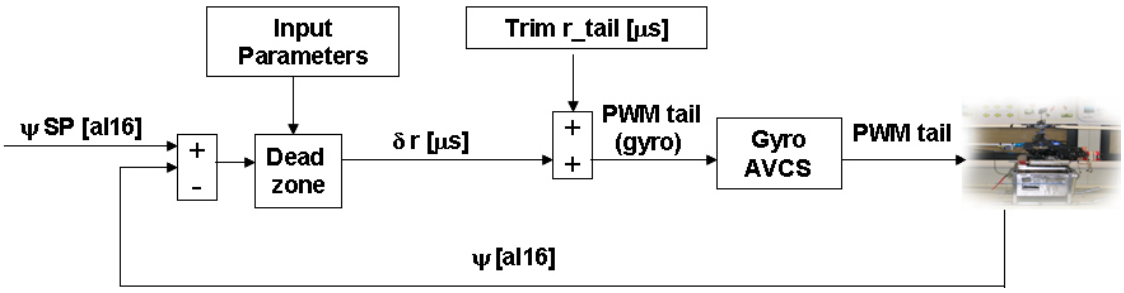


Figure 91: schematic of the heading control

Heading control is achieved using the onboard GYRO HL-AVCS system. Therefore, an algorithm is implemented which gives only a reference yaw rate to the gyro HL-AVCS based on the heading error, calculated with respect to the reference heading set point. The

algorithm is able to discriminate the sense of rotation, so that the helicopter will rotate always in the shorter direction to reach the set point.

The yaw rate, calculated from the dead zone block in figure 89, is intended to be a variation with respect to the condition of zero yaw rate. Therefore, this value must be added to the initialization trim value. The obtained command is the PWM high time in microseconds, which is used to generate the PWM signal for the gyro AVCS (see PWM generation algorithm section 4.5.2).

Unity of measure must be scaled as shown in figure 90 taking into account the scale factor defined in table 12.

Tables 24-25 summarize the input and output parameters needed for the algorithm to work properly.

psi dead zone Input Parameters	
psi SP [aI16]	User defined profile or from Navigation System
Current psi [aI16]	from AHRS data
Dead zone high limit [aI16]	546
Dead zone low limit [aI16]	-546
Output High [μs]	20
Output Low [μs]	-20
K_Psi saturation coefficient	21

Table 164: psi dead zone Input Parameters

psi dead zone Outputs
Yaw rate variation with respect to trim [μs]
N.B. Trim value to be added after ouput: 1515 μs

Table 175: psi dead zone Outputs

5.3 COMPLETE ONBOARD SOFTWARE IMPLEMENTATION

The complete RUAV onboard software architecture follows the typical CRIO advised programming technique, explained in chapter 4, and is illustrated in figure below. The source code of the full RUAV software is completely reported in the enclosed CD.

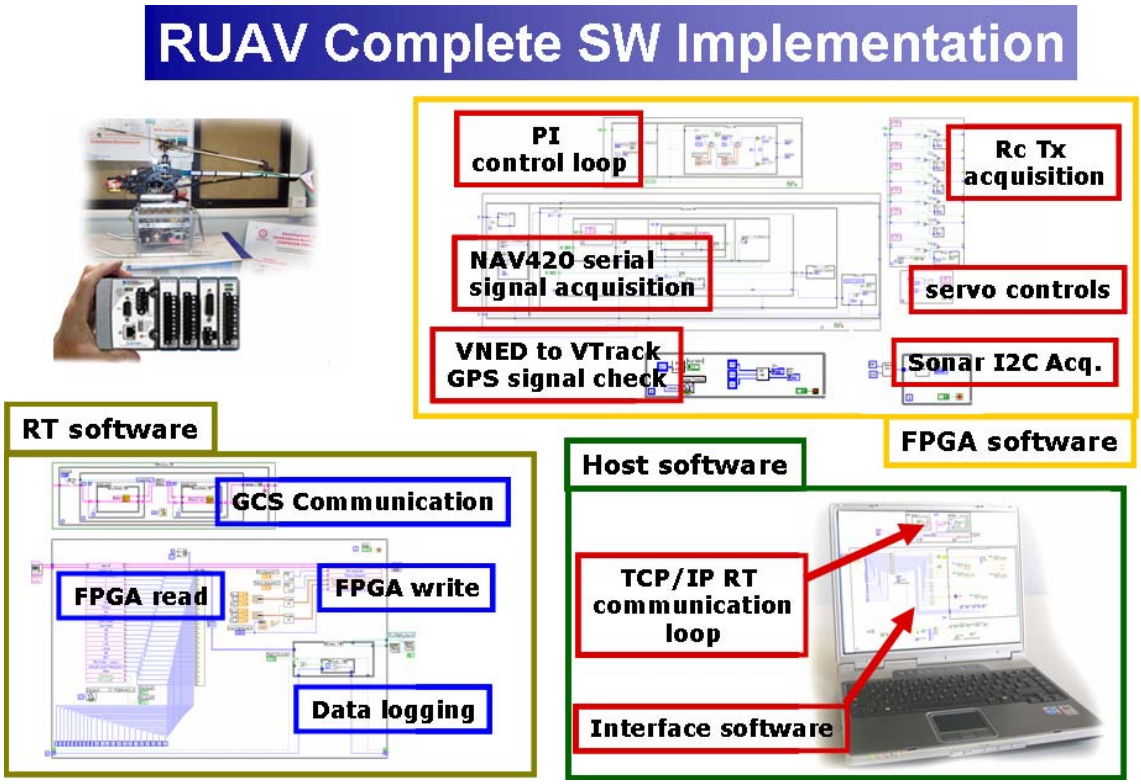


Figure 92: RUAV Complete Software Implementation

The complete RUAV software is divided in three main parts:

-**The FPGA software** is constituted by six independent loops in order to increase determinism. It includes:

- nested PI control loop as described in the previous sections
- NAV 420 data acquisition loop as described in section 4.4.1.
- radio PWM signal acquisition loop as described in section 4.5
- sonar sensor data acquisition loop as described in section 4.4.2
- PWM signal generation loop as described in section 4.5
- a V track calculation loop which transform the NED velocity coming from the NAV 420 into velocity along the trajectory, which will be used by the controller [74]. The same loop perform also a GPS signal check. If the GPS has poor signal, the NAV 420 velocity data are not reliable any more; therefore the velocity control is automatically disabled and the controller become a merely attitude control.

