# TRABAJO FINAL DE CARRERA

**TÍTULO DEL TFC:** **Setup and Integration of a Quadrotor used as a research demonstrator**

**TITULACIÓN:** **Ingeniería Técnica Aeronáutica, especialidad Aeronavegación**

**AUTOR:** **Kenneth Llonch González**

**DIRECTOR:** **Cristina Barrado Muxí**

**CO-DIRECTOR:** **Jorge Ramírez Alcántara**

**FECHA: 22 de Noviembre de 2011**

**Título:** Setup and Integration of a Quadrotor used as a research demonstrator

**Autor:** Kenneth Llonch González

**Director:** Cristina Barrado Muxí

**Co-director:** Jorge Ramírez Alcántara

**Fecha:** 22 de Noviembre de 2011

## Resumen

Este trabajo final de carrera está enmarcado dentro de la realización de un PROFEDI (Programa de Formació d'Estudiants en Departaments i Instituts) dentro del grupo de investigación ICARUS de la UPC (Universitat Politècnica de Catalunya).

El grupo ICARUS está localizado en el Parc Mediterrani de la Tecnologia de Castelldefels (Barcelona). Actualmente tiene distintos proyectos en marcha siendo el Sky-Eye el más importante. El proyecto Sky-Eye consiste en la detección de hot spots y está subvencionado por la Generalitat de Catalunya.

Un PROFEDI consiste en la realización de un proyecto en un departamento de la Universidad, en este caso de la Escola d'Enginyeria de Telecomunicacions i Aeroespacial de Castelldefels en el departamento de Arquitectura de Computadores. En él, el alumno investiga nuevos campos y ramas de sus estudios y amplía así sus conocimientos.

Este PROFEDI consiste en la construcción de un quadrotor que será usado en un futuro como demostrador docente por la universidad. El proyecto lo he realizado junto con otro estudiante, Francesc Pera González. Inicialmente se documentó un parte inicial de "Safety" en común, y posteriormente decidí hacer todo el proceso de integración y "setup".

En este TFC se detallarán qué segmentos integran el UAV, cómo interaccionan entre ellos y la correcta integración de los sistemas que los forman. Así mismo, se explicará la correcta instalación, configuración y edición de todos los aspectos software necesarios para poder simular a través de la "Ground Control Station" o poder volar el quadrotor siempre con las medidas de seguridad necesarias.
El software usado en el proyecto es de acceso libre y ha sido creado por el grupo Paparazzi de la ENAC (Ecole Nationale de l'Aviation Civile), los cuales tienen una web [1] que nos sirvió de soporte durante todo el proyecto.

Tratándose de un demostrador docente donde la interacción con estudiantes (a los cuales se les permitirá editar y configurar el vuelo del quadrotor) será constante, el aspecto de la seguridad, es uno de los más importantes.

**Title:** Setup and Integration of a Quadrotor used as a research demonstrator

**Author:** Kenneth Llonch González

**Director:** Cristina Barrado Muxí

**Co-director:** Jorge Ramírez Alcántara

**Date:** November, 22nd 2011

## Overview

This final degree project has been realized within a PROFEDI (Programa de Formació d'Estudiants en Departaments i Instituts) in the ICARUS research group which belongs to the UPC (Universitat Politècnica de Catalunya).

The ICARUS research group is located in the PMT (*Parc Mediterrani de la Tecnologia*) of Castelldefels (Barcelona). Currently they have several projects underway, the Sky-Eye being the most important one. The Sky-Eye project involves the detection of hot spots and is funded by the Generalitat de Catalunya.

A PROFEDI consists in the realization of a project within a department of the university, in this case the *Escola d'Enginyeria de Telecomunicacions i Aeroespacial* of Castelldefels, in the Computer Architecture Department. Here, students explore new fields and branches of study in order to expand their knowledge.

This PROFEDI involves the implementation of a quadrotor that will be used in future as a university demonstrator. This project has been carried out with another student, Francesc Pera González.
Initially we redacted the first steps in common (mainly safety process), and then I decided to draft the system integration and setup process.

In this TFC the segments which make up the UAV, how they interact with each other and the correct integration of the systems will be detailed.
Furthermore, the proper installation, configuration and software setup, which are essential to correctly simulate by way of the Ground Control Station and to fly the Quadrotor with the proper safety measures will be explained.

The software used in this Project is open source and has been developed by the Paparazzi group from ENAC (Ecole Nationale de l'Aviation Civile) who have a website [1] which was used as a help tool during the entire project.

As the quadrotor is intended to be a demonstrator with teaching purposes where there will be significant interaction with students (who will be allowed to edit and configure the quadrotor flight), the safety will be one of the most important considerations.

**Agradecimientos**

First of all I would like to thank
Gautier Hattenberger and Michel Gorraz,
members of the Paparazzi developers team,
who were always willing to help me and solve any
problem that arose over the course of the project,
during our stay in Toulouse and also by email.

Al grupo de investigación ICARUS
los cuales confiaron en mí y
me permitieron aprender tantas
cosas de aspectos tan variados.

A Cristina Barrado y a Jorge Ramírez,
los cuales me ayudaron y aconsejaron
en todas las dudas que surgieron.

A Francesc Pera, porque mucho tuvimos
que luchar para conseguir los resultados.

A mis padres, porque gracias a ellos llegué hasta aquí.

*Nunca consideres el estudio como una obligación, sino como una
oportunidad para penetrar en el bello y maravilloso mundo del saber.
Albert Einstein*

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONIMS

| | |
|---|---|
| *AGT* | *Air Ground Transition* |
| *CCA* | *Common Cause Analysis* |
| *DC* | *Direct Current* |
| *ENAC* | *Ecole Nationale de l’Aviation Civile* |
| *FDD* | *Floppy Disk Device* |
| *FDM* | *Flight Dynamics Model* |
| *FHA* | *Functional Hazard Assessment* |
| *FM* | *Frequency Modulation* |
| *FP* | *Flight Plan* |
| *GB* | *Gigabyte* |
| *GPS* | *Global Positioning System* |
| *GS* | *Ground Station* |
| *HDD* | *Hard Disk Device* |
| *ICD* | *Interface Control Document* |
| *IMU* | *Inertial Measurement Unit* |
| *OZ* | *Operational Zone* |
| *PC* | *Paparazzi Center* |
| *PMT* | *Parc Mediterrani de la Tecnologia* |
| *PPM* | *Pulse Position Modulation* |
| *PRA* | *Particular Risk Analysis* |
| *PS* | *Power Supply* |
| *RC* | *Radio Controller* |
| *UAS* | *Unmanned Aircraft System* |
| *UAV* | *Unmanned Aerial Vehicle* |
| *USB* | *Universal Serial Bus* |
| *VTOL* | *Vertical Take-off and Landing* |
| *WGS84* | *World Geodetic System 1984* |
| *X-BLDC* | *Extreme - Brushless Direct Current* |
| *XML* | *Extensible Mark-up Language* |
| *ZSA* | *Zonal Safety Analysis* |

# INTRODUCTION

Currently, the world of aeronautics is one of the most dynamic branches of engineering. This is due not only to the evolution and improvement of existing aircraft but also because of the layout and development of new models.

The UAS (Unmanned Aircraft System) is an aircraft, which is as indicated by the name, an aircraft that can fly without a flight crew on board and belongs to this new group. In order to fly this type of aircraft, only the aircraft and a Ground Station where someone will be managing it and checking the systems behaviour are necessary.

While UAS has been in use for some years, it is only in recent times that it has gained ever greater importance.

It is within the military world where UAS is developing fastest, and new systems are being tested and taken into consideration for later application in civil UAS.
The function of UAVs depend on their civil or military goal. Military UAS have mainly surveillance and exploration functions. On the other hand, for civil applications, UAS are used mainly for search and rescue purposes.

Nonetheless in civil applications, smaller UAS (called microUAVs) are emerging. They are used by many people for hobby and enjoyment purposes during their free time.

You can see in Table Intro.1 the UAV categories according to their features.
In the table there are UAVs for civil and also military applications. [2]

| UAV Categories | Acronym | Range (Km) | Flight Altitude (m) | Endurance (Hours) | Take-off Mass |
|---|---|---|---|---|---|
| Tactical | | | | | |
| Micro | μ | <10 | 250 | 1 | <5 |
| Mini | Mini | <10 | 150 to 300 | <2 | <30 |
| Close Range | CR | 10 to 30 | 3000 | 2 to 4 | 150 |
| Short Range | SR | 30 to 70 | 3000 | 3 to 6 | 200 |
| Medium Range | MR | 70 to 200 | 5000 | 6 to 10 | 1250 |
| Medium Range Endurance | MRE | >500 | 8000 | 10 to 18 | 1250 |
| Low Altitude Deep Penetration | LADP | >250 | 50 to 9000 | 0,5 to 1 | 350 |
| Low Altitude Long Endurance | LALE | >500 | 3000 | >24 | <30 |
| Medium Altitude Long Endurance | MALE | >500 | 14000 | 24 to 28 | 1500 |

| | | | | | |
|---|---|---|---|---|---|
| Strategic | | | | | |
| High Altitude Long Endurance | HALE | >2000 | 20000 | 24 to 48 | 12000 |
| Special Purposes UAVs | | | | | |
| Unmanned Combat Aerial Vehicle | UCAV | 1500 | 10000 | approx. 2 | 10000 |
| Lethal | LETH | 300 | 4000 | 3 to 4 | 250 |
| Decoy | DEC | 0 to 500 | 5000 | 4 | 250 |
| Stratospheric | STRATO | >2000 | >20000 & >30000 | >48 | ND |
| Exo-Stratospheric | EXO | ND | >30000 | ND | ND |

**Table Intro.1** UAV Classification

Almost all the civil UAVs belong to the micro and short range group.
Some examples of MicroUAVs are the small aircraft and the helicopters controlled by RC (Radio Controller). It is within this last group where we find the Quadrotors which are MicroUAVs and which will be developed in this project.

In Table Intro.2 there is a comparison between the most well-known short range UAVs [2].

| Airframe | Helicopter | Airplane | Dirigible | Quadrotor |
|---|---|---|---|---|
| Hover Capability | *** | | **** | *** |
| Movement Speed | *** | **** | * | ** |
| Maneuverability | *** | * | * | **** |
| Flight range | ** | *** | **** | * |
| Resistance to external disturbances | ** | **** | * | ** |
| Stability | * | *** | *** | ** |
| Vertical Flight availability | **** | * | ** | **** |
| Payload availability | *** | **** | * | ** |
| Flight indoors availability | ** | * | *** | **** |
| Maximum Altitude | ** | **** | *** | * |

**Table Intro.2** Short Range UAVs features

A Quadrotor is a VTOL (Vertical Take-Off Landing) MicroUAV formed by 4 rotors.
Two rotors spin clockwise and two anticlockwise.

The reason for that is because each engine produces thrust and torque, but also drag is induced in the opposite direction. With this configuration the quadrotor does not start spinning around its yaw axis.

Now, some details about the structure of this project:

Before the completion of this Final Degree Project, some Safety analyses were carried out. A very brief description is detailed in ANEX A.

This project firstly details in chapter 1, the OPERATIONAL CONCEPT, where the area and the most important requirements that the quadrotor must fulfil will be defined.

In chapter 2, a brief analysis of the ARCHITECTURE will be carried out. To see all the associated architecture, see ANEX B.

After the architecture, the INTEGRATION process will be explained in chapter 3. In this chapter, all the connections and interactions between all the systems will be detailed.

At this point it is necessary to start dealing with the software and this is done in chapter 4, SOFTWARE CONFIGURATION. In this chapter the setup process will be detailed step by step: Linux installation, Paparazzi downloading and the correct setup of the xml files so that the quadrotor flies well.
To see the entire xml files, see the ANEX C.

Once the setup process has been explained it will be time to detail the user-Paparazzi interface and how they interact with one another.
This will be detailed in chapter 5, PAPARAZZI CENTER INTERFACE.

# CHAPTER 1. OPERATIONAL CONCEPT

Once the main information about the project has been seen, this chapter will explain the aim more accurately, features of the quadrotor and some of the project's limitations.

The EETAC (Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels) faculty, which belongs to the UPC University, carries out diverse activities for secondary school students (from around 12 to 17 years old). In these activities, students gain knowledge and experience in an interactive way such as through the building of "rockets" which are propelled by the reaction resulting from mixing vinegar and bicarbonate.

You can see this activity in progress in Figure 1.1. [3]

**Figure 1.1** Vinegar and bicarbonate rocket activity

One of the most important goals of this type of activity is to allow the students to perform the experiments by themselves. For this reason, we explain to them how to complete it with only a brief demonstration of the process and the chemistry involved.

While these activities are interesting, activities which are more relevant to the aeronautical world would be much more challenging.

For this reason, there was a desire within the faculty to carry out more technically challenging activities such as a project which would involve integrating various hardware components and establishing software protocol which could then be demonstrated to students.

The ICARUS research group team, which is also located in the EETAC faculty and which is composed of teachers, researchers and some students, came up with two ideas that could accomplish this aim.

As UAVs are becoming increasingly more important, ICARUS thought it could be one of the best ways in which to boost this aeronautical branch in the university. Furthermore, doing these activities might allow students to discover what can be done in the university and stimulate their interest in the aeronautical world.

The two vehicles proposed by ICARUS were UAV prototypes. [4]
It was believed that the helium zeppelin and the quadrotor to be developed in this project could satisfy this new goal for more technological based activities. The university appraised the ideas presented and accepted the projects.

Thus, the idea of developing a quadrotor emerged as a project which would be presented by the ICARUS research group for the EETAC so that it could be used for demonstration purposes.

Now that we understand how this project arose, let’s take a look at it in more detail.

The airframe used is an AscTec X-3D BL (see Figure 1.2) and the Paparazzi Booz Autopilot will be used instead of the AscTec Autopilot (X-Base). [5]
The quadrotor will be flown in a restricted area called the Operational Zone (OZ). Therefore, entry in this zone will be strictly forbidden, especially during flight tests.

It is better to cordon off the area with a fence or a ribbon to prevent intrusion.

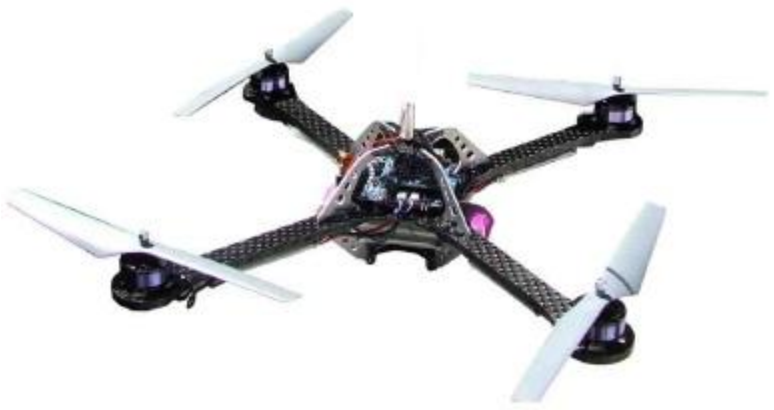


**Figure 1.2** AscTec x-3D BL Airframe

The quadrotor will be used as a research prototype for demonstration purposes. For this reason it must strictly adhere to safety requirements. The quadrotor will be unable to leave the Operational Zone due to a redundant safety system.
This is accomplished through the use of hardware and software restrictions.

In order to give some idea as to what activities can be accomplished by students with the quadrotor, I will outline two:

One activity that could be carried out with students would be as follows:

Rocks would be left in the Operational Zone (see Figure 1.3) [6]. 3 or 4 would be fine. Students would then try to follow a path such that the rocks always remained below the quadrotor.
The more accurate the path, the higher the “score” would be for the student.

Another activity could be to make an easy “circuit”, also using rocks. The aim would be to try to complete the circuit without going outside. The faster the lap without crashing or leaving the path, the higher the “score” would be for the student.

The interactive point of the quadrotor project is, as you can conclude from the activities proposed, to allow the user to fly the quadrotor and edit a flight plan, and permit the quadrotor to easily follow instructions.

If the hardware system fails, and the quadrotor tries to leave the Operational Zone, the software will prevent this by using a software restriction.

If during the flight, one of the waypoints is placed outside the Operational Zone, and the quadrotor tries to go to this waypoint, when it crosses the maximum navigation radius of 20 m, the quadrotor will return to the Home waypoint.

The quadrotor will be flown through an RC (Radio Controller) system and data will be monitored on the Ground Station. There, we can control the displays such as for the Battery, the waypoints, the telemetry, etc.
In order to obtain this information we will use a USB-miniUSB connection.

It is very important that the quadrotor is always flown in good weather conditions, meaning, only on sunny days without wind, to avoid likely damage to the structure.
Also moisture can lead to certain problems for the systems and the wires.

The quadrotor will be flown over the plain located at the EETAC University with a virtual box of 40*40*15m, centred as shown in Figure 1.3.