-**The RT (Real Time) software** is constituted by two independent loops:

- the time critical loop (timed at 10 ms), which perform high rate read/write communication with the FPGA software. The time critical loop acquires sensor

data from the FPGA which are either recorded on the CRIO volatile memory for post processing or communicated to the ground control station

-the normal priority loop which perform TCP/IP communication with the ground control station and is timed at 100 ms

-**The Host Software** is constituted by two independent loops:

-the communication loop which perform communication with the onboard computer for flight data transmission. Communication between the ground station and the onboard computer is bidirectional since the operator at the ground can interact with the onboard software by changing flight parameter values (for example the PI gains can be changed during flight test for controller final tuning).

-the user interface loop which contains the code to generate the user interface for flight test control and monitoring. The user interface code is made completely independent from all the other code so that different type of Graphical User Interface can be used, without need to change any other part of the source code. Depending on the flight test to be performed, different GUIs were developed: figures 93, 94, 97 show the ones used during PI tuning tests and flight data acquisition tests.

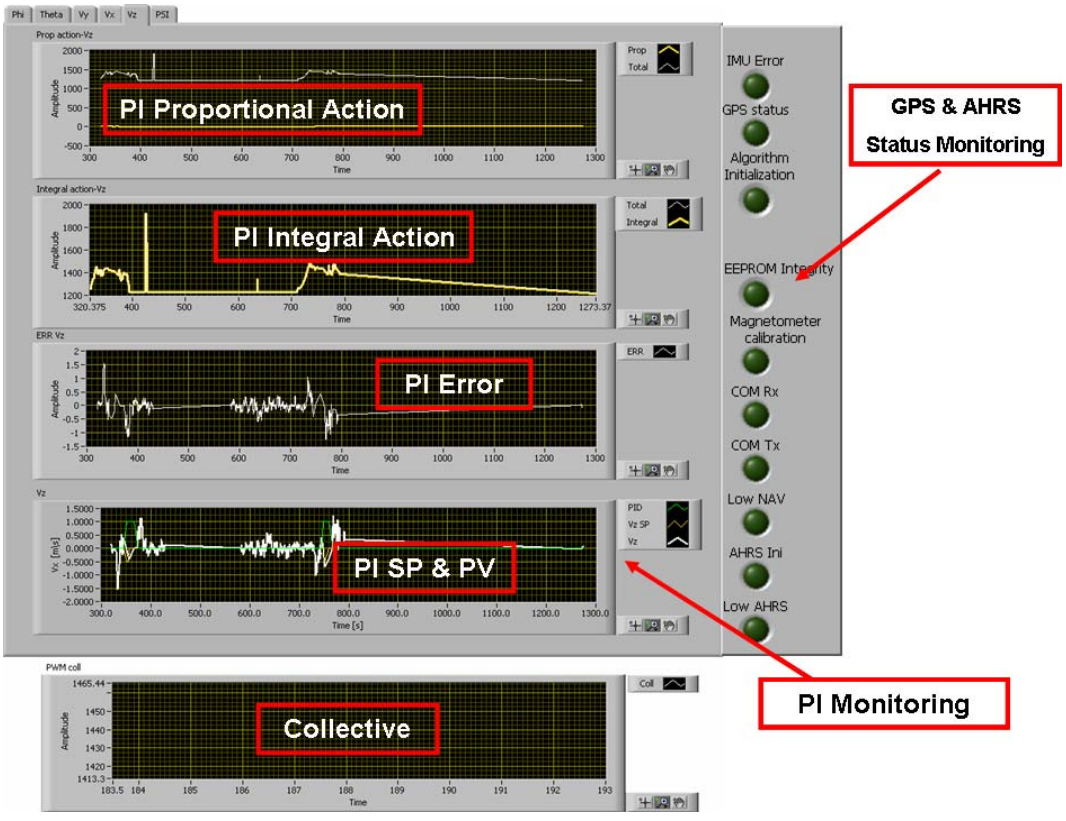


Figure 93: PI tuning tests GUI (1)