The altitude limitation (15 meters) will be set by the software. In case of a software error, a fishing line will only allow it to fly, at a maximum of 20 meters above ground. As the quadrotor will not fly above 300 meters, communication with ATC will not be required. [7]

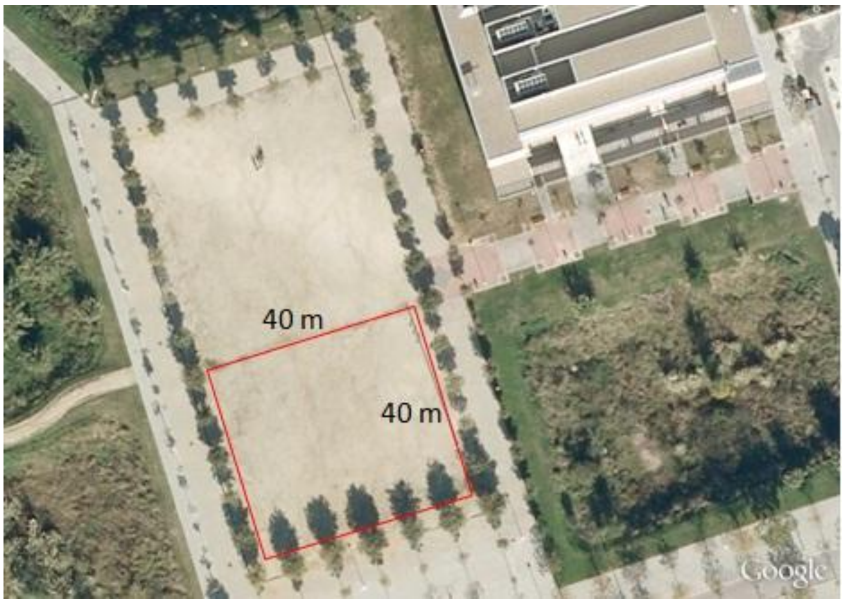


**Figure 1.3** Operational Zone

# CHAPTER 2. ARCHITECTURE

In this chapter a brief explanation of the physical architecture of the quadrotor will be presented.

The quadrotor system will be divided into smaller subsystems. This will allow us to detail the architecture in a simpler way:

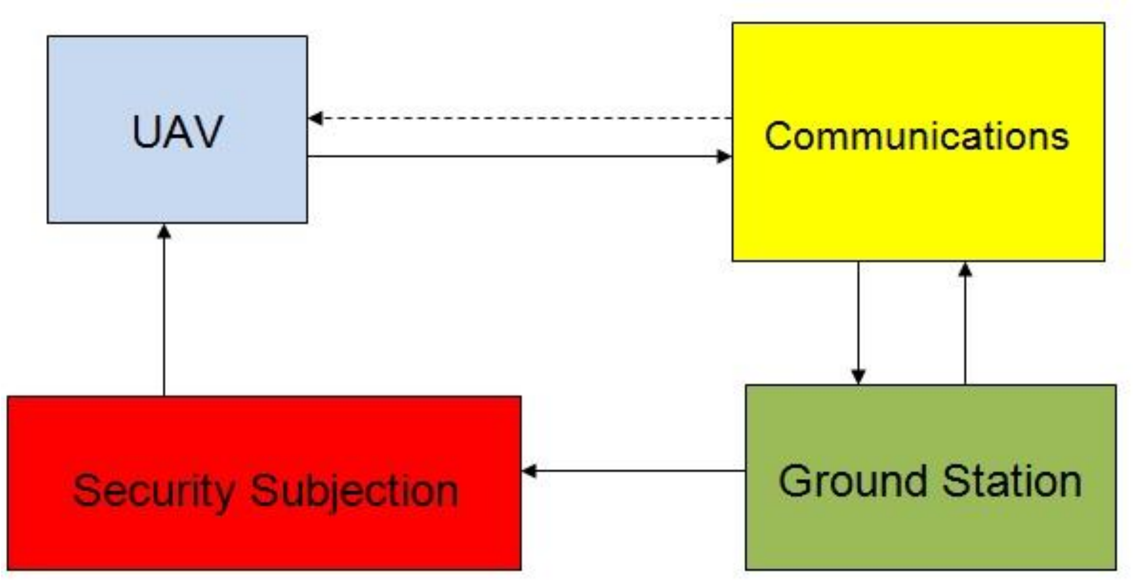


**Figure 2.4** Subsystems scheme

As shown in Figure 2.4 there are 4 main subsystems:

UAV
Communications
Security Subjection
Ground Station

The solid arrow means that the connection is physical and the dashed arrow means that it is wireless.

To see the architecture of the quadrotor in detail, see ANEX B

# CHAPTER 3. INTEGRATION

In this chapter the logical and physical connections between all the systems that belong to the quadrotor through the corresponding ICD (Interface Control Document) will be described.

*Note: Henceforth, the wires suitability must be checked before soldering or crimping the connectors in order to plug them in correctly.*

*Note: All schematics present in this chapter have been taken from the Paparazzi website. [1]*

## 3.1 Main Board – GPS

The connections have 8 pins in both boards, but we will only be using 4 pins since the other 4 are not required.
The GPS (Global Positioning System) gives longitude and latitude data in order to locate the quadrotor at any given moment.

When you activate the GPS for the first time, it takes approximately 10 minutes to download the ephemerides. Thereafter, this waiting time will no longer be necessary in order for it to find its location.

In Figure 3.5 the lower view of the GPS board is shown. The connection into which the connector must be plugged is circled in red.
On the other hand, in Figure 3.6 the scheme of the upper view of the board and the data for each pin is shown.

Bear in mind that the data for each pin is written on the board.

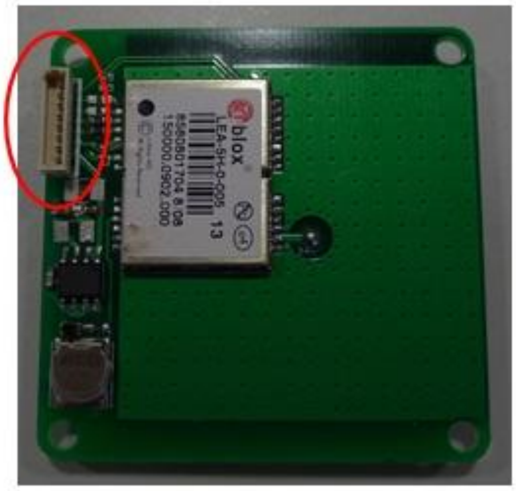


**Figure 3.5** GPS Board (Back side)

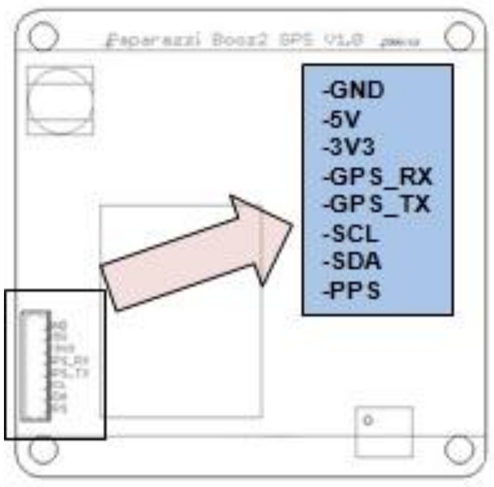


**Figure 3.6** GPS Schematic (front side)

In order to know where to plug each wire, it is first necessary to state the aim of each pin. Pin nomenclature is a follows:

GND: This is the ground pin

5V: This supplies the board with 5 Volts.

3V3: This supplies the board with 3'3 Volts. Pin not used.

LPC_TXD_0 and GPS_TX: Both pins used for data emission.

LPC_RXD_0 and GPS_RX: Both pins used for data reception.

SCL: I2C line. Pin not used.

BOOT/SDA: I2C line. Pin not used.

PPS: Synchronization signal. Pin not used.

Wires must be plugged in as follows:

| Main Board Pin | GPS Pin |
|---|---|
| GND | GND |
| 5V | 5V |
| 3V3 | --- |
| LPC_RXD_0 | GPS_TX |
| LPC_TXD_0 | GPS_RX |
| SCL | --- |
| SDA | --- |
| PPS | --- |

**Table 3.3** Pin connections between Main Board and GPS

The order of the pins in the Main Board is detailed in Figure 3.7:

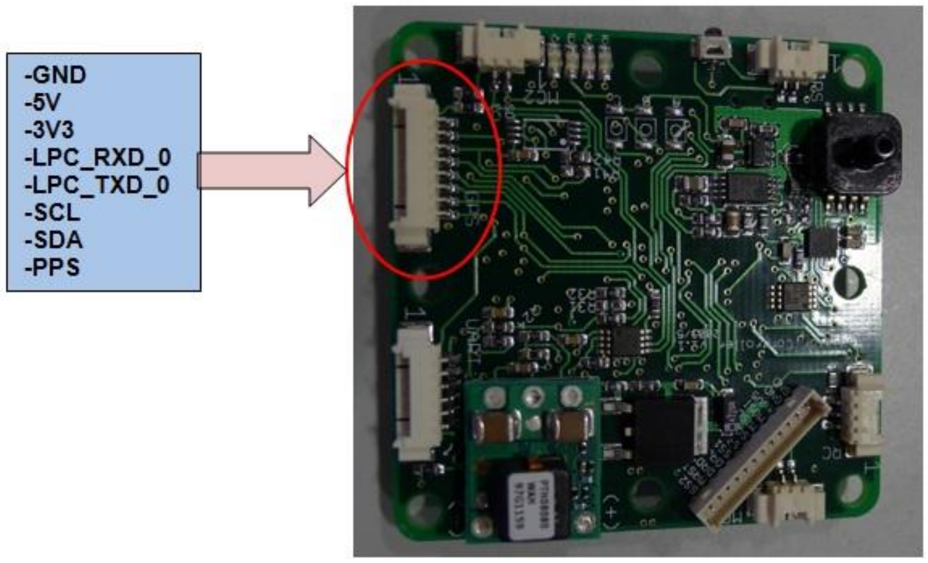


**Figure 3.7** Main Board with pins information

We will use an 8 pin Picoblade connector, but remember that only the connections relationships as shown in Table 3.3 are soldered or crimped.

## 3.2 Main Board – IMU

The IMU used in this project is Booz IMU v1.01.

We will use a 12 pin Picoblade connector, but only the 10 central pins are soldered or crimped.

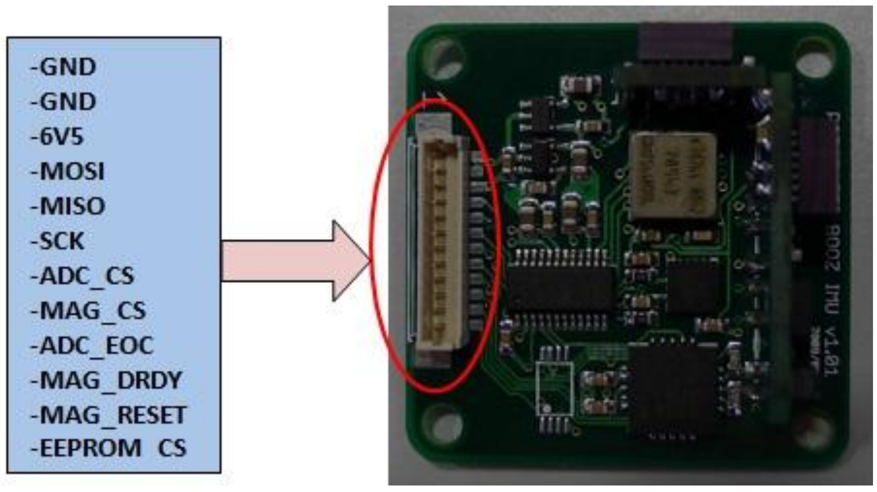


**Figure 3.8** IMU board with Pins information

The purpose of each pin is as follows:

GND: Ground. Pin not used.

GND: Ground.

6V5: Pin that supplies with 6,5 V the IMU.

MOSI: Master Output, Slave Input (output pin from master).

MISO: Master Input, Slave Output (output pin from slave).

SCK: Serial Clock.

ADC_CS: Analogical to Digital Converter Chip Select.

MAG_CS: Magnetometer Chip Select.

ADC_EOC: Analogical to Digital Converter End of Conversion.

MAG_DRDY: Magnetometer Data Ready.

MAG_RESET: Magnetometer Reset.

EEPROM_CS: Electrically Erasable Programmable Read Only Memory. Pin not used.

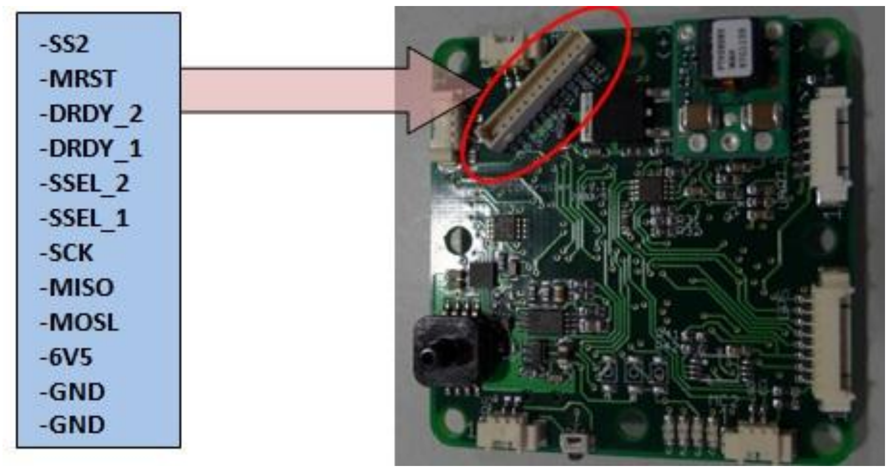


**Figure 3.9** Main Board and pins information

| Main Board Pin | IMU Pin |
|---|---|
| GND | --- |
| GND | GND |
| 6V5 | 6V5 |
| MOSI | MOSL_1 |
| MISO | MISO_1 |
| SCK | SCK_1 |
| ADC_CS | SSEL_1 |
| MAG_CS | SSEL_2 |
| ADC_EOC | DRDY_1 |
| MAG_DRDY | DRDY_2 |

| | |
|---|---|
| MAG_RESET | MAG_RESET |
| EEPROM_CS | --- |

**Table 3.4** Pins between the Main Board and the IMU

The wires must be soldered as shown in Table 3.4.

## 3.3 Main Board – Power Supply

Both boards will be connected by a wire soldered onto the Power Supply and plugged into the Main Board.
In the Figure 3.10 the place where the wire is to be soldered on the Power Supply Board is circled.

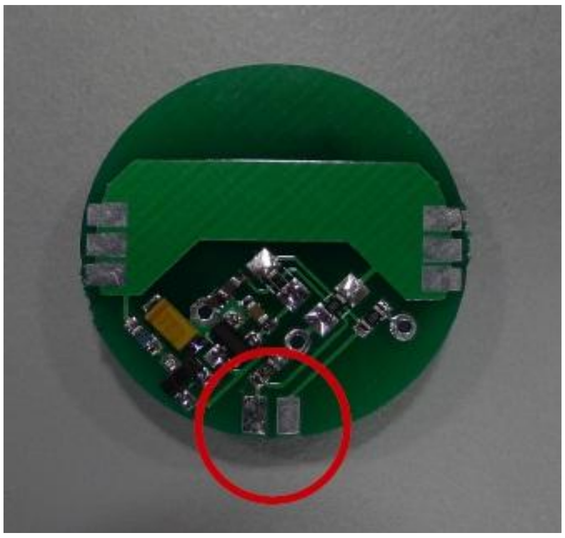


**Figure 3.10** Inferior side of the Power Supply

The wire will be plugged into the connection circled in red in Figure 3.11:

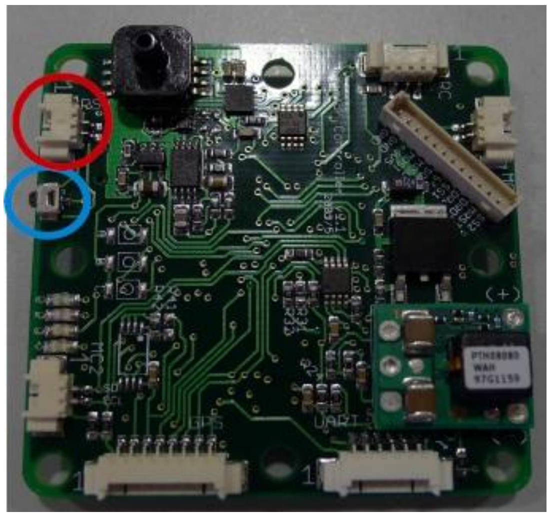


**Figure 3.11** Power Supply connection in the Main Board

The blue circle in Figure 3.11 shows the switch that will allow us to turn the Main Board on and off in future.
In Figure 3.12 the yellow circle indicates where the connector is to be plug in:

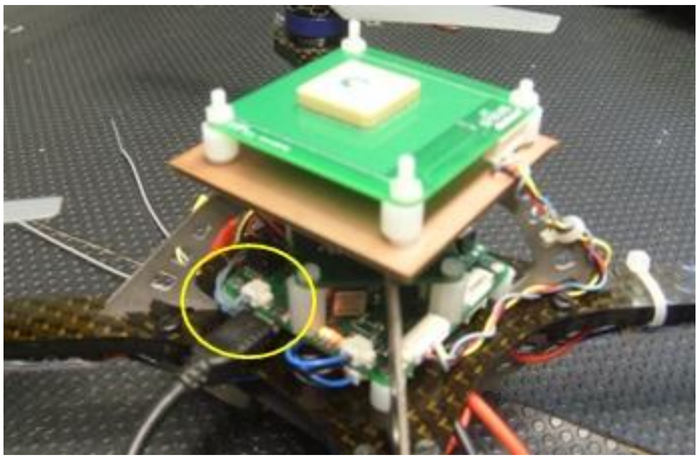


**Figure 3.12** Image detailing where to plug the PS connector

## 3.4 Main Board – Encoder

In this ICD we will see the two different encoder boards that were used during the project and how to connect them to the Main Board.

### 3.4.1 ATMEGA328 Encoder Board

Firstly, we wanted to use the ATMEGA328 Encoder Board that was advertised on the Paparazzi website and available from PPZUAV. [8]

We had some problems such as there being no output signal from the encoder, and so we contacted David Conger from PPZUAV.

We tried to solve the problems by re-soldering, but they continued.

Finally, we decided to reload the firmware instead, but neither HyperTerminal nor avrdude could establish connection with the micro. [9]

For this reason we thought that maybe the Bootloader had not been correctly set up and that this could be leading to these problems.
To solve it, we needed and ISP Bootloader, but as we did not have, we discarded the idea.