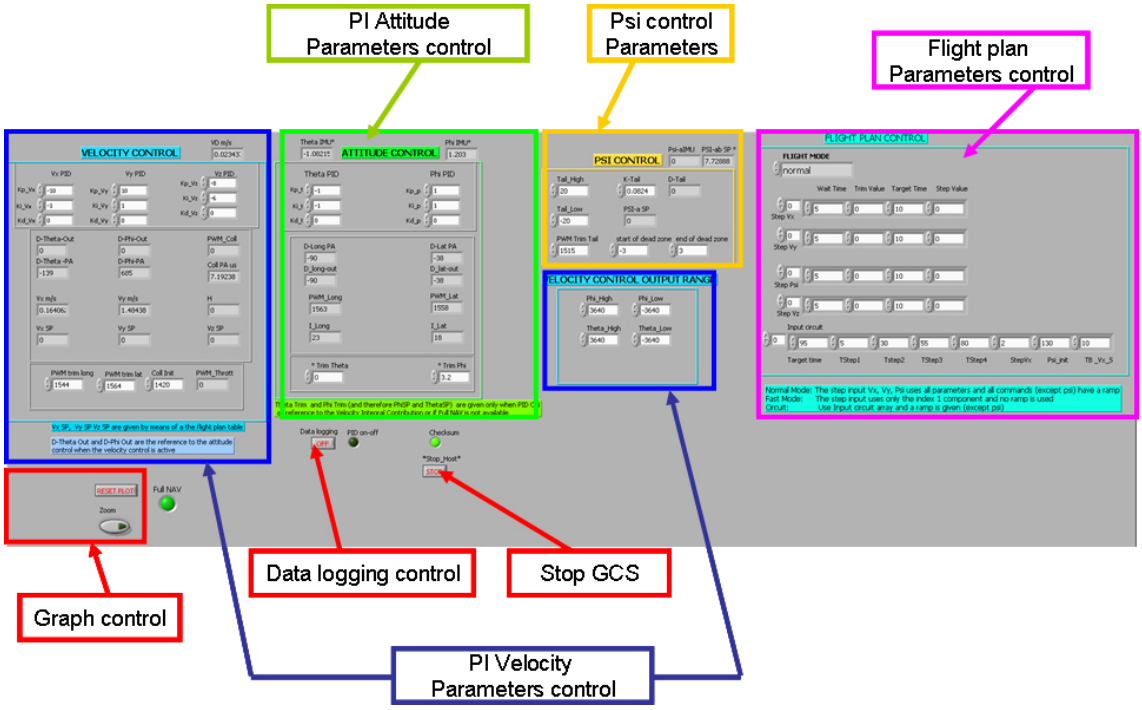


Figure 94: PI tuning tests GUI (2)

During PI tuning tests, the graphical interface was used to monitor the helicopter PI responses by means of dedicated diagrams, to enter a given flight plan (or a set point time history) and to change the PI gains if necessary. Six windows of diagrams are available, one for each variable under control. Moreover, the user can start or stop the onboard data logging by means of the front panel button shown in figure 94. The GUI interface can be stopped independently from the onboard software so that, if TCP/IP communication is lost, this will not affect the onboard program and the flight test can be concluded without any data loss. Three different type of automatic flight mode are available:

- "Normal mode": the user sets a velocity or heading profile to be maintained by the helicopter. The generated profiles are shown in figure 94 and can be defined by the user setting the wait time, the target time, the trim set point and the desired step. All set points (a part form heading) are given by means of a ramp to avoid sharp transitions. This setpoint mode is very safe because, if communication between the GCS and the onboard computer is lost, the helicopter will safely complete the task and flight tests have not to be abrupted.

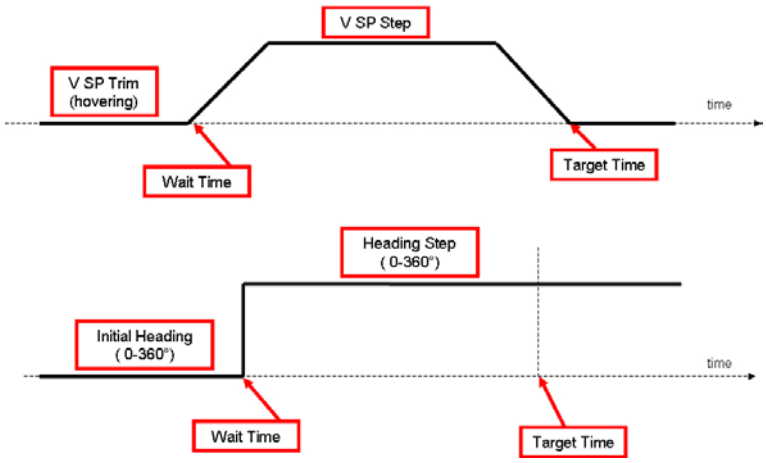


Figure 95: Typical Setpoint Profiles

- " Fast Mode": the user sets a velocity or heading step profile only by giving the step value and no ramp will be used; the reference signal stops when the user drives back the set point to trim condition. This mode is more risky with respect to the first one since, if communication between the GCS and the onboard computer is lost, the helicopter will not go back to the trim condition after a short period of time. The only way to recover from this situation is switching back to radio controlled mode. However, this mode was used only for preliminary and fast tests.
- "Flight Pattern Mode": the user can set a complete squared flight pattern to be track by the helicopter as shown in figure 96.

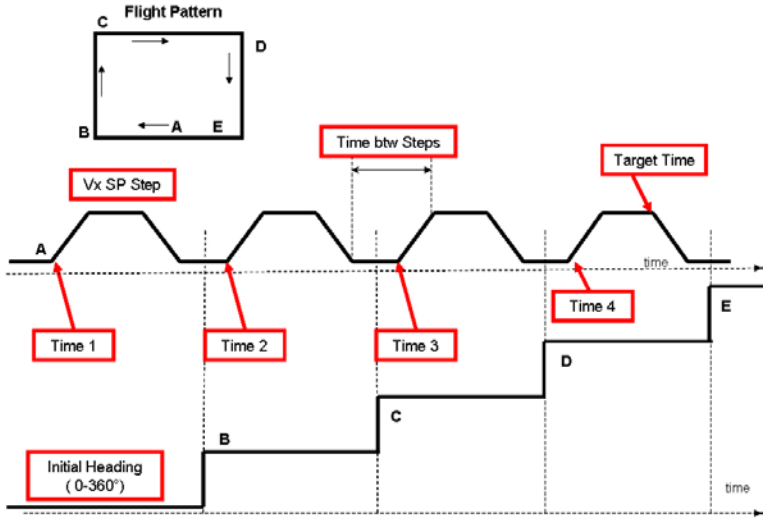


Figure 96: Typical Flight Pattern Profile

Inputs, defined by the user, are: the helicopter initial heading, the Vx step reference point, the target time, the time of each velocity step and the time between each velocity step (this time is used to rotate the helicopter in heading while the Vx

velocity has been driven to zero). When the helicopter is running in this mode, Vy and Vz are automatically set to zero.