If in future, the ATMEGA328 board is purchased, the connection should be as shown in Figure 3.13.

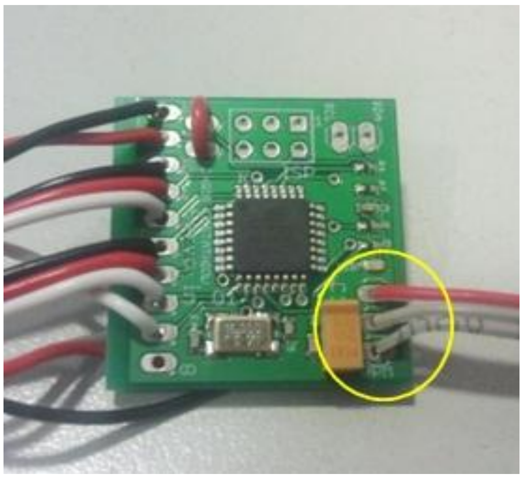


**Figure 3.13** Wires Soldering onto the Encoder ATMEGA328

The red wire is the PPM signal, the wire located in the middle is the supply and the other wire (outermost) the GND.
The connection to the Main Board will be in the same place and will have the same wire order as shown in Figure 3.14.

### 3.4.2 MeekPE Encoder Board

During our stay in Toulouse, Paparazzi developers gave us the first prototype of another encoder called MeekPE PPM Encoder Board. We opted to use this board until such time as we could decide which board would be best for us.
The Main Board and the MeekPE encoder will be connected through 3 wires. The GND, the supply of 5V and the PPM signal wire (black, red and yellow wires respectively as in Figure 3.14).

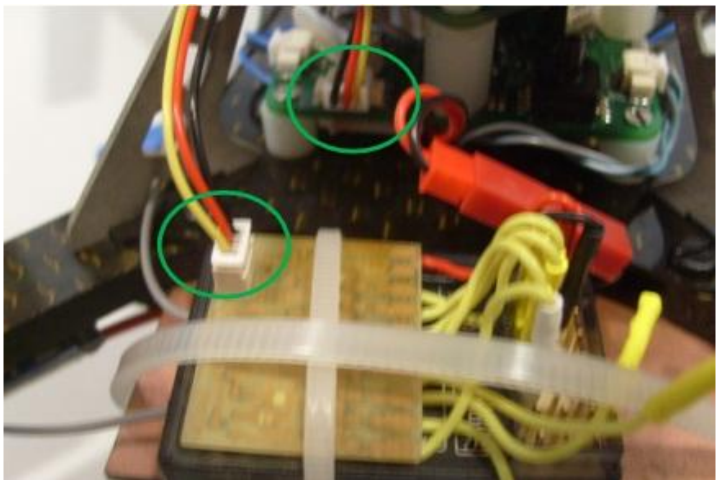


**Figure 3.14** Connection Encoder MeekPE - Main Board

In the next figure, the location where the connector with the 3 wires must be plugged into the Main Board is circled.
The correct order of the wires can be checked in Figure 3.14.

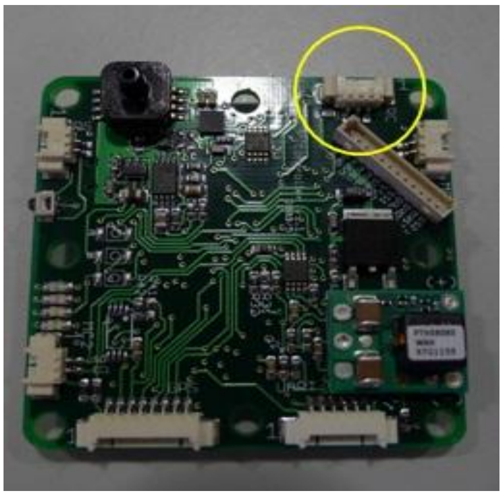


**Figure 3.15** Location of the RC connector in the Main Board

*Note: Remember to always check how to crimp or solder the wires properly to fix into the Main Board connector.*

## 3.5 Power Supply – Battery

In this chapter the connections between the Power Supply and the Battery will be detailed.

The Battery used is a Li-Po. It has 11.1V, 800Ah and its weight is 67gr.
It will be located in the inferior half of the core as shown in Figure Anex 7.

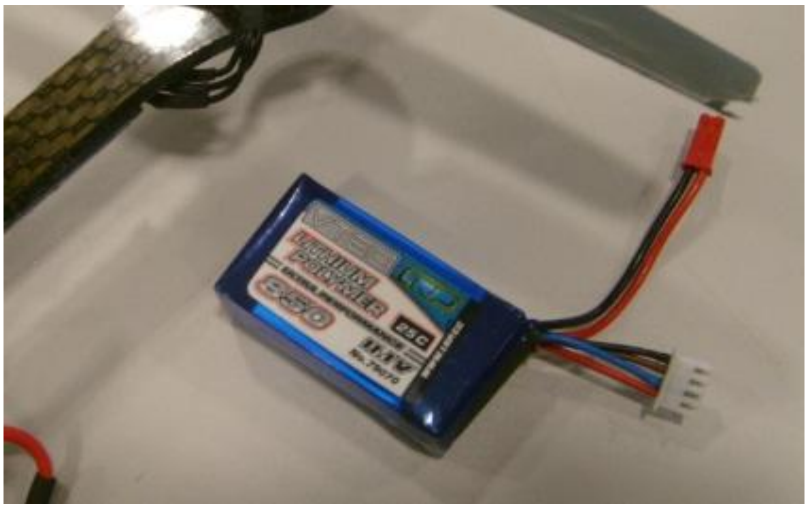


**Figure 3.16** Battery used in the project

The battery has two groups of wires: one to charge it (white connector) and the other to supply the PS (red connector).
We removed the red connector (JST-RCY) which is shown in Figure 3.16 because the wires may not withstand many flights.

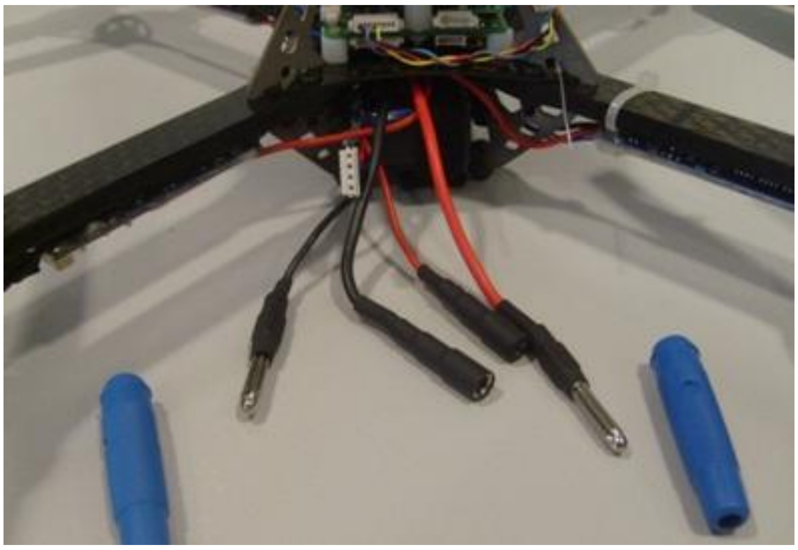


**Figure 3.17** Connectors between Battery and Power Supply

In Figure 3.17 the two thinner wires (red-female and black-male) which are the wires from the battery, can be seen. So we removed the JST-RCY connector and the wires and we changed to these wires with banana connectors.
The other two banana wires are soldered onto the Power Supply, and they have the correct size to fit the connectors properly.
It is necessary to connect the two red wires (supply) and the two black (GND). Be careful doing this last step so as not fry the board due to a short circuit.

## 3.6 Power Supply – Controllers

There are 4 controllers on the quadrotor, one for each engine.

All of them must be supplied from the PS and if it is working properly and is well supplied, a green solid LED should be on when turning on the Main Board. As the arrow indicates below, there are two wires that connect each engine with the Power Supply: the red wire, which is the supply, and the black wire which is the Ground.

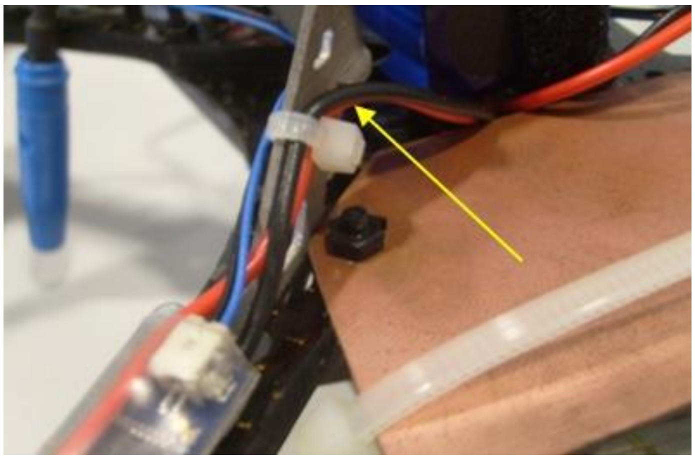


**Figure 3.18** The yellow arrow shows where are the black and red wires

These two wires are already soldered onto the controller when it is purchased.
The 8 wires from the controllers are soldered as shown in Figure 3.19:

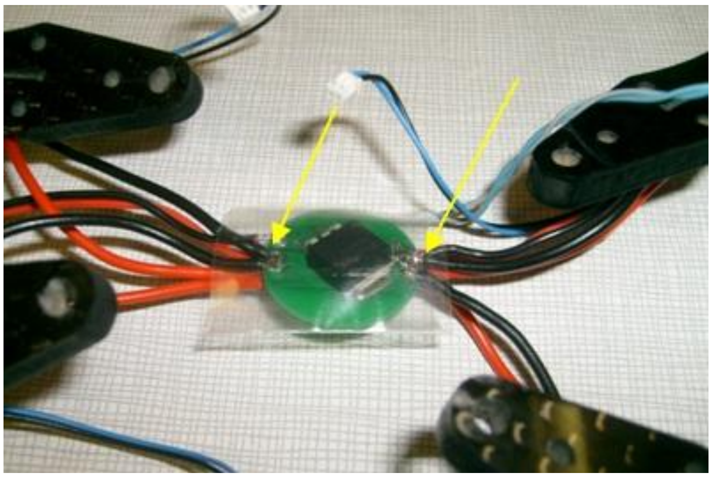


**Figure 3.19** It shows where to solder the wires on the Power Supply

Remember that the supply and the GND wires are soldered on different sides of the PS. The yellow arrows indicate the GND wires, but on the other side, the supply wires for the controllers must be also soldered.

## 3.7 Controller - Engine

In this paragraph a brief description of the integration between the controller and the engines will be explained.

There are 3 wires to connect the controller with the engine. To connect them, it is necessary to solder the 3 wires of the engine to the three wires of the controller (see Figure 3.20), but before doing this, we must prepare the heat shrink tub and leave them with the wire inside in order to heat them later.
The connection should be similar to what is shown below:

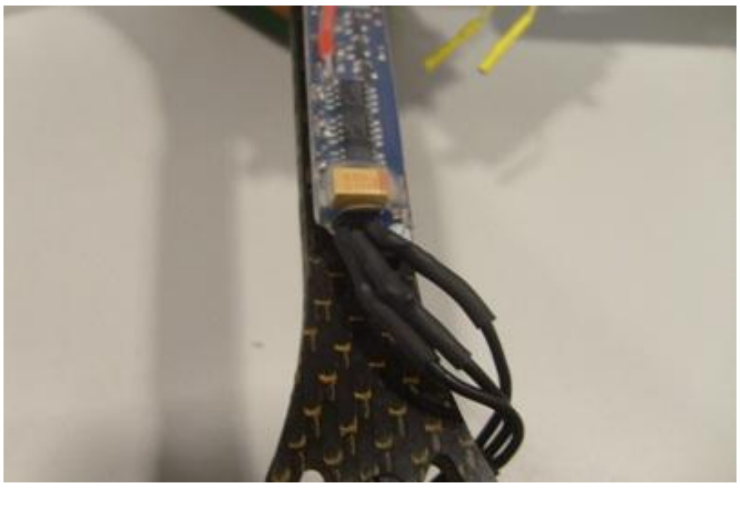


**Figure 3.20** Connection between the Controllers and the Engines

The wires are included when you purchase the engine and the controllers.
The soldering order of the wires is not important.

*Note: The XBDL Controllers are specially designed to work with the X-BL 52s engines, so if the engines are changed, the behaviour of the engine must be tested*.

## 3.8 Encoder - Receiver

As explained before, our intention was to use the encoder ATMEGA328 provided by PPZUAV, but due to various problems, in the end, we used the MeekPE encoder provided by Paparazzi developers during our stay in Toulouse.

### 3.8.1 ATMEGA328 Encoder Board

If the ATMEGA328 encoder is purchased in future, the wires must be soldered as shown in Figure 3.21.

To connect the encoder with the receiver, we have to solder (or plug in) 9 wires (circled in red as shown in Figure 3.21).
The black wire soldered on the corner is the ground signal. The red wire next to the ground signal will supply the receiver automatically.
The other 7 wires are the channel inputs from the receiver that the encoder will multiplex.

The yellow circle highlights the jumper which must be in place if the receiver, the servos and the encoder are supplied from the Main Board. If the servos are supplied independently by an external battery instead of from the Main, the jumper must be removed.

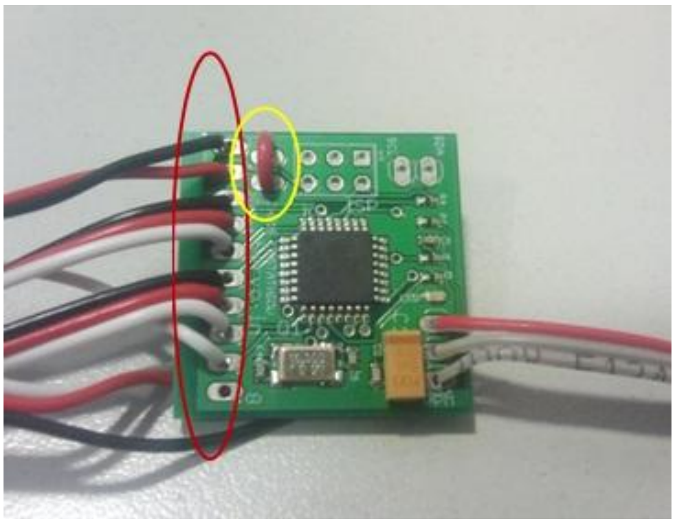


**Figure 3.21** Connection between Receiver and the Encoder ATMEGA328

On the other end, the connection between the 9 wires (GND, 5V and the 7 channels) using the proper connector and the receiver is shown below:

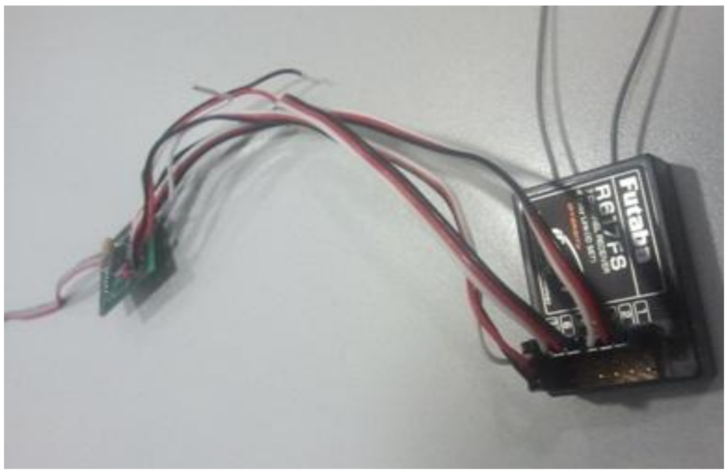


**Figure 3.22** How to plug the wires into the Receiver

Using Figure 3.23 it may be easier to understand how to connect all the wires:

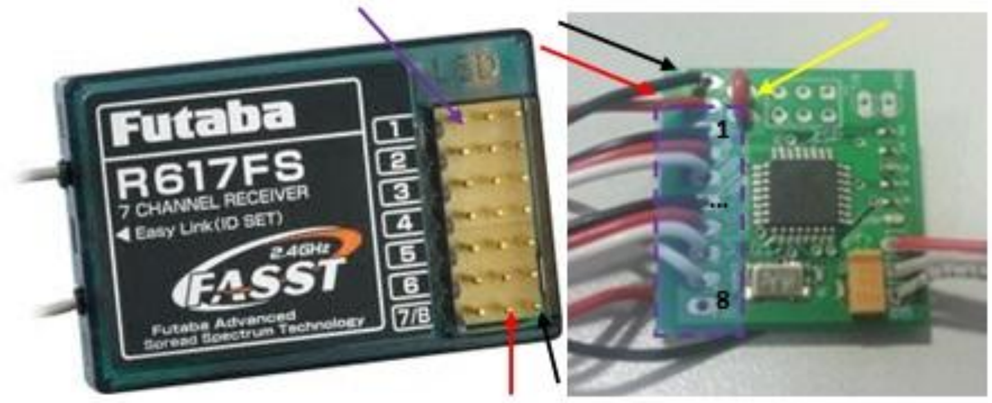


**Figure 3.23** Accurate image of the connection

The yellow arrow indicates the Jumper (red wire) which has to be in place because we supply the servos from the Main Board.
The black arrow points to the black wire which is the ground. It must be soldered onto the encoder board, and plugged into the receiver where the arrow indicates. This is the same for the supply wire. Although we recommend connecting the GND and supply wires where the arrows indicate, you can connect wherever you want using the same order (GND outermost line and Supply in the middle row). The purple dashed rectangle shows the 8 available channels from the encoder, from 1 up to 8. Our receiver only has 7 channels so the last one will be free. To connect to the receiver, just plug in at the correct number.