The second interface available is much simpler and was used for flight data acquisition tests. It is composed by two windows:

- the first one reproduces a virtual cockpit together with the radio stick movement and a GPS 2D flight path
- the second one displays the helicopter states and the commanded inputs (see figure. 97)

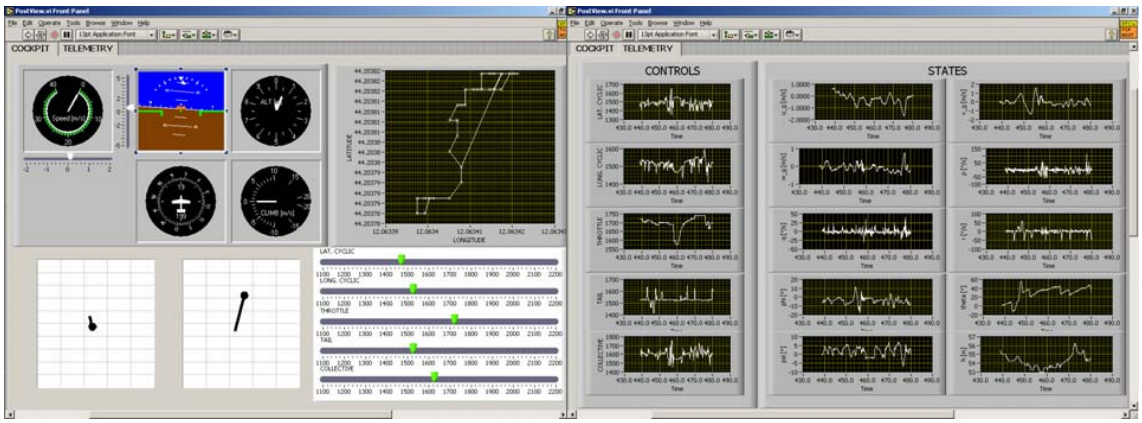


Figure 97: Flight Data Acquisition GUI

Chapter 6

HIL SIMULATION

To allow safe, risk-free onboard software testing, the PI controller was first implemented in a HIL simulator, shown in figure 98.

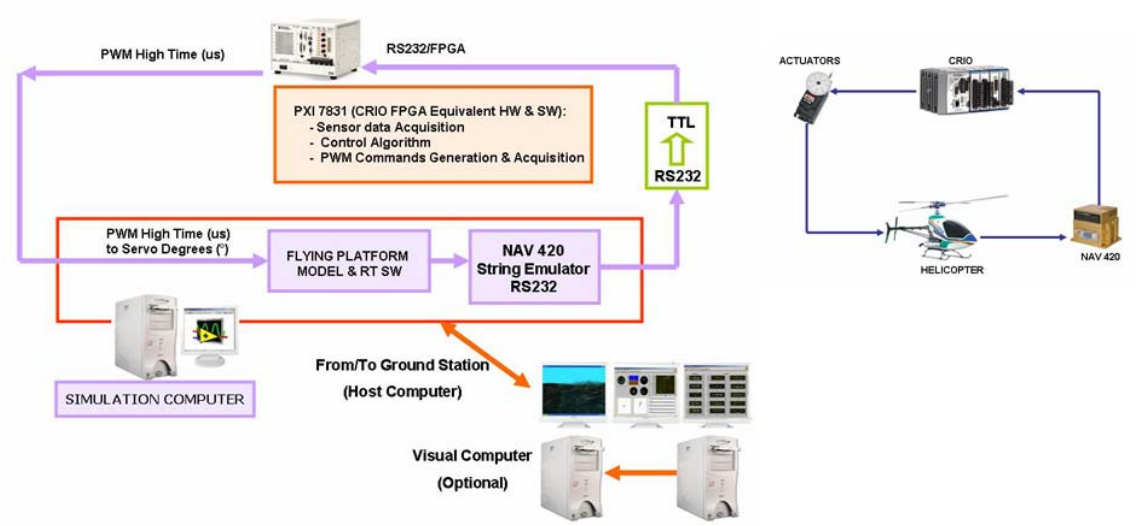


Figure 98: Schematic of HIL Simulator

The HIL test bench includes as much of the flight hardware in the testing loop as possible and is constituted by:

- an exact duplicate of the flight computer (the CRIO System) and of the onboard software including the PI controllers. At this aim, a National Instrument PXI 7831 was used which is equivalent to the CRIO FPGA modules. FPGA PXI communication with the computer (emulating the CRIO Real time core) can be performed by means of a FPGA interface card.
- a computer which simulates the helicopter plant, through the identified transfer functions reported in chapter 5, and the onboard sensor outputs. The helicopter simulation model receives inputs from the PI controllers: the software implemented

inside the FPGA PXI 7831 is able both to generate PWM electrical signal for the actuators and to acquire them by reading the PWM high time, which is sent to the helicopter simulator. The PWM high time in microsecond is then translated into degrees of servo rotation (see section 4.5), which is the actual input accepted by the identified transfer function. In this simplified model, states outputs are θ, *Vn*, φ, *Ve*, *Vd*, ψ, *p*, *q*, *r* which are formatted into a NAV 420 string emulator and are sent to the PXI by means of the computer serial port. The NAV 420 FPGA acquisition software, running inside the PXI, will acquire the helicopter states, such closing the loop. An hardware interface card was also used for converting computer RS232 output level to TTL level acceptable by the PXI. For practicality reasons, the helicopter dynamics transfer functions has been developed in the LabView environment; moreover, the helicopter simulator and the real time code runs on the same machine. This is possible because all the source code is organized using independently loops.

- a GCS computer which contains the host source code and communicate with the simulation computer by means of a TCP/IP protocol
- an OpenGL visual system computer can be optionally added for rendering the helicopter as it moves around in a virtual scenery. The visual system can receive input from the GCS computer using a TCP/IP protocol

The result of this setup is that the on-board computer effectively "thinks" it is flying the vehicle, as all of its configuration/data flow is identical to an autonomous flight setup.

In this scenario, performance and possible errors of the onboard software can be detected during intensive ground safe simulations, before performing any flight test. Figure below shows an example of HIL simulations results.

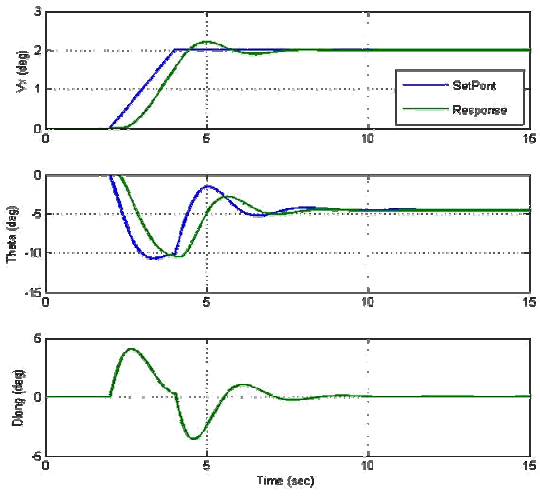


Figure 99: **Recorded HIL Simulation**

Chapter 7

FLIGHT TESTS & PI GAINS TUNING

The onboard control software was tested in flight. The complete flight campaign was done following five major subsequent steps as show in figure 100.