### 3.8.2 MeekPE Encoder Board

Due to problems with the other encoder board, we used the MeekPE encoder Board for the project. You can see the integration between the encoder Board and the Receiver MeekPE in Figure 3.24:

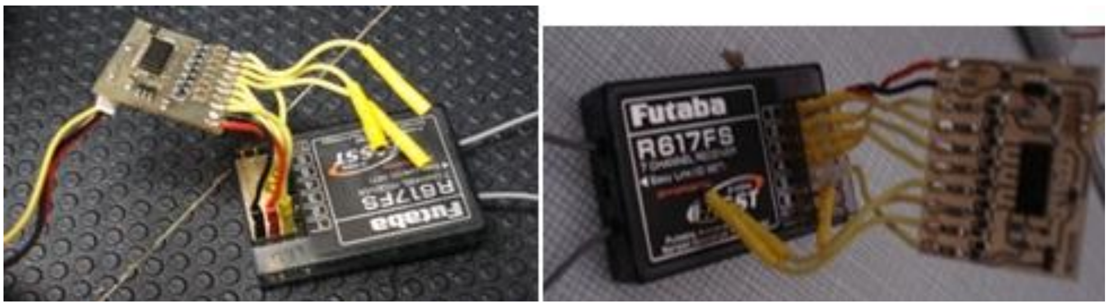


**Figure 3.24** Proper connection of the Receiver with the MeekPE Encoder

The connection of this Encoder board is different from the other board as explained previously. We are going to use 5 channels, so as you can see in the image on the right, we connect the 4 yellow wires and the white wire. The white wire is used in order to inform the encoder that the signal from this wire is the last. In Figure 3.24 left, you can see the supply (red) and GND (black) wires connected.

*Note: Remember to set the radio.xml up correctly if you are using this encoder board. This is further detailed in chapter Radio xml.*
.

## 3.9 Controllers – Main Board

There are two wires connecting each controller with the Main Board.
The wires came with the controller and we had to change the connectors because they were set up so as to be plugged into the AscTec Autopilot.
The connector we used was the same as for the Power Supply wire, a Molex Picoblade consisting of 2 pins.
In Figure 3.25 you can see the connection with the AscTec controller and the two wires (blue and black).

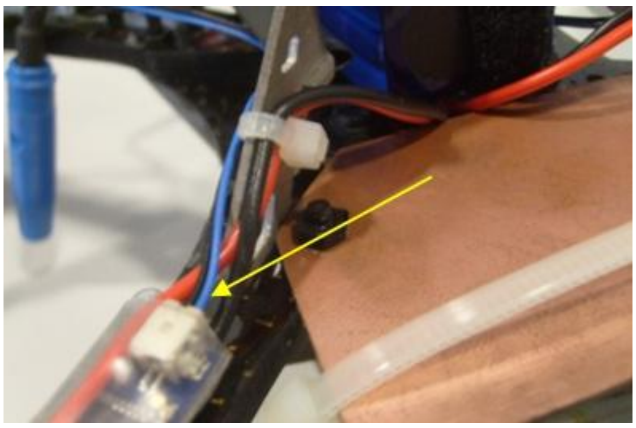


**Figure 3.25** Yellow arrow shows the black and blue wires

Below, the connection of the wires to the Main Board is indicated by the circle as can be seen in Figure 3.26.
The wires in the upper side of the Main Board are blue-black from left to right, and the opposite way on the lower side of the Main Board.

On the other side of the Main Board there are 2 plus connectors for the other two engines. The order is the same as mentioned above; in the upper side it is blue-black from left to right, and the opposite way on the lower side of the Main Board.

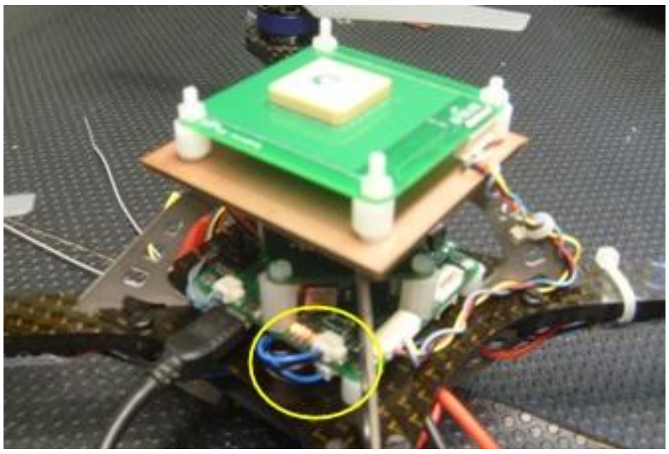


**Figure 3.26** Location of the Controllers connection in the Main Board

# CHAPTER 4. SOFTWARE CONFIGURATION

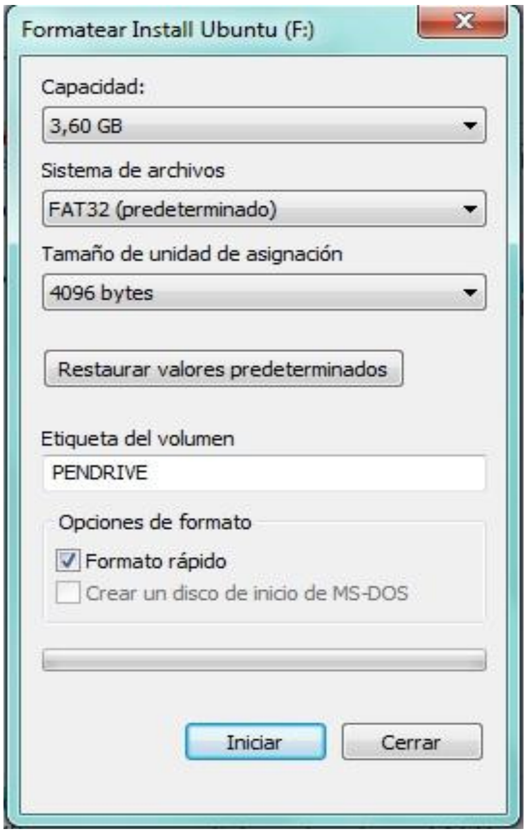
Now that we have seen the physical systems and how they are integrated, it is now time to take a look at the software that will make it possible for the quadrotor to fly.

## 4.1 Linux Setup

In this chapter all the steps needed to get the Paparazzi Software running on any computer from scratch will be described.

The first step is to get a Linux OS into a USB which will permit us to fly the quadrotor wherever we wish with only this USB and a computer.

Firstly, it is necessary to have an empty USB stick on which to install the Linux OS.
The Paparazzi website states that the USB must have at least 2 GB.
Bearing in mind that a USB of this size gave problems during the project, I decided to use a USB stick with a larger capacity of 4 GB.
The best option is to first format the USB stick before starting the installation as show in Figure 4.27:


Formatear Install Ubuntu (F:)

Capacidad: 3,60 GB

Sistema de archivos: FAT32 (predeterminado)

Tamaño de unidad de asignación: 4096 bytes

Etiqueta del volumen: PENDRIVE

Opciones de formato

- [✓] Formato rápido
- [ ] Crear un disco de inicio de MS-DOS

**Figure 4.27** Features to format the USB stick

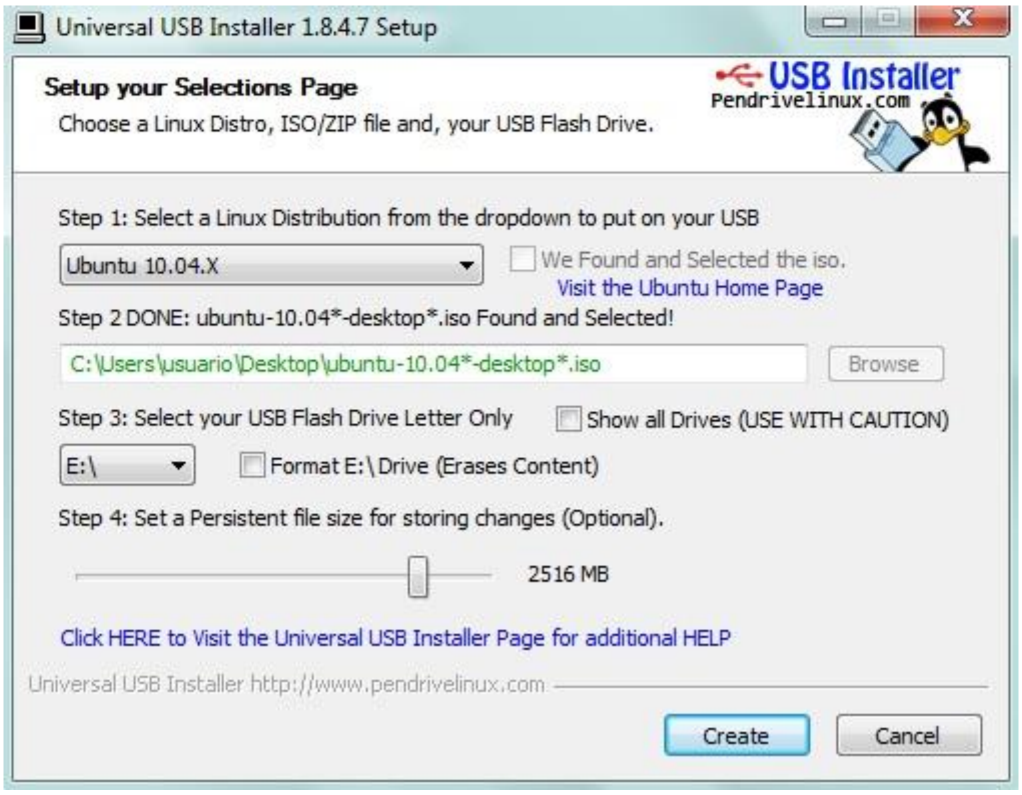
In order to open the window as shown above, all that is required is to right-click the USB icon that appears on the PC and click format. The default features are correct for what we need, so we can carry out the format process.

Once done, it is necessary to download the Linux 10.04 LTS (Lucid) iso.
This is easily downloaded from the Ubuntu website making sure that we download either the 32 bit or the 64 bit version. [10]

*Note: Note down the folder where the iso is downloaded. You will need the root in the next step.*

When the iso is downloaded we have to use the Universal USB installer (this can be easily obtained also from the Ubuntu website or by way of any search engine) to install the iso onto the USB stick correctly. [10]

Once the USB installer has been installed (in this project the version 1.8.4.7 is used) we have to carry out a custom setup for our needs. When executing, it should be similar to Figure 4.28:


Universal USB Installer 1.8.4.7 Setup

Setup your Selections Page
Choose a Linux Distro, ISO/ZIP file and, your USB Flash Drive.

Step 1: Select a Linux Distribution from the dropdown to put on your USB
Ubuntu 10.04.X
We Found and Selected the iso.
Visit the Ubuntu Home Page
Step 2 DONE: ubuntu-10.04*-desktop*.iso Found and Selected!
C:\Users\usuario\Desktop\ubuntu-10.04*-desktop*.iso
Step 3: Select your USB Flash Drive Letter Only
Show all Drives (USE WITH CAUTION)
E:\
Format E:\ Drive (Erases Content)
Step 4: Set a Persistent file size for storing changes (Optional).
2516 MB
Click HERE to Visit the Universal USB Installer Page for additional HELP
Universal USB Installer http://www.pendrivelinux.com

**Figure 4.28** USB installer screenshot

Step 1: Select which version is to be installed (10.04 LTS).

Step 2: Detail the root of the iso in the computer.

Step 3: Check which port the USB stick is plugged into.

Step 4: Maintain a minimum free space within the file in order to store changes.

*Note: The size in Step 4 will be different depending on the total size of the USB stick used in the project.*

This process can take several minutes.

When it is finished, the Ubuntu installation on the USB stick is ready to be booted.

## 4.1 USB Boot

One of the goals of this project is, as mentioned previously, to develop a simple way to fly the quadrotor in any given location. For this reason, how to download and install the Linux OS and the Paparazzi software onto a USB stick have been detailed previously.

But one more step is needed in order to boot the USB when a computer is turned on.
It is necessary to correctly set the BIOS up. To do this, the USB stick has to be plugged into the USB port before launching the laptop.

When the laptop begins to launch, it is important to be attentive and read the messages that appear (usually in the lowest part of the screen), since this is where it will be stated which key must be pressed in order to enter into the BIOS.
It is usually F1, F2, F8, F10 or F12, but this depends on each computer.

On the laptop on which this project has been carried out the key is F2.
It is important to press this key repeatedly and quickly when we discover which one will launch the BIOS.

Once in the BIOS, following the instructions, the tab BOOT must be selected.
A list with the HDD (Hard Disk Device), also maybe a FDD (Floppy Disk Device) and the USB appears. We have to change the order of this list such that the USB is in the first position of the list.

Each laptop allows us to move items upward and downward in a different way; with the arrow keys, with F5 and F6 or another key as indicated.

When the USB is in the first position, the changes must be saved and exit from BIOS. Then reboot the laptop without unplugging the USB and it should automatically boot the USB stick.

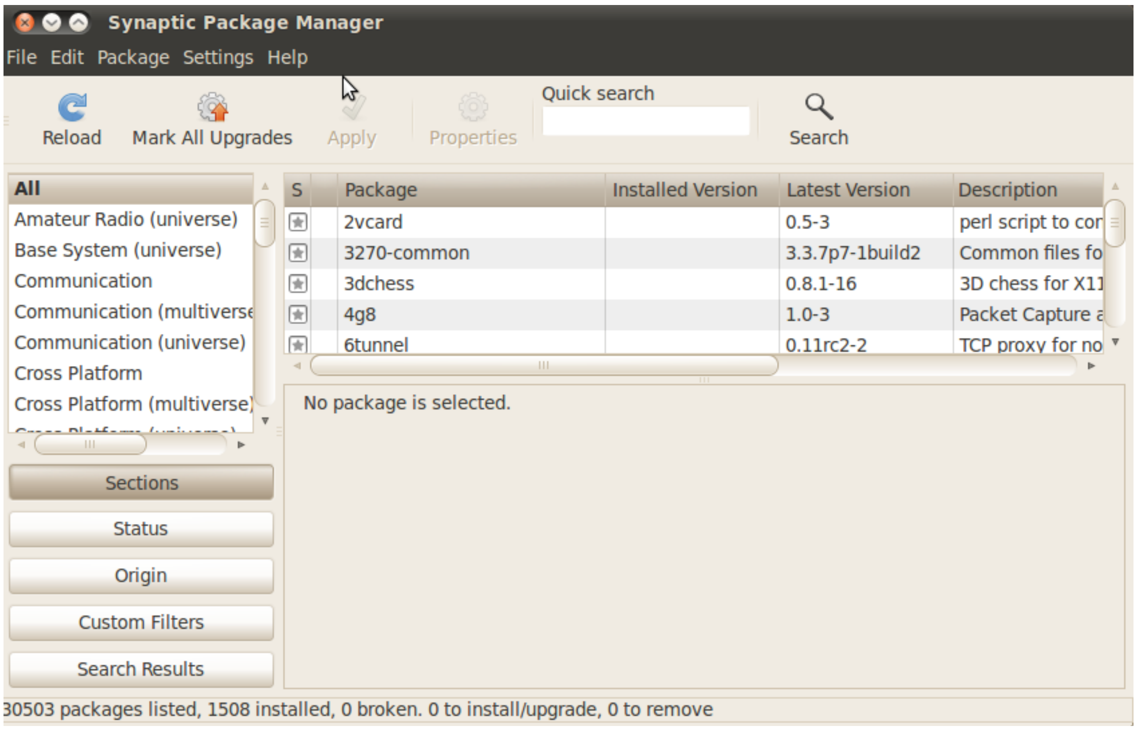
In some laptops, once the USB is unplugged (of course when the laptop is turned off) the computer loses the Boot changes and it is necessary to set it again as explained above.

## 4.2 Paparazzi Setup

Once Ubuntu is correctly installed onto the USB stick, it is then time to detail how to obtain the Paparazzi software and prepare it in order to be launched.

To download the Paparazzi software we have to execute the Synaptic Package Manager (System/Administration).

Then, a window should appear. It should be similar to Figure 4.29.



| S | Package | Installed Version | Latest Version | Description |
|---|---|---|---|---|
| ☐ | 2vcard | | 0.5-3 | perl script to cor |
| ☐ | 3270-common | | 3.3.7p7-1build2 | Common files fo |
| ☐ | 3dchess | | 0.8.1-16 | 3D chess for X11 |
| ☐ | 4g8 | | 1.0-3 | Packet Capture a |
| ☐ | 6tunnel | | 0.11rc2-2 | TCP proxy for no |

No package is selected.