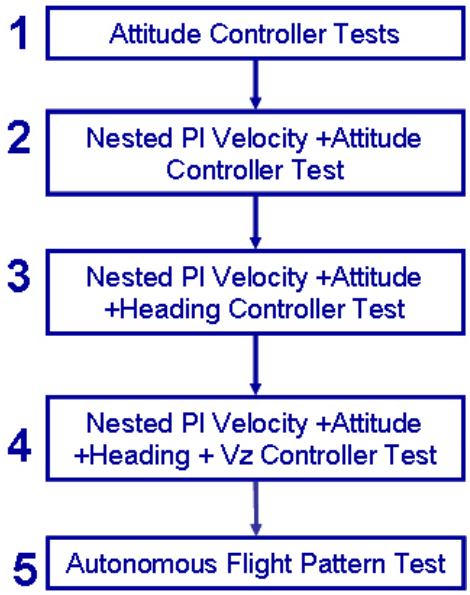


Figure 100: Flight Tests Procedures

- First only the attitude (φ and θ) PI controllers were tested. During these tests collective and tail commands were left to the RC pilot for safety reasons. As shown in table 11, the final proportional PI gains find by simulation results were almost correct while the integral gains were increased of an order of magnitude. This may be due to effects unmodelled by the transfer funtions . By using an higher integral gain these effects can be attenuated.

- Once the attitude controllers were somehow calibrated, the nested PI Velocity – attitude controllers were tested. During these tests, collective and tail commands were still left to the RC pilot for safety reasons. As shown in table 10-11, the final gains were much closer to the one found by simulations.
- The third step was to test the heading control together with the nested PI velocity controller. During these flight tests only collective was left to the RC pilot for safety reasons. The value to be calibrated during these flights was the yaw rate output in microseconds to be sent to the gyro HL-AVCS system. Starting with a very small value equal to ± 5μs, the same value was increased till finding an adequate yaw rate for the helicopter. The calibrated final value was ± 20μs corresponding to about 10 °/s. This value was kept intentionally low for safety reason but can be increased (or varied) if necessary.
- In the fourth step the full, PI controller was tested including the vertical velocity control. During these tests no commands was left to the pilot and the helicopter was flying completely autonomously. As shown in the previous tables, the final calibrated PI gains were higher with respect to the one calculated by simulations. This was due to the fact that, during simulations, the gains were kept intentionally low for the helicopter to maintain a very low rate of climb/descent. Vertical velocity flight tests can be very dangerous since, if the collective-throttle curve is not good calibrated or the PI gains are to high, the helicopter can crash to the ground without any hope to recover it. Therefore, the helicopter team decided to keep the gains small at the beginning and increase them once it was sure that the helicopter was flying safely. The first test performed with the simulated gains showed that the helicopter was able to maintain hover conditions. However, the rate of climb/descent was quite very low and the PI gains were, therefore, increased.
- Finally, after each controller was good tuned, the full control system was tested over a squared flight pattern. The distance tracked by the helicopter and, therefore, the Vx track velocity were kept within the RC transmitter range and pilot good line of sight, in order to recover the helicopter if needed. Some experimental results are shown in figures 101-104.

During all the flight tests the helicopter was brought into hover condition by the RC pilot and then switched into autonomous mode.

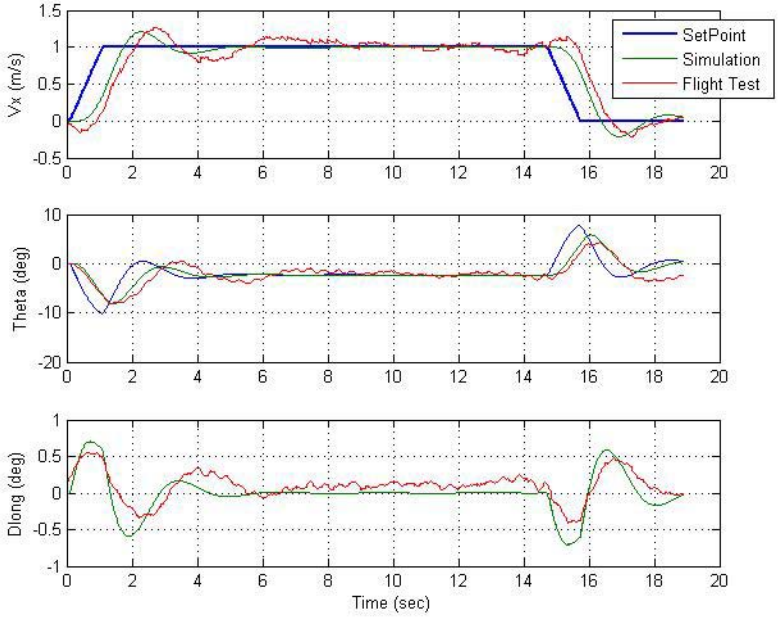


Figure 101: Simulate vs Experimental longitudinal controller tracking performance

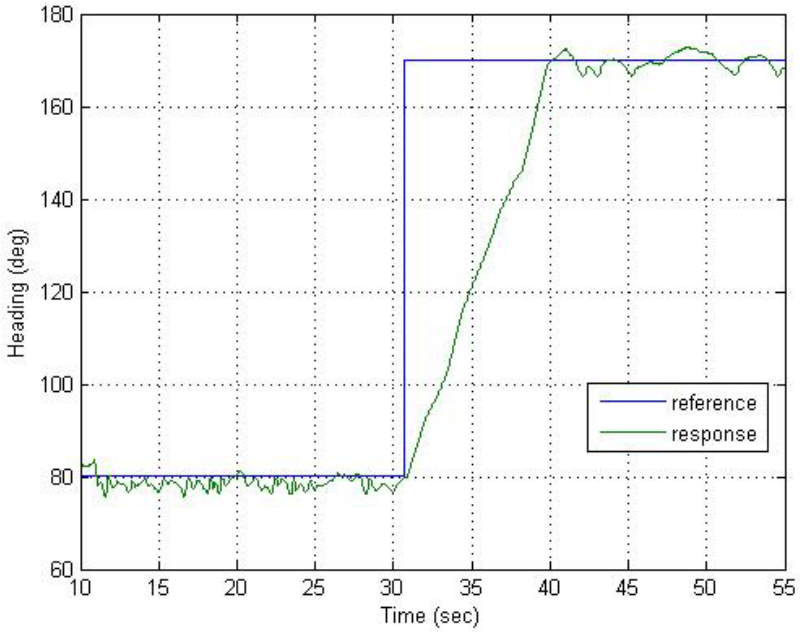


Figure 102: Heading "Control" flight test

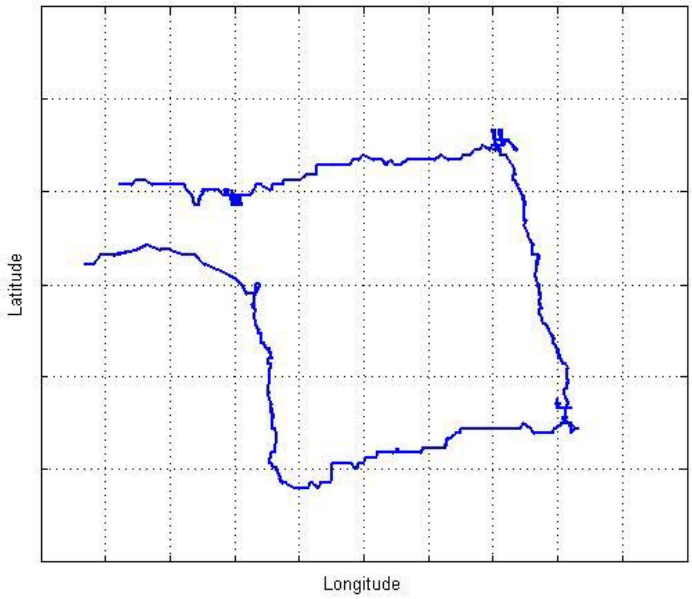


Figure 103: Flight Path

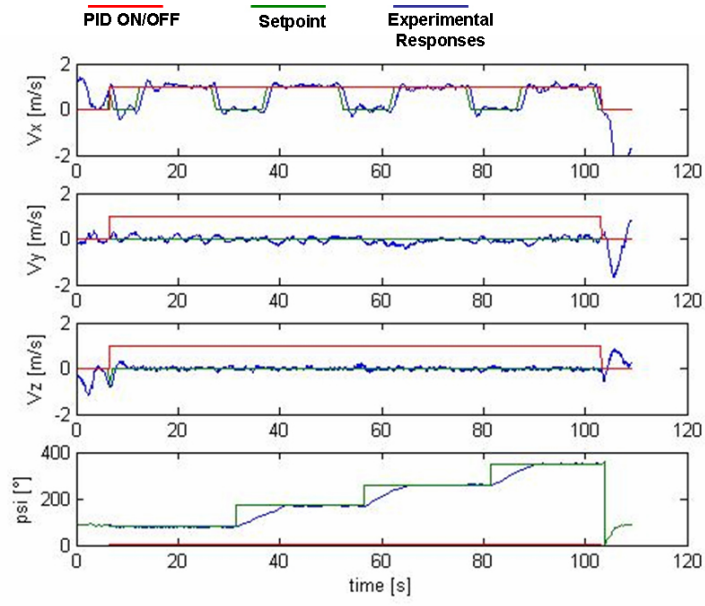


Figure 104: Flight Pattern Test

7.1 CONCLUSION AND OUTLOOK

An RUAV platform was set up using commercial and cost effective technology. Both the hardware and the software were integrated placing attention to modularity, growth potential, versatility and possibility for ease reconfiguration and software implementation.
HIL simulations and experimental flights were performed in order to test the feasibility to use the selected hardware and the developed software for helicopter control. The controller architecture was developed based on a simple nested PID structure. Results demonstrated that the RUAV system was able to provide accurate flight data measurements and good helicopter control capabilities.
The research activities carried out at the University of Bologna opened several and new research directions concerning the following major fields: RUAV Platform future developments, HIL improvement, mission simulation environment future activities.

Mission Simulation Environment and HIL Simulator

These simulation platform will be further improved. More sophisticated dynamics models will be implemented on the HIL simulator, including a more accurate model of al the flight sensors. Coupled with the developed RUAV platform, these simulation environments will provide useful test beds for safe ground pre-flight tests or for studying different control and navigation strategy. Researches in man machine interface and air system integration could be also be performed, which were addressed as one of the most critical technology aspect for the future development of the civil UAVs and their integration into the airspace.

RUAV Platform

The onboard navigation system software will be also tested in flight and further flights at higher speed will be made. Automatic take-off and landing flight mode will be also implemented and further flight tests will be performed.
Thanks to its modular architecture and accurate flight data measurement capabilities, the RUAV system may become a useful research test bench in several different field such as:
-aircraft /rotorcraft dynamic model identification
-researches in control and navigation laws (fast and ease software implementation could results also in a speed up of the research time)
- support in main machine interface research activities.

The feasibility to install the designed avionics hardware, integrated with additional redundant systems, on an ultralight helicopter will be also investigated.

References

[1] U.S. Air Force Scientific Advisory Board, *"Unmanned Aerial Vehicles in perspective: Effects, Capabilities and Technologies"*, SAB-TR-03-01, July 2003.

[2] CAPECON Consortium, *"Annex 1 Description of Work"*, CAPECON Project No. GRD1-2001-40162, Starting Date May 2002.