30503 packages listed, 1508 installed, 0 broken. 0 to install/upgrade, 0 to remove

**Figure 4.29** Synaptic Package Manager Window

Now, the APT line has to be added (Settings/ Repositories/ Software Sources/ Other software)

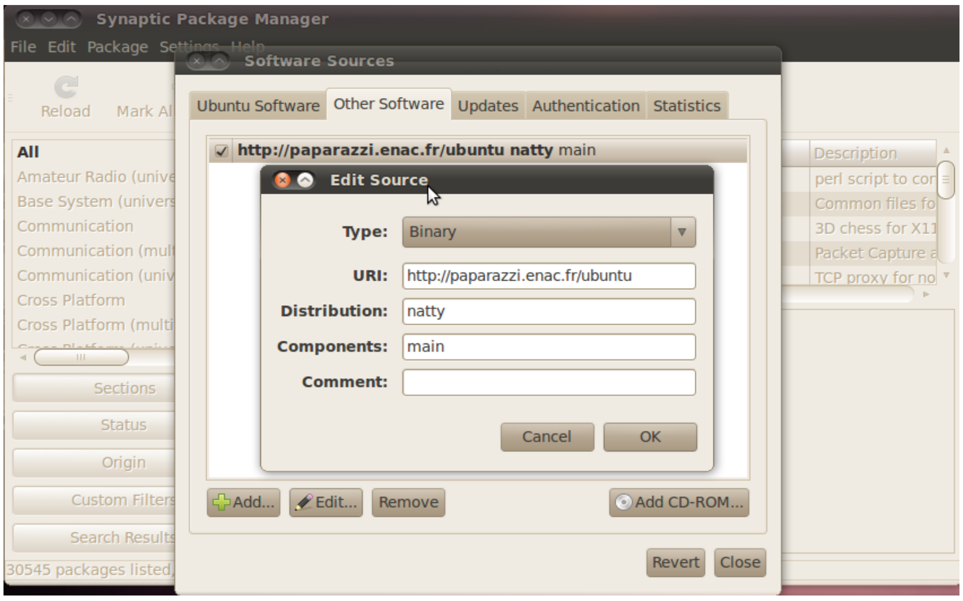
It is then necessary to click Add and insert the most recent APT line:

*deb http://paparazzi.enac.fr/ubuntu natty main*

Then click, Add source.

Initially, it can be seen that this APT line will not work for us since it is for the most recent version of Linux (11.04), so we have to edit and adapt it to our version.

Select the APT line inserted by us and click on the Edit button.
A window like the one in Figure 4.30 should appear.



http://paparazzi.enac.fr/ubuntu natty main

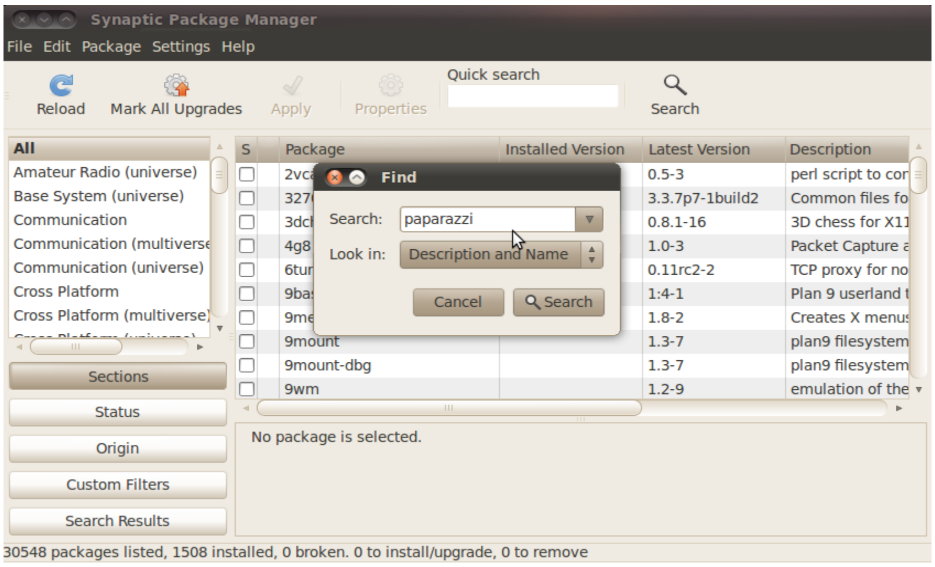
Type: Binary
URI: http://paparazzi.enac.fr/ubuntu
Distribution: natty
Components: main
Comment:

**Figure 4.30** Details of the APT line

As mentioned before, the Linux version used in this project is called *lucid*, so where it says *natty* it is necessary to write *lucid* in small letters.
Then we click OK and the window disappears. Then click Close.

When Linux detects that the APT line has been changed, it will inform us about it with a new window. The window will tell us that in order to update the code, it is necessary to click the Reload button located on the left hand-side of the Synaptic Package Manager screen.
So, click the Reload button and it will start updating the data.
When the reload has been successfully completed, we have to click the button with the magnifying glass (the quick search sometimes gives problems).

In Figure 4.31 you can see how the screen should look like at this step.



Search: paparazzi
Look in: Description and Name

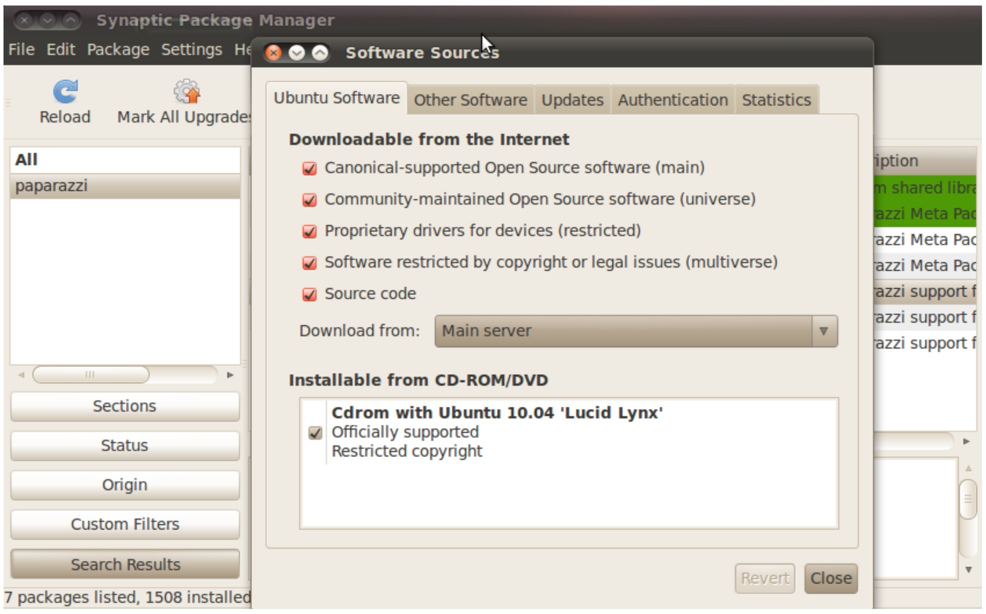
No package is selected.

30548 packages listed, 1508 installed, 0 broken. 0 to install/upgrade, 0 to remove

**Figure 4.31** Search Code Window

There we have to write paparazzi and click *search*.
It is important now to set the software sources which are located in
Synaptic Package Manager/ Settings/ Repositories
Select all the available sources as shown in Figure 4.32 and close the window.

*Note: It is important to select all the software sources, if not, some dependency problems will occur when trying to download the tools.*



**Downloadable from the Internet**

- [✓] Canonical-supported Open Source software (main)
- [✓] Community-maintained Open Source software (universe)
- [✓] Proprietary drivers for devices (restricted)
- [✓] Software restricted by copyright or legal issues (multiverse)
- [✓] Source code

Download from: Main server

**Installable from CD-ROM/DVD**

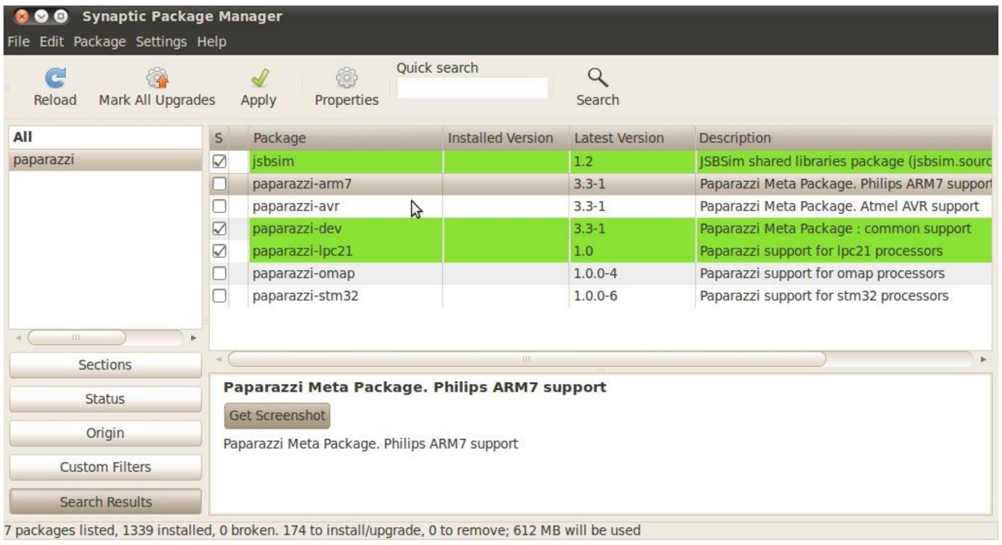
[✓] **Cdrom with Ubuntu 10.04 'Lucid Lynx'**
Officially supported
Restricted copyright

7 packages listed, 1508 installed

**Figure 4.32** Software sources

Then click the Reload button again.

It is not necessary to download all the Paparazzi tools available from the APT line, only the ones which are highlighted in green in Figure 4.33 as well as the arm7 package:



| S | Package | Installed Version | Latest Version | Description |
|---|---|---|---|---|
| ☑ | jsbsim | | 1.2 | JSBSim shared libraries package (jsbsim.sourc |
| ☐ | paparazzi-arm7 | | 3.3-1 | Paparazzi Meta Package. Philips ARM7 support |
| ☐ | paparazzi-avr | | 3.3-1 | Paparazzi Meta Package. Atmel AVR support |
| ☑ | paparazzi-dev | | 3.3-1 | Paparazzi Meta Package : common support |
| ☑ | paparazzi-lpc21 | | 1.0 | Paparazzi support for lpc21 processors |
| ☐ | paparazzi-omap | | 1.0.0-4 | Paparazzi support for omap processors |
| ☐ | paparazzi-stm32 | | 1.0.0-6 | Paparazzi support for stm32 processors |

**Paparazzi Meta Package. Philips ARM7 support**

Paparazzi Meta Package. Philips ARM7 support

7 packages listed, 1339 installed, 0 broken. 174 to install/upgrade, 0 to remove; 612 MB will be used

**Figure 4.33** Tools needed for downloading

JSBSIM is an open source FDM (Flight Dynamics Model). The FDM is essentially the physics/math model that simulates the movement of an aircraft.

The LPC21 is the microprocessor of the Booz Autopilot.

The dev package includes the C and OCaml compilers and XML handlers.

The arm7 should not be necessary but one of the packages that arm7 has is the multilib library. Without this library, an error appears when compiling. So I downloaded this too.
It may be useful to know that the multilib library will not be used.

Then we click apply and the download starts. It could take few minutes.

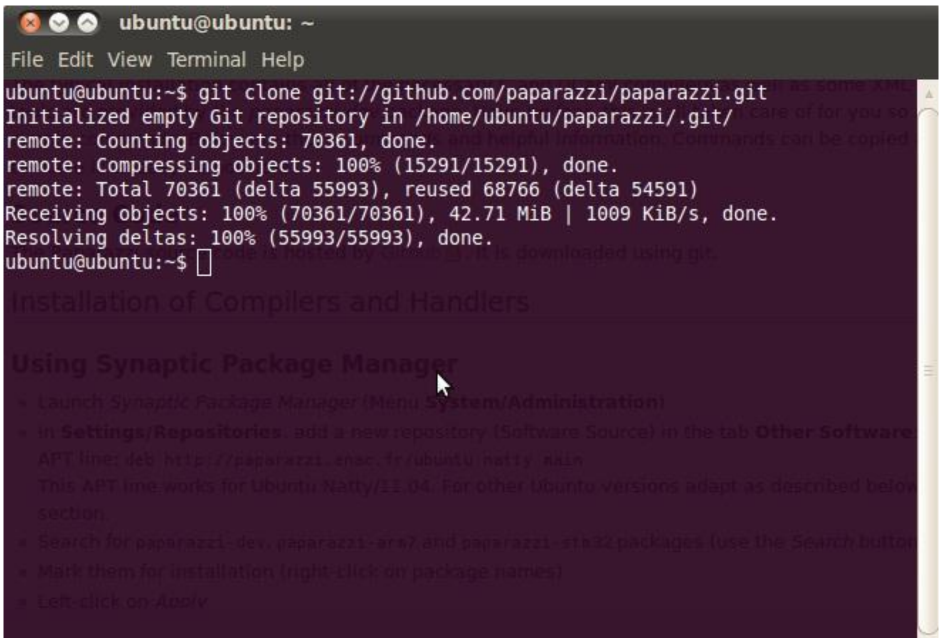
When it finishes, a new window appears saying that everything was fine.

If we complete the entire process and this window appears at the end, this means we have correctly downloaded the tools for the Paparazzi software.

Now we have to download the source code from GitHub Paparazzi repository.

To download the code, you have to open the Terminal (*Applications/ Accessories/ Terminal)*

Then you have to insert the first code line as shown in Figure 4.34 in order to obtain the code from the GitHub repository. Then click enter.



```
ubuntu@ubuntu:~$ git clone git://github.com/paparazzi/paparazzi.git
Initialized empty Git repository in /home/ubuntu/paparazzi/.git/
remote: Counting objects: 70361, done.
remote: Compressing objects: 100% (15291/15291), done.
remote: Total 70361 (delta 55993), reused 68766 (delta 54591)
Receiving objects: 100% (70361/70361), 42.71 MiB | 1009 KiB/s, done.
Resolving deltas: 100% (55993/55993), done.
ubuntu@ubuntu:~$ 
```

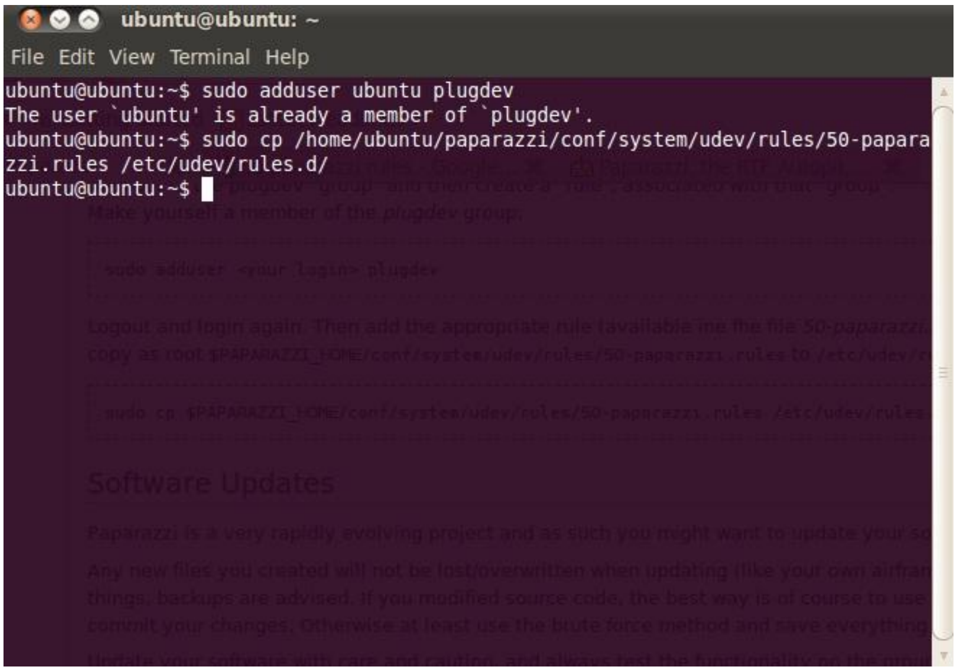
**Figure 4.34** GIT line to download the code

At this step, we have the tools and the source code. Now it is time to complete one of the final steps.

In order to flash the Paparazzi Boards directly through the USB it is necessary to establish the *50 paparazzi rules* line.
The paparazzi rules are required for flashing, without them, we may have problems flashing the firmware to the quadrotor.

As shown in Figure 4.35, it is necessary to add our user to the *plugdev* group if we are not the root user. In this case here, we are already a member.

Then you have to write the line to copy the rules from the paparazzi folder to the etc folder:

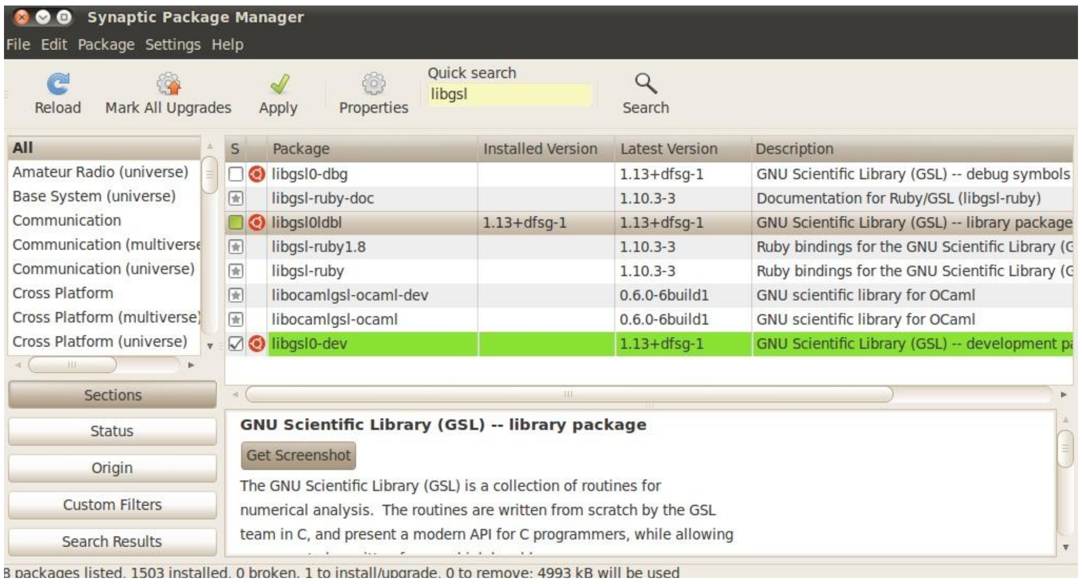


```
ubuntu@ubuntu:~$ sudo adduser ubuntu plugdev
The user `ubuntu' is already a member of `plugdev'.
ubuntu@ubuntu:~$ sudo cp /home/ubuntu/paparazzi/conf/system/udev/rules/50-papara
zzi.rules /etc/udev/rules.d/
ubuntu@ubuntu:~$ 
```

**Figure 4.35** 50 Paparazzi rules

Then it is necessary to go to the Terminal, write *cd paparazzi* intro, and then *make* to compile the firmware.