[3] Mouritsen S., Boer J-F., *Operational concept for local surveillance UAV missions,* DLR and NLR CAPECON report, January 2003

[4] Sevin C., *Internal Report on RW UAVs requirements and equipment*, CAPECON Report ECD D 2.4/1, 23 January 2003

[5] Mouritsen S., *Rotary wing requirements interim technical report,* DLR CAPECON report ID3/3, 4 February 2003

[6] Gian Marco Saggiani, Barbara Teodorani *"A matrix method for defining potential applications of a multirole Rotary UAV (RUAV) in Italy ",* CAPECON Project Internal Report no.1, January 2003.

[7] Saggiani G.M., Teodorani B., *"Rotary wing UAV potential applications: an analytical study through a matrix method ",* University of Bologna, on Aircraft Engineering and Aerospace Technology, vol. 76, No.1, 2004, pp.6-14

[8] Basset P.M., *Operational concept for rotary wing UAV out-of-line-of sight mission,* ONERA CAPECON report, February 2003

[9] Sevin C., *Internal Report on survey of potential applications for rotary-wing UAVs*, ECD CAPECON Report ID 2.1/2, 25 September 2002

[10] Cyrille Sevin, *"Rotary Wing - Civil UAVs - Concept Of Employment",* Rochester CAPECON Technical Meeting, July 2002

[11] R. Pretolani, G. Saggiani, B. Teodorani, *"Mini/Small Rotary Wing UAV Technologies"*, CAPECON Technical Report, Report ID 10.3, November 2005.

[12] L. A. Young, E. W. Aiken, NASA Ames Research Center, et al., *"New Concepts and Perspectives on Micro-Rotorcraft and Small Autonomous Rotary-Wing Vehicles"*.

[13] S. Mouritsen, E. Rehwald, DLR-German Aerospace Research Centre*, " Micro UAV Payloads and Data Link Survey Final Technical Report"*, CAPECON Report ID 10.6/10.7, November 2005

[14] R. E. Weibel R. J. Hansman, MIT*, "Safety Considerations for Operation of Different Classes of UAVs in the NAS",* AIAA-2004-6421, AIAA's 4th Aviation Technology, Integration and Operations (ATIO) Forum, 20 - 22 September 2004, Chicago, Illinois.

[15] B. L. Aponso, E. N. Bachelder, D. L. Lee, System Technology Inc., *"Automated Autorotation for Unmanned Rotorcraft Recovery"*, AHS International Specialist's MeetingOn Unmanned Rotorcraft, 18-20 January 2005, Chandler, Arizona.

[16] R. E. Weibel R. J. Hansman, MIT, *"An Integrated Approach to Evaluating Risk Mitigation Measures for UAV Operational Concepts in the NAS",* AIAA-2005-6957, AIAA's 5th Infoech@Aerospace Conference, 26-29 September 2005, Arlington, Virginia.

[17] S. Tsach, D. Penn and A. Levy Israel Aircraft Industries (IAI), *"Advanced Technologies and Approaches for Next Generation UAVs"*, ICAS 2002.

[18] R. Pretolani, G. Saggiani, B. Teodorani, *"RUAV Ground Support: Basic Requirements & First Layout Internal Report"*, CAPECON Report UNIBO ID 6.1/1, 31 July 2003.

[19] R. Pretolani, *"Progetto e realizzazione del sistema di navigazione, guida e controllo per un elicottero con capacità di volo autonomo",* II School of Engineering, University of Bologna, PhD Thesis, 2007

[20] B. Mettler, *"Identification, Modeling and Characteristics of Miniature Rotorcraft",* Kluver Academic Publishers,Boston, MA, 2002.

[21] R. Pretolani, G. Saggiani, B. Teodorani, *"Development of a mission simulation environment for Rotary Wing UAV"*, CAPECON Technical Report, Report ID 6.4, September 2004.

[22] M.L. Preatoni (AGUSTA, Varese Italy), R. Pretolani, G. M. Saggiani, B. Teodorani (DIEM- University of Bologna), *"An Integrated Simulation Environment as a Strategy in Rotorcraft UAVs preliminary design",* Presented at AHS meeting, Phoenix-Arizona, January 2005

[23] Leonard J., *Systems engineering fundamentals* Defense Systems Management College Press, Fort Belvoir Virginia, 1999

[24] Lee J., Nugent J. and Taylor B**.** *Advanced rotor control concepts*. Georgia Institute of Technology AE 6370 Team Project – American Helicopter Society RFP.

[25] Grinsell C., Thompson B., O'Brien P., Senga M. and Schrage D.P. *Gtmars.* Georgia Institute of Technology, American Helicopter Society student design competition.

[26] Cohen L., *Quality function deployment*, Addison-Wesley, Mass, USA, 1995

[27] RAO A et al., *Total quality management: a cross functional perspective,* John Wiley & Sons Inc., USA, 1996

[28] Knowles G., *Advanced quality tools,* Module Notes, WMG The University of Warwick, Coventry UK, 1997

[29] Raymer D.P:, *"Aircraft Design: A Conceptual Approach"*, AIAA EDUCATION SERIES

[30] B. Mettler, M.B. Tischler, and T. Kanade, *"System identification modeling of a small-scale unmanned rotorcraft for control design"*, Journal of the American Helicopter Society, 47(1):50–63, January 2002.

[31] B. Mettler, M. Tischler, T. Kanade, and W. Messner, *"Attitude control optimization for a small-scale unmanned helicopter",* Denver, CO, August 2000. AIAA Guidance, Navigation and Control Conference.

[32] B. Mettler, V. Gavrilets, E. Feron, and T. Kanade, *"Dynamic compensation for high-bandwidth control of small-scale helicopter"*, San Francisco, CA, January 2002. American Helicopter Society Specialist Meeting.

[33] M. McConley, *"Draper small autonomous aerial vehicle dynamic model",* Technical Report E41-98-091, Draper Laboratory, August 1998.

[34] J. G. Leishman, *"Principles of helicopter aerodynamics",* Cambridge University Press, New York, 2000.

[35] Ulrik B. Hald, Mikkel V. Hesselbaek, Jacob T. Holmgaard, Christian S. Jensen, Stefan L. Jakobsen, Martin Siegumfeldt*, "Autonomous Helicopter Modelling and Control",* Aalborg University, May 2005.

[36] T. D. Talbot, B. E. Tingling, W. A. Decker, and R.T. Chen, *"A mathematical model of a single main rotor helicopter for piloted simulation"*, Technical Memorandum 84281, NASA, 1982.

[37] A.R.S. Bramwell, *"Bramwell's Helicopter Dynamics",* AIAA, Reston VA, 2001.

[38] G.D. Padfield, *"Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling",* AIAA Education Series, Reston, VA, 1996.

[39] V. Gravilets, B. Mettler, E. Feron, *"Dynamic model for a miniature acrobatic helicopter",* MIT-LIDS report, # LIDS-P-2580, 2003

[40] Einthoven,P., Morse,C., *"Energy Management",* presented at the AHS Flight Controls and Crew Systems Design Specialists' Meeting,Philadelphia, PA, October 9-11, 2002

[41] Prouty R.W., *"Helicopter Performance, Stability and Control",* Krieger Publishing Company, Inc., 1990.

[42] S. Mouritsen (DLR,Braunschweig, Germany), R. Pretolani, G. M. Saggiani, B. Teodorani (DIEM- University of Bologna), *"Application of an Active Joystick in a Rotorcraft UAV Ground Control Station",* Presented at AHS meeting, Phoenix-Arizona, January 2005

[43] M. Niculescu, *"Lateral Track Control Law for Aerosonde UAV",* AIAA 2001-0016, University of Washington, Seattle, January 2001

[44] P. S. Anderson, "*Development of a UAV ground control station",* M.SC. Thesis Linköping University, 2002

[45] C. Munzinger, *"Development of a Real-time flight simulator for an experimental model helicopter",* Georgia Institute of Technology School of Aerospace Engineering, Atlanta Dec. 1998

[46] A. Boccalatte, F. De Crescenzio, F. Flamigni, F. Persiani *"A training environment for aircraft pilots by means of virtual reality techniques"*, Proceedings of the XIII ADM International Conference, Napoli, June 4th - 6th 2003.

[47] ftp://edcsgs9.cr.usgs.gov/pub/data/srtm/

[48] Preatoni Marzio, Agusta, *"Rotary Wing UAV Configurations –Final Technical Report",* CAPECON Technical Report, Report ID 6.4, May 2004.

[49] J.F. Boer, F. Fresta, H. Haverdings, M.L. Preatoni, R.Pretolani, G.M. Saggiani, B.Teodorani, AGUSTA, NLR, DIEM-University of Bologna, *" Small/Mini Rotary Wing UAV Configuration"*, CAPECON Report ID 10.5, November 2005.

[50] Cooper,G.E., Harper,R.P., *"The Use of Pilot Rating in the Evaluation of Aircraft Handling Qualities"*, NASA TN D-5153, April 1969.

[51] www.ni.com

[52] Joerg S. Dittrich, *"Design and Integration of an Unmanned Aerial Vehicle Navigation system",* School of Aerospace Engineering Georgia Institute of Technology, May 2002

[53] *"NAV 420 CrossBow User Manual"*; www.xbow.com

[54] *"SRF08 Ultra sonic range finder Technical Specification";* www.robotitaly.com

[55] *"The I2C-BUS Specification",* Version 2.4, Doc. N° 93983934001, Philips Semiconductors, Janaury 2000

[56] "*Using Quadrature encoder with E series DAQ Cards"*, National Instruments application notes

[57] *"Vibration and Shock Theory;* www.lordmpd.com

[58] http://www.mathworks.com/products/matlab

[59] The MathWorks Inc. ed. T.M.W. Inc.; Natick, MA. *"User's Guide for Matlab Simulink, Toolboxes"*, 2001

[60] *"Fundamentals of FFT signal Anlysis and measurements",* Labview Bookshelf, 2004

[61] A.V. Oppenheim and R.W. Schafer, *"Discrete time signal processing",* Signal Processing Series, Prentice Hall, Englewood Cliffs, 1989

[62] www.boeing.com ; Penn State University study on Raptor 60 "Good Vibrations"

[63] István Kollár, *"Frequency Domain System Identification Toolbox",* MATLAB® User Guide

[64] L. Liung, *"System Identification*", Prentice Hall, 1987

[65] P.G. Hannel and R.V. Jategeonkov, *"The evolution of Flight vehicle system Identification",* Agard Lectures Series on Rotorcraft System Identification, 1995

[66] J.S. Bendat and A.G. Piersol *"Engineering Applications of Correlation and Spectral Analysis",* John Whiley & Son, New York, NY, 1993

[67] M. Tibaldi, *"Progetto di sistemi di controllo",* Pitagora Editrice, Bologna 1995.

[68] G. Buskey, J. Roberts, P. Corke, G. Wyeth, *"Helicopter Automation using a low-cost sensing system"*, IEEE International Conference on Systems, Man and Cybernetics, 2005.

[69] K. Sprague, V. Gravilets, D. Dugail, B. Mettler, E. Feron, *"Design and applications of an avionics system for a miniature acrobatic helicopter",* Digital Avionics Systems Conference, October 2001, Daytona Beach, Florida.

[70] C. Castillo, W, Alvis, M. Castillo-Effen, K. VAlavanis, W. Moreno, *"Small scale helicopter analysis and controller design for non-aggressive flights"*, Center for Robot Assisted Search and Rescue, University of South Florida.

[71] M. LaCivita, W. Messner, and T. Kanade, *"Modeling of small-scale helicopters with integrated first-principlesand integrated system identification techniques",* Monteal, Canada, June 2002. Presented at 58th Forum of American Helicopter Society.

[72] V. Gavrilets, I. Martinos, B. Mettler and E. Feron, *"Control Logic for Automated Aerobatic Flight of a Miniature Helicopter"*, Massachusetts Institute of Technology, Cambridge, MA 02139.

[73] LabView PID Control Toolset User Manual

[74] J.A. Farell, M. Barth, *"The Global Positioning System & Inertial Navigation"*, Mc GrawHill, 1999