At this point I tried to start with an easy simulation, but in the Console an error appeared. It was related to the GSL (GNU Scientific Library), so I downloaded the package “libgslo-dev” from the Synaptic Manager (highlighted in green in Figure 4.36).



| S | Package | Installed Version | Latest Version | Description |
|---|---|---|---|---|
| ☐ | libgsl0-dbg | | 1.13+dfsg-1 | GNU Scientific Library (GSL) -- debug symbols |
| | libgsl-ruby-doc | | 1.10.3-3 | Documentation for Ruby/GSL (libgsl-ruby) |
| 🟩 | libgsl0ldbl | 1.13+dfsg-1 | 1.13+dfsg-1 | GNU Scientific Library (GSL) -- library package |
| | libgsl-ruby1.8 | | 1.10.3-3 | Ruby bindings for the GNU Scientific Library (G |
| | libgsl-ruby | | 1.10.3-3 | Ruby bindings for the GNU Scientific Library (G |
| | libocamlgsl-ocaml-dev | | 0.6.0-6build1 | GNU scientific library for OCaml |
| | libocamlgsl-ocaml | | 0.6.0-6build1 | GNU scientific library for OCaml |
| ☑ | libgsl0-dev | | 1.13+dfsg-1 | GNU Scientific Library (GSL) -- development pa |

**GNU Scientific Library (GSL) -- library package**

The GNU Scientific Library (GSL) is a collection of routines for numerical analysis. The routines are written from scratch by the GSL team in C, and present a modern API for C programmers, while allowing

8 packages listed, 1503 installed, 0 broken. 1 to install/upgrade, 0 to remove; 4993 kB will be used

**Figure 4.36** Downloading the GSL Package

Once complete, a message appears informing us that the download has been successful.

Now you should be able to compile the files without any problems and correctly carry out the simulation using the files detailed in chapter: XML Settings for Simulation.

The Paparazzi folder will be located in the Home Folder.

## 4.3 Launcher

It may be more comfortable for the user if a launcher is installed on the desktop.

To set it, first it is necessary to download this pic:

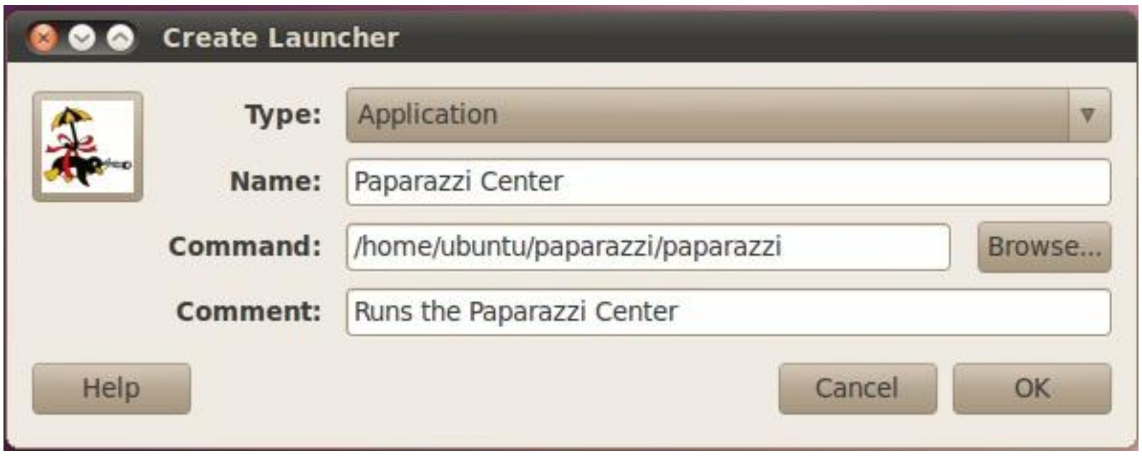
*http://paparazzi.enac.fr/wiki_images/Paparazzi_logo.png*

Then you make a right-click on the desktop and select “Create Launcher”.

The user has to set the features as showed in Figure 4.37.

Bear in mind that where it says Command, this is the root of the launcher so it could be different in each laptop.

The image logo displayed on the upper left-hand corner can be modified with a double click on the image and selecting the image that will be used as an icon.


Create Launcher

Type: Application

Name: Paparazzi Center

Command: /home/ubuntu/paparazzi/paparazzi

Comment: Runs the Paparazzi Center

**Figure 4.37** Settings to set up a Launcher on the Desktop

## 4.4 XML Settings for Simulation

Nowadays, there are a lot of programming codes: C++, C#, Java, FORTRAN, xml… The Paparazzi files used in this project are mainly in xml.

As expected, some changes in the files were necessary in order to correctly set the flight up.

In this chapter I will detail which xml files must be modified to get a proper simulation. The changes in the xml files will be detailed step by step.

*Note: Remember that if you want to take a look at the entire xml files are in ANEX C.*

### 4.4.1 Airframe xml

The airframe file used for simulation will be the *Booz2_a1.xml*.

The *Booz2_a1.xml* used in the simulation will be almost the same as the default one available on the GitHub repository.
Only one change in the modes in order to begin simulating in NAV will be carried out. These will be left as:

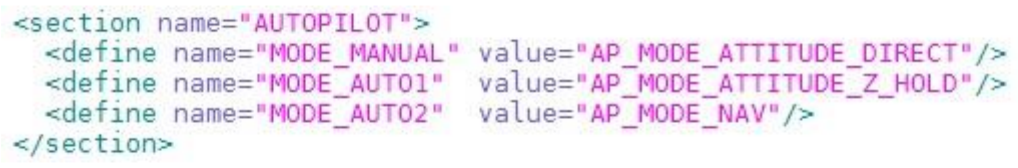

```
<section name="AUTOPILOT">
  <define name="MODE_MANUAL"  value="AP_MODE_ATTITUDE_DIRECT"/>
  <define name="MODE_AUTO1"   value="AP_MODE_ATTITUDE_Z_HOLD"/>
  <define name="MODE_AUTO2"   value="AP_MODE_NAV"/>
</section>
```

**Figure 4.38** Booz2_a1 modes for Simulation

### 4.4.2 Flight Plan

The FP (Flight Plan) is the most edited file in the project. It is very easy to understand how it works due to its simple structure. This makes it incredibly easy to edit the file since you have free reign to simulate almost everything such as shape, size etc. and at almost any location by simply editing the coordinates.

Our FP will be called EETAC due to university faculty name.

The Flight Plan is divided into different parts.

First of all, there are the general settings.
There you can set the maximum altitude that you want to simulate, the geodetic coordinates to center the FP (these coordinates will represent the *central point*), the security height when the aircraft enters in an exception, the maximum distance from the central point, etc.

Then you create the location of the waypoints. The best thing about these is that their position is referenced to the central point, so if you change the geodetic coordinates of the FP because you want to fly at another location, the waypoints move with the central point, and maintain the distances already set.

Finally you have to select the block that it must complete.
The blocks are the orders it has to follow. For example one block could go from waypoint 1 to waypoint 7, crossing first waypoint 5.
When one block has been completed, then the aircraft carries out the next one. Only in the event that it is a “stay waypoint” or a “stay home”, does it stay in this place.
A lot of shapes could be set in the blocks; circles, figures of 8 between two waypoints, straight lines between two waypoints…
It is worthwhile visiting the Flight Plans section in the website, where you can see all the different possibilities for a FP.

As all the Flight Plan was done from scratch, the entire file is in the ANEX C.

I recommend to enjoy the simulation editing the current FP and seeing what things can be carried out with the quadrotor in the future.

*Note: At least one waypoint must be named HOME for the failsafe procedure.*

### 4.4.3 Reset00 xml

The most important thing to detail now is how we setup the coordinates for the flight.

There are two files where coordinates must be set up.
One, as you have seen, is on the Flight Plan xml where there are two coordinate values: the longitude and the latitude (which determines the central point). As seen before, the waypoints that are located on the map, are refered to the central point (not detailing their longitude and latitude).

There is also another file where we have to modify the coordinates and in fact, this is more important than the coordinates of the Flight Plan.

This file is the *reset00.xml* file.

This file is requested by the airframe file (Booz2_a1) as its initial conditions.
It determines where the simulation will be carried out, so the landscape that you see on the Ground Control Station will depend on these coordinates.

The coordinates that the reset00 file by default has are from Muret, a town close to Toulouse. So if you simulate the Microjet (which is another A/C) or the Booz2_a1 without the correct set up of the coordinates, you will simulate in Muret instead of the EETAC campus.
It is very important to say that in the reset00 file the coordinates are in the geocentric system instead of WGS84 as in the FP.
In order to correctly simulate in the EETAC campus plain, I had to edit the coordinates and leave them as the reset00 file in the ANEX C.

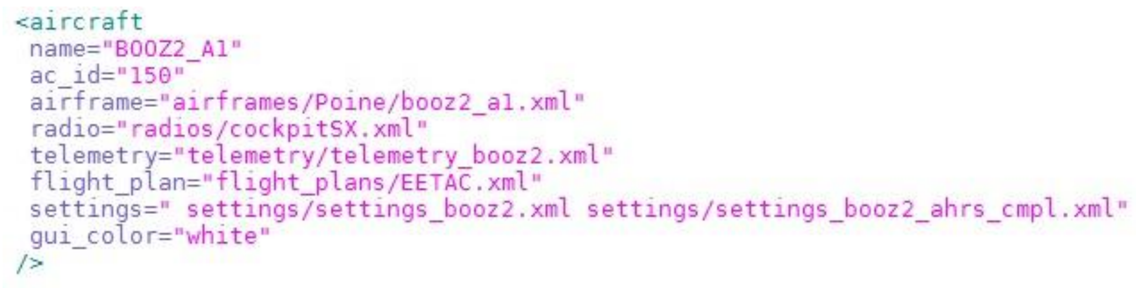
It is also noteworthy, that if you do not change the coordinates to the EETAC campus in the Flight Plan when you launch the simulation, the Ground Station detects by itself the difference between the Flight Plan and the reset00.
So by itself, it moves the flight plan from wherever it has been located to the *central point* detailed in the reset00.
The waypoints will refer to this point instead of the Flight Plan coordinates.
In this file you can also set wind direction and speed.

In AP flight the file reset00 will not be used. This is because it takes the coordinates from the GPS or if it is not possible to obtain the coordinates from GPS, they are obtained from the FP.

### 4.4.4 Conf xml

We will use the Booz2_a1, because it was recommended by Paparazzi developers as the most appropriate aircraft for quadrotors that use Booz Main Board.

So in this file, in the aircraft booz2_a1, we have to change the flight plan from the dummy.xml to the one we created called EETAC.xml, so the aircraft block is left as shown in Figure 4.39.



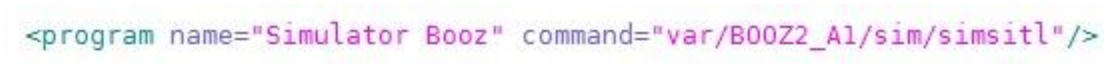
```
<aircraft
 name="BOOZ2_A1"
 ac_id="150"
 airframe="airframes/Poine/booz2_a1.xml"
 radio="radios/cockpitSX.xml"
 telemetry="telemetry/telemetry_booz2.xml"
 flight_plan="flight_plans/EETAC.xml"
 settings=" settings/settings_booz2.xml settings/settings_booz2_ahrs_cmpl.xml"
 gui_color="white"
/>
```

**Figure 4.39** Booz2_a1 settings for Simulation

Now, the flight plan launched will be the one that we have created before.

### 4.4.5 Control_panel xml

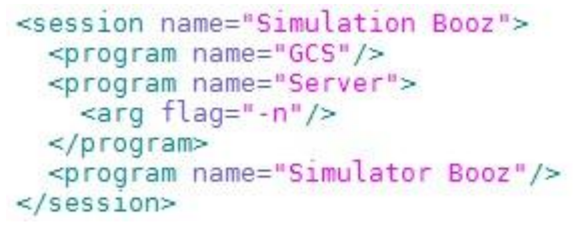
In the Conf panel we have to create the running agent “Simulator Booz”.
The default one is “Simulator” but it is simulated using launchsitl, so instead of changing the default one, I preferred to create a new one with the proper setup:



```
<program name="Simulator Booz" command="var/BOOZ2_A1/sim/simsitl"/>
```

**Figure 4.40** Program created for Simulation

Once complete, we have to create the next session to correctly simulate the flight:



```
<session name="Simulation Booz">
  <program name="GCS"/>
  <program name="Server">
    <arg flag="-n"/>
  </program>
  <program name="Simulator Booz"/>
</session>
```

**Figure 4.41** Session created for Simulation

You see that one of the programs, Simulator Booz, is the one that was created in the previous step.

Then when we go to the Paparazzi Center we can select the A/C Booz2_A1, the target sim and the session Simulation Booz.

The simulation should then start.

## 4.5 XML Settings for AP flight

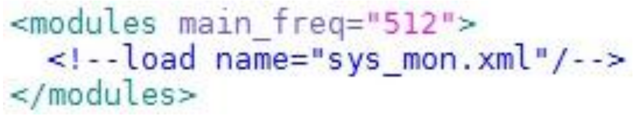
Once seen the files used for simulation, let’s see now the files for AP flight:

### 4.5.1 Airframe xml

For the Booz Main Board there are several airframe files, so we base our work on the file that was recommended on the Paparazzi website for AscTec quadrotors.
The airframe xml used as a starting point is called Booz2_g1.

First of all you have to save the xml with a new name (UPC was used in this project) to do all the changes without reversing the configuration of the Booz2_g1 file.
Then let’s start with the correct set up of the new file.

First, it is necessary to remove almost all of the modules only leaving the main_freq.

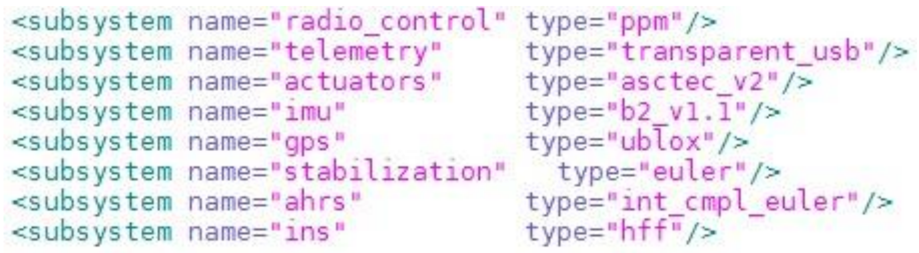


```
<modules main_freq="512">
  <!--load name="sys_mon.xml"/-->
</modules>
```

**Figure 4.42** Frequency module

The next features to be configured are the telemetry types and actuators subsystems.
The telemetry subsystem says *transparent* by default. This means that the Datalink will be through Xbee.
We have to change it correctly to have the Datalink through the USB. So instead of *transparent*, we must write *transparent_usb*.
Also, the AscTec actuators have to be set correctly and the actuators subsystem type is left as *asctec_v2* because of the version.
Finally, as we will use the default baud rate (57600), it is not necessary to keep the value on the airframe file, so we remove this from the GPS subsystem.

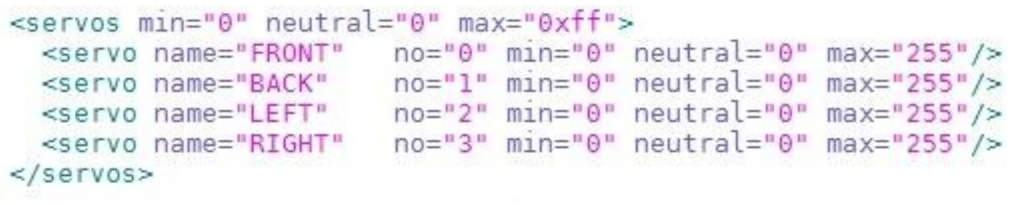
The group of subsystems explained above should be left as:



```
<subsystem name="radio_control" type="ppm"/>
<subsystem name="telemetry"     type="transparent_usb"/>
<subsystem name="actuators"     type="asctec_v2"/>
<subsystem name="imu"           type="b2_v1.1"/>
<subsystem name="gps"           type="ublox"/>
<subsystem name="stabilization"   type="euler"/>
<subsystem name="ahrs"          type="int_cmpl_euler"/>
<subsystem name="ins"           type="hff"/>
```

**Figure 4.43** Subsytems code

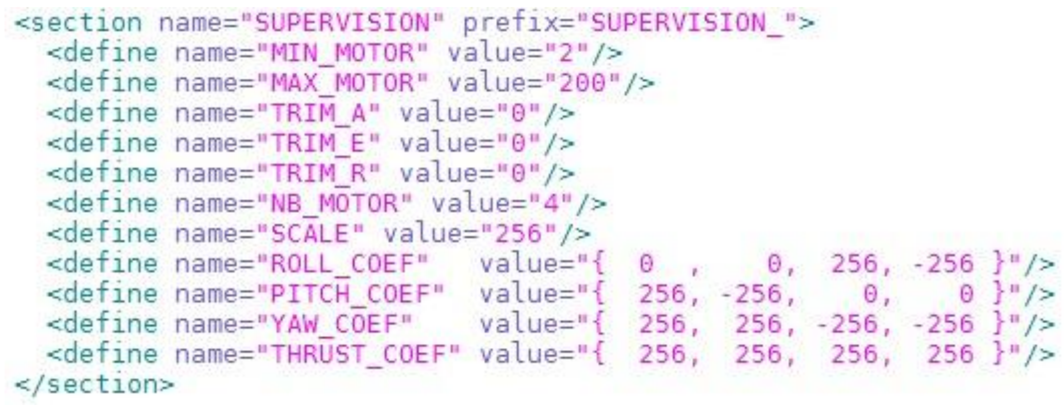
Now let’s define the engines of the quadrotor in a simpler way as are detailed by default. This change will make their management easier for us.
In Figure 4.44 you can see that the modified block refers to the engines:



```
<servos min="0" neutral="0" max="0xff">
  <servo name="FRONT"   no="0" min="0" neutral="0" max="255"/>
  <servo name="BACK"    no="1" min="0" neutral="0" max="255"/>
  <servo name="LEFT"    no="2" min="0" neutral="0" max="255"/>
  <servo name="RIGHT"   no="3" min="0" neutral="0" max="255"/>
</servos>
```

**Figure 4.44** Engine code details

The next section to be modified is the Supervision, where we have to add several xml lines which are required to be set up as follows:



```
<section name="SUPERVISION" prefix="SUPERVISION_">
  <define name="MIN_MOTOR" value="2"/>
  <define name="MAX_MOTOR" value="200"/>
  <define name="TRIM_A" value="0"/>
  <define name="TRIM_E" value="0"/>
  <define name="TRIM_R" value="0"/>
  <define name="NB_MOTOR" value="4"/>
  <define name="SCALE" value="256"/>
  <define name="ROLL_COEF"   value="{  0  ,    0,  256, -256 }"/>
  <define name="PITCH_COEF"  value="{  256, -256,    0,    0 }"/>
  <define name="YAW_COEF"    value="{  256,  256, -256, -256 }"/>
  <define name="THRUST_COEF" value="{  256,  256,  256,  256 }"/>
</section>
```

**Figure 4.45** Supervision Section

If you prefer, you can copy the code shown above from the file booz2_a1 file, where it is correctly set up, instead of writing the lines yourself.

*Note: The value of TRIM_E in Booz2_g1 file is 6 by default and MAX_MOTOR is 210, remember to change them to 0 and 200 respectively if you fix the code manually instead of copying from the booz2_a1.*

#### *4.5.1.1 IMU Calibration*

The IMU used in this project was purchased by PPZUAV and was originally sent from United States of America to Spain.
Also, it has been in the laboratory for a long time.

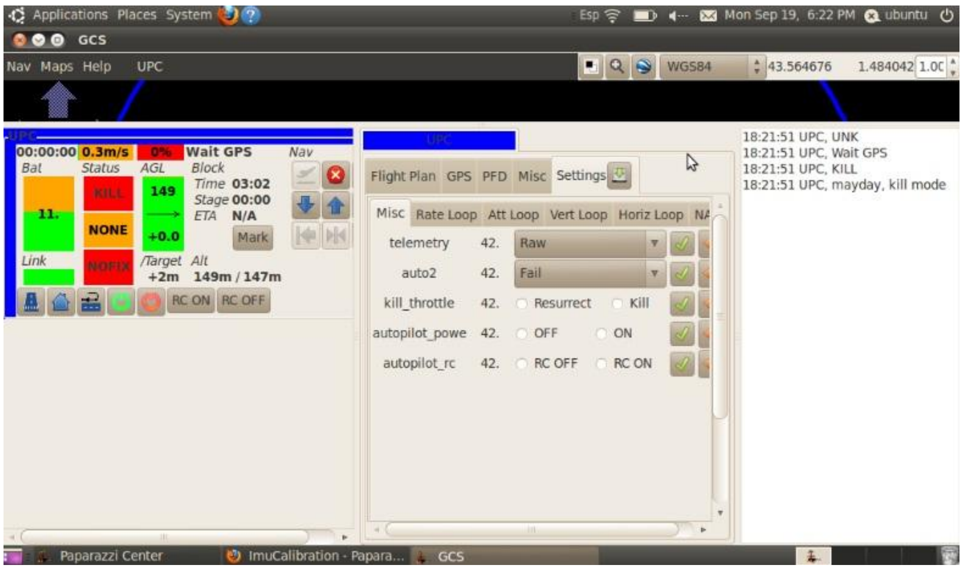
For this reason it was essential to recalibrate the IMU before flight.
At this step of the UPC.xml, we have to set the correct values for the IMU, so let’s first look at how to calibrate it:

First of all it is necessary to launch the Paparazzi Center.

Once complete, you have to verify that the A/C and their corresponding xml files are set up so as to fly the quadrotor (not for Simulation). Therefore, the files must remain the same and with the same settings as if you were going to carry out a flight, as in Figure 5.64.

As always, click on clean, build and the upload buttons in this order.
Once finished, click the execute button.
If everything has been properly set up; the GCS, Server and Datalink agents should be launched in the Paparazzi Center and the GCS should appear in a new window.

In the GCS, go to the Settings tab. Then click on Misc and in the Telemetry drop-down menu, select “raw” data and then click the green tick which is on the right. This will provide us with the current data that the IMU is generating.
In Figure 4.46 you can see in the middle the Telemetry location, where the Raw data is and the green tick on the right of its drop-down list.


Applications Places System
GCS
Nav Maps Help UPC
WGS84 43.564676 1.484042
Flight Plan GPS PFD Misc Settings
Misc Rate Loop Att Loop Vert Loop Horiz Loop
Paparazzi Center ImuCalibration - Papara... GCS

```
18:21:51 UPC, UNK
18:21:51 UPC, Wait GPS
18:21:51 UPC, KILL
18:21:51 UPC, mayday, kill mode
```

| | | | |
|---|---|---|---|
| telemetry | 42. | Raw | |
| auto2 | 42. | Fail | |
| kill_throttle | 42. | Resurrect | Kill |
| autopilot_powe | 42. | OFF | ON |
| autopilot_rc | 42. | RC OFF | RC ON |

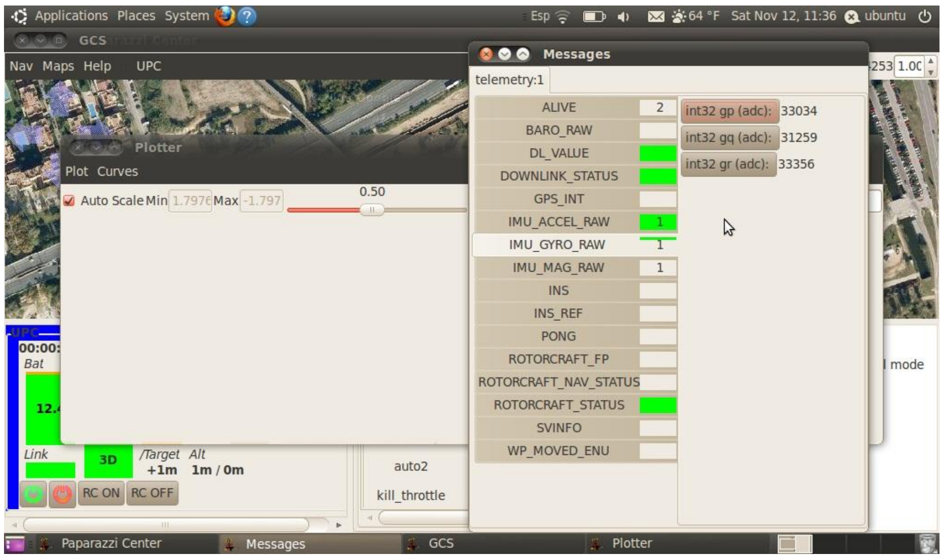
**Figure 4.46** GCS settings for raw data

Once the green tick has been clicked, the 42 located to the right of "telemetry" in Figure 4.46 should change to Raw.
If not, you have not correctly set the GCS to get the raw data.

If you start receiving the raw data, go to the Paparazzi Center. There, click on Tools, and click the Real Time Plotter (a window will appear) and then again click on Tools and now select Messages.
The plotter will display the data that we want to check or control.

The Messages window will display the subsystems values.
We will be able to select the data that we want to check.

At this step the screen should be similar to that shown in Figure 4.47.


Applications Places System
GCS
Nav Maps Help UPC
Plotter
Plot Curves
Auto Scale Min 1.797 Max -1.797
0.50
Messages
telemetry:1
Paparazzi Center Messages GCS Plotter

| | |
|---|---|
| ALIVE | 2 |
| BARO_RAW | |
| DL_VALUE | |
| DOWNLINK_STATUS | |
| GPS_INT | |
| IMU_ACCEL_RAW | 1 |
| IMU_GYRO_RAW | 1 |
| IMU_MAG_RAW | 1 |
| INS | |
| INS_REF | |
| PONG | |
| ROTORCRAFT_FP | |
| ROTORCRAFT_NAV_STATUS | |
| ROTORCRAFT_STATUS | |
| SVINFO | |
| WP_MOVED_ENU | |

| | |
|---|---|
| int32 gp (adc): | 33034 |
| int32 gq (adc): | 31259 |
| int32 gr (adc): | 33356 |

**Figure 4.47** Paparazzi Center with Plotter and Messages windows selected

The Real Time Plotter does not display the values by default, because we have to first select which data we want to review.

We will need to calibrate the gyros, so we select the IMU_GYRO_RAW button. As in Figure 4.47, three different kind of datum will be provided by the IMU_GYRO_RAW. Each one corresponds to an axis.

To display the data you have to drag and drop each button on the Plotter window. So select one (for instance: int32 gp), drag it to the Plotter window and drop there. You have to do the same for all three axes.
The values will be displayed on the Plotter.

We can add some lines to find out more accurately the exact value of each axis. We can guess the average of each line and it can be written on the right where it says “Constraint”.

So there we write a number near the average of the z-axis and enter. Then a black line should appear.

If we have made a mistake or we have introduced an incorrect value, it is easy to delete the lines in the same Plotter window. In the Curves down-drop menu the values of all the curves should appear, so we can easily select the one which we want to delete.

It is necessary to do this step for each axis. We will write the different values three times in Constraint and the black lines will appear in the Plotter as shown in Figure 4.48.
The nearest line to the average will provide us with the best settings for the flight.

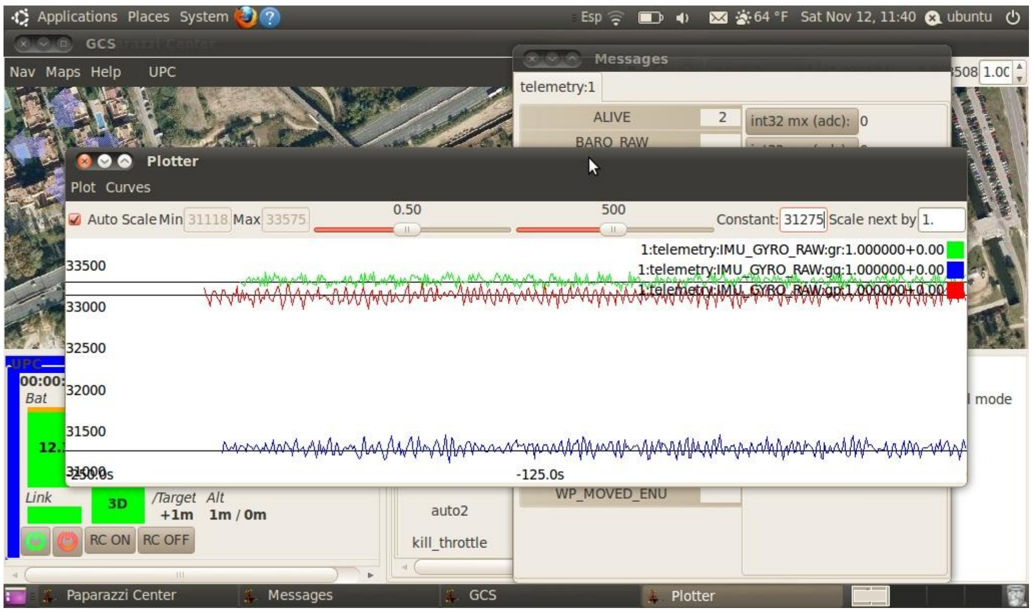
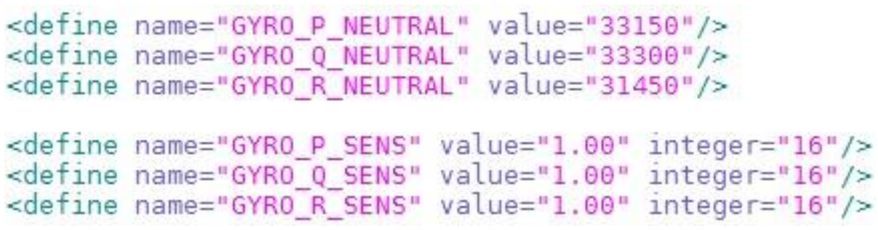


**Figure 4.48** Proper Gyro Calibration of the three axes

We have to note down the three gyro values, and then write as shown below in the UPC xml:



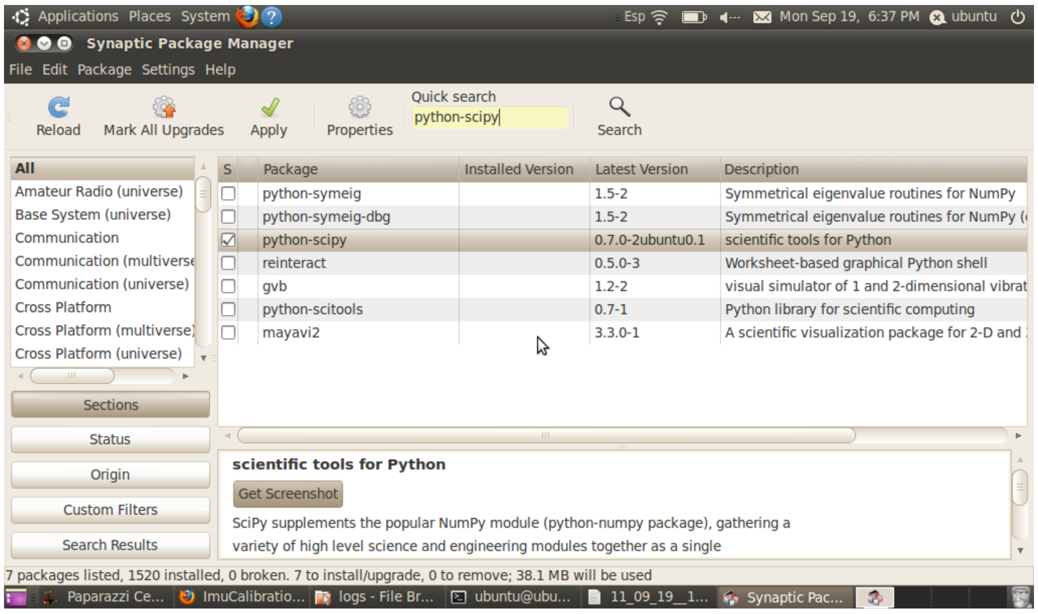
```
<define name="GYRO_P_NEUTRAL" value="33150"/>
<define name="GYRO_Q_NEUTRAL" value="33300"/>
<define name="GYRO_R_NEUTRAL" value="31450"/>

<define name="GYRO_P_SENS" value="1.00" integer="16"/>
<define name="GYRO_Q_SENS" value="1.00" integer="16"/>
<define name="GYRO_R_SENS" value="1.00" integer="16"/>
```

**Figure 4.49** Gyro values in the xml file

In order to calibrate the accelerometer and the magnetometer, we will use Python. It will compute the average of some measurements.

We first have to download it through the Synaptic Package Manager the scipy and the matplotlib as shown in Figure 4.50 and Figure 4.51, or through the Terminal simply writing:

*sudo apt-get install python-scipy*
*sudo apt-get install python-matplotlib*


Applications Places System
Esp Mon Sep 19, 6:37 PM ubuntu
Synaptic Package Manager
File Edit Package Settings Help
Reload
Mark All Upgrades
Apply
Properties
Quick search
python-scipy
Search
All
Amateur Radio (universe)
Base System (universe)
Communication
Communication (multiverse
Communication (universe)
Cross Platform
Cross Platform (multiverse)
Cross Platform (universe)
Sections
Status
Origin
Custom Filters
Search Results
Get Screenshot
Paparazzi Ce... ImuCalibratio... logs - File Br... ubuntu@ubu... 11_09_19_1... Synaptic Pac...

| S | Package | Installed Version | Latest Version | Description |
|---|---|---|---|---|
| ☐ | python-symeig | | 1.5-2 | Symmetrical eigenvalue routines for NumPy |
| ☐ | python-symeig-dbg | | 1.5-2 | Symmetrical eigenvalue routines for NumPy ( |
| ☑ | python-scipy | | 0.7.0-2ubuntu0.1 | scientific tools for Python |
| ☐ | reinteract | | 0.5.0-3 | Worksheet-based graphical Python shell |
| ☐ | gvb | | 1.2-2 | visual simulator of 1 and 2-dimensional vibrat |
| ☐ | python-scitools | | 0.7-1 | Python library for scientific computing |
| ☐ | mayavi2 | | 3.3.0-1 | A scientific visualization package for 2-D and |

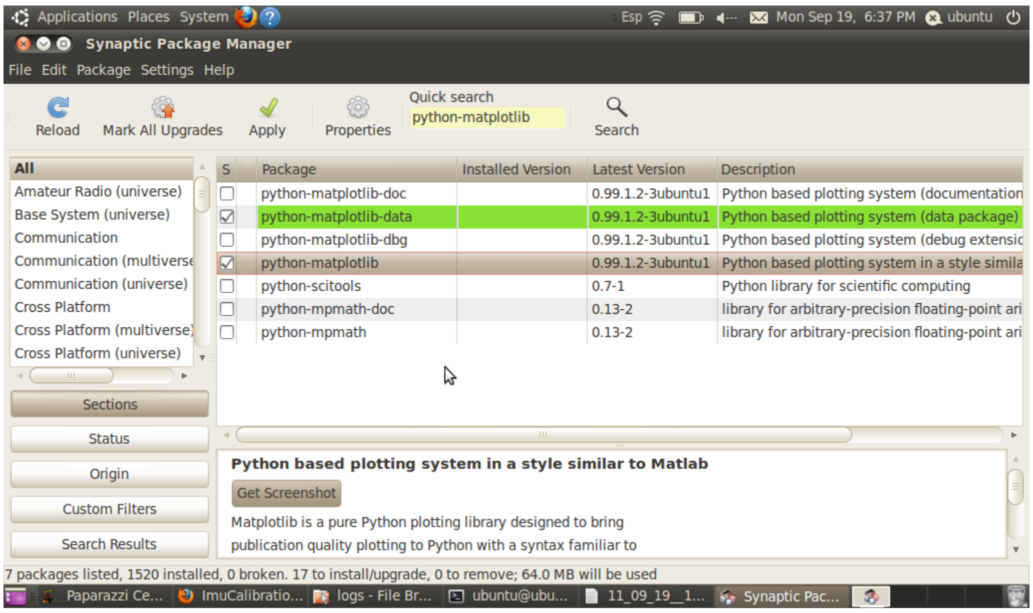
**scientific tools for Python**

SciPy supplements the popular NumPy module (python-numpy package), gathering a variety of high level science and engineering modules together as a single

7 packages listed, 1520 installed, 0 broken. 7 to install/upgrade, 0 to remove; 38.1 MB will be used

**Figure 4.50** Python scipy library

You should take into account that when you select the matplotlib package, the matplotlib-data package is also automatically selected.



| S | Package | Installed Version | Latest Version | Description |
|---|---|---|---|---|
| ☐ | python-matplotlib-doc | | 0.99.1.2-3ubuntu1 | Python based plotting system (documentation |
| ✓ | python-matplotlib-data | | 0.99.1.2-3ubuntu1 | Python based plotting system (data package) |
| ☐ | python-matplotlib-dbg | | 0.99.1.2-3ubuntu1 | Python based plotting system (debug extensic |
| ✓ | python-matplotlib | | 0.99.1.2-3ubuntu1 | Python based plotting system in a style simila |
| ☐ | python-scitools | | 0.7-1 | Python library for scientific computing |
| ☐ | python-mpmath-doc | | 0.13-2 | library for arbitrary-precision floating-point ari |
| ☐ | python-mpmath | | 0.13-2 | library for arbitrary-precision floating-point ari |

**Python based plotting system in a style similar to Matlab**

Matplotlib is a pure Python plotting library designed to bring
publication quality plotting to Python with a syntax familiar to

7 packages listed, 1520 installed, 0 broken. 17 to install/upgrade, 0 to remove; 64.0 MB will be used

**Figure 4.51** Python matplotlib library

If we have the running agent Server on the Paparazzi Center, there will always be a file that records the data that we have set up.
When we have pre-selected RAW data for the IMU calibration, the file automatically changed the measurements it was recording from the default to the raw accel, mag and gyro data.
We will use this file with Python to get the correct calibration.

The log files created by the Server agent are always saved according to this pattern: year_month_day_hour_minute_second.data

So in order to calibrate the accelerometer and the magnetometer, we have to adapt the next line to our project:

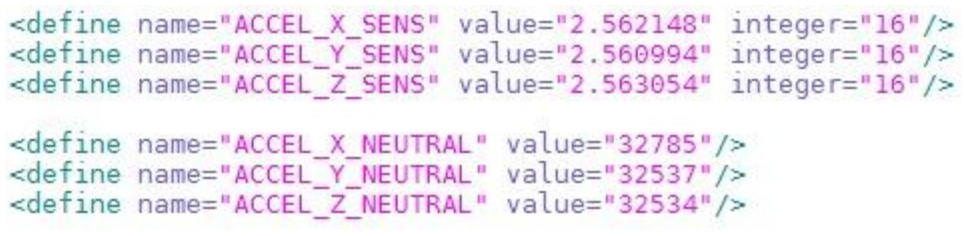
*sw/tools/calibration/calibrate.py -i <your_ac_id> -s ACCEL <path_to_data_file var/logs/xxxxxxx.data>*

*Note: Our id is UPC. Pay attention since there are two underscores between the day and the hour in the .data file.*

When computed, it gives:

```
initial guess : avg 9.75698633255 std 0.0302746646126
optimized guess : avg 9.80993941294 std 0.0243794045253

    <define name="ACCEL_X_NEUTRAL" value="32785"/>
    <define name="ACCEL_Y_NEUTRAL" value="32537"/>
    <define name="ACCEL_Z_NEUTRAL" value="32534"/>
    <define name="ACCEL_X_SENS" value="2.56214892546"   integer="16"/>
    <define name="ACCEL_Y_SENS" value="2.56099452361"   integer="16"/>
    <define name="ACCEL_Z_SENS" value="2.56305442006"   integer="16"/>
```

Now, we will write them in the airframe xml:

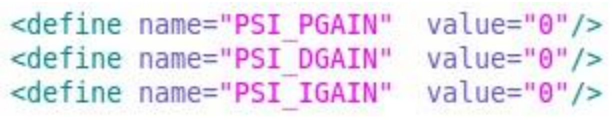


```
<define name="ACCEL_X_SENS" value="2.562148" integer="16"/>
<define name="ACCEL_Y_SENS" value="2.560994" integer="16"/>
<define name="ACCEL_Z_SENS" value="2.563054" integer="16"/>

<define name="ACCEL_X_NEUTRAL" value="32785"/>
<define name="ACCEL_Y_NEUTRAL" value="32537"/>
<define name="ACCEL_Z_NEUTRAL" value="32534"/>
```

**Figure 5.52** ACCEL values

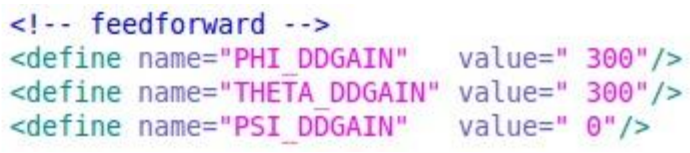
While calibrating with Python, I realized that the magnetometer was not working properly because it stated that the values were 0 0 0. This lead to a problem, in that the quadrotor did not exactly know where north was and so, it started to spin slightly around its yaw axis.

Once the ACCEL calibration change was completed, I had to set up the Gains in order to decrease the effect of the MAG on the quadrotor as much as possible. So I set the gains that affect the yaw axis to 0:



```
<define name="PSI_PGAIN"  value="0"/>
<define name="PSI_DGAIN"  value="0"/>
<define name="PSI_IGAIN"  value="0"/>
```

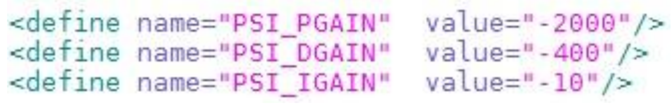
**Figure 5.53** PSI Gains



```
<!-- feedforward -->
<define name="PHI_DDGAIN"   value=" 300"/>
<define name="THETA_DDGAIN" value=" 300"/>
<define name="PSI_DDGAIN"   value=" 0"/>
```

**Figure 5.54** PSI_DD Gain

If in future a new Magnetometer is purchased, the values should be:



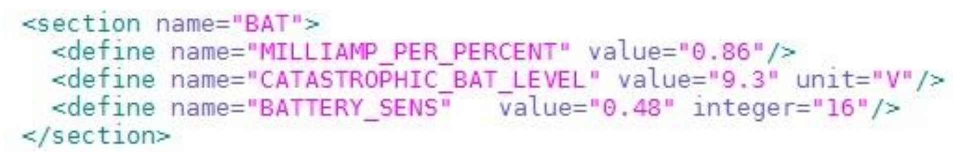
```
<define name="PSI_PGAIN"  value="-2000"/>
<define name="PSI_DGAIN"  value="-400"/>
<define name="PSI_IGAIN"  value="-10"/>
```

**Figure 5.55** PSI Default Gains

*Note: With a new Magnetometer the PSI_DDGAIN should be 300*.

Another feature that we should set up is the Catastrophic Battery Level.
This means that the Paparazzi Center will inform us to end the flight when the battery charge is running dangerously low.

This value in our battery is 9V but as we can see the value on the xml is more restrictive, so it is better to leave it at the default 9.3V.

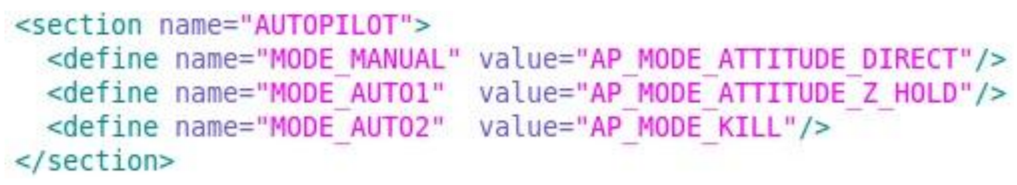


```
<section name="BAT">
  <define name="MILLIAMP_PER_PERCENT" value="0.86"/>
  <define name="CATASTROPHIC_BAT_LEVEL" value="9.3" unit="V"/>
  <define name="BATTERY_SENS"   value="0.48" integer="16"/>
</section>
```

**Figure 5.56** Battery Settings

*Note: It is important to change the Catastrophic Battery Level if another battery is purchased. Remember to increase a little the value to leave a safety margin.*

The last things that we have to modify are the modes of each switch: Manual, Auto 1 and Auto 2. You can see in the next image the proper setup.



```
<section name="AUTOPILOT">
  <define name="MODE_MANUAL" value="AP_MODE_ATTITUDE_DIRECT"/>
  <define name="MODE_AUTO1"  value="AP_MODE_ATTITUDE_Z_HOLD"/>
  <define name="MODE_AUTO2"  value="AP_MODE_KILL"/>
</section>
```

**Figure 5.57** AP modes

### 4.5.2 Flight Plan

The Flight plan used in the AP flight has been the same as for simulation.

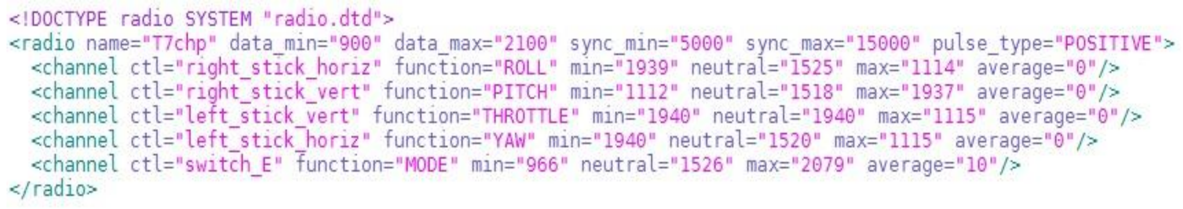
You can use different FP for AP and for Simulation.

To see the configuration of the FP see the chapter Flight Plan.

### 4.5.3 Radio xml

The next file to be set will be the radio xml. The xml file used as a starting point and we will carry out some code changes to the T7chp.xml.

We will set it properly to adapt it to the MeekPE Encoder Board and change the name to FUT7C.



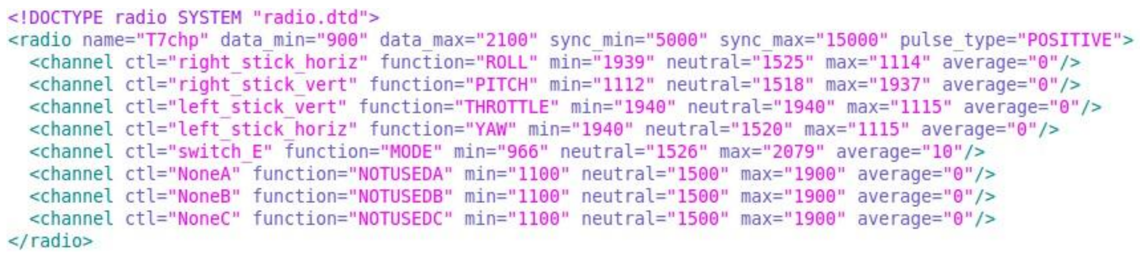
```
<!DOCTYPE radio SYSTEM "radio.dtd">
<radio name="T7chp" data_min="900" data_max="2100" sync_min="5000" sync_max="15000" pulse_type="POSITIVE">
  <channel ctl="right_stick_horiz" function="ROLL" min="1939" neutral="1525" max="1114" average="0"/>
  <channel ctl="right_stick_vert" function="PITCH" min="1112" neutral="1518" max="1937" average="0"/>
  <channel ctl="left_stick_vert" function="THROTTLE" min="1940" neutral="1940" max="1115" average="0"/>
  <channel ctl="left_stick_horiz" function="YAW" min="1940" neutral="1520" max="1115" average="0"/>
  <channel ctl="switch_E" function="MODE" min="966" neutral="1526" max="2079" average="10"/>
</radio>
```

**Figure 5.58** Radio XML for MeekPE Encoder Board

See that we located the switch_E on the 5th channel, leaving the throttle on the 3rd channel.
Another important consideration is the pulse type. Our RC is a Futaba, so we must write “positive” here. If another RC is purchased, it may be necessary to

change the pulse type to “negative”. The channel ranges are left at default values.

If you use the ATMEGA328 Encoder, we have to add three empty channels that will not be used:



```
<!DOCTYPE radio SYSTEM "radio.dtd">
<radio name="T7chp" data_min="900" data_max="2100" sync_min="5000" sync_max="15000" pulse_type="POSITIVE">
  <channel ctl="right_stick_horiz" function="ROLL" min="1939" neutral="1525" max="1114" average="0"/>
  <channel ctl="right_stick_vert" function="PITCH" min="1112" neutral="1518" max="1937" average="0"/>
  <channel ctl="left_stick_vert" function="THROTTLE" min="1940" neutral="1940" max="1115" average="0"/>
  <channel ctl="left_stick_horiz" function="YAW" min="1940" neutral="1520" max="1115" average="0"/>
  <channel ctl="switch_E" function="MODE" min="966" neutral="1526" max="2079" average="10"/>
  <channel ctl="NoneA" function="NOTUSEDA" min="1100" neutral="1500" max="1900" average="0"/>
  <channel ctl="NoneB" function="NOTUSEDB" min="1100" neutral="1500" max="1900" average="0"/>
  <channel ctl="NoneC" function="NOTUSEDC" min="1100" neutral="1500" max="1900" average="0"/>
</radio>
```

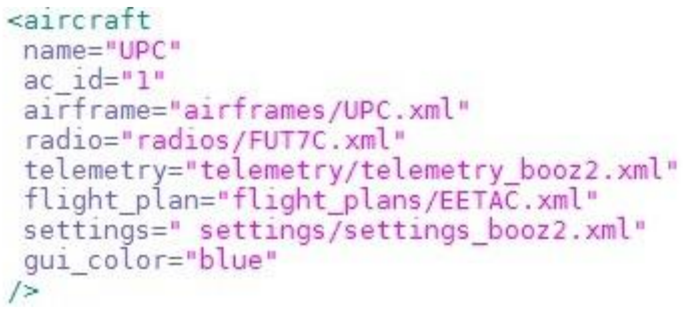
**Figure 5.59** Radio XML for ATMEGA328 Encoder Board

### 4.5.4 Conf xml

The next xml to be modified will be the conf.xml.
We will save the airframe in this file and the associated xml configuration, this means that the files that will appear in the configuration area in the Paparazzi Center (see Figure 5.63) when we select the A/C.

The configuration file can be selected by editing the following xml code or directly on the Paparazzi Center saving the A/C Changes.

The aircraft that we will build will have the following xmls associated with it:



```
<aircraft
 name="UPC"
 ac_id="1"
 airframe="airframes/UPC.xml"
 radio="radios/FUT7C.xml"
 telemetry="telemetry/telemetry_booz2.xml"
 flight_plan="flight_plans/EETAC.xml"
 settings=" settings/settings_booz2.xml"
 gui_color="blue"
/>
```

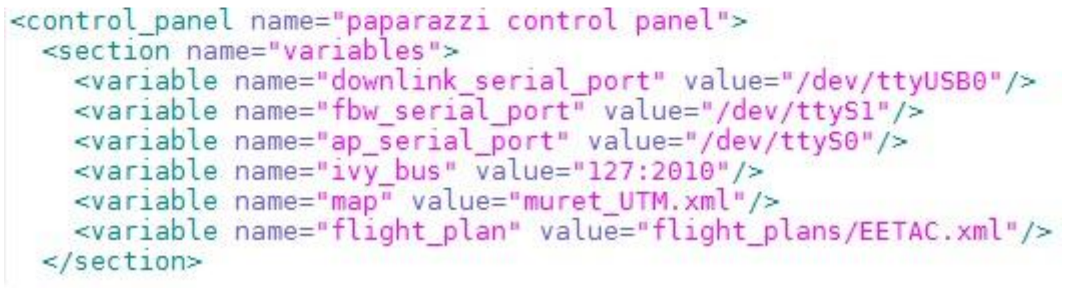
**Figure 4.60** Aircraft created in the Conf xml

The airframe and radio xml have already been explained.
The default settings and telemetry files that were used no changes were carried out.
The flight plan was done from scratch and has been detailed in the chapter Flight Plan.
The gui_colour is the colour of the airborne that will appear in the GCS when in flight.

### 4.5.5 Control_panel xml

The conf panel is the xml where all compilation files, associated running agents (Tools in Paparazzi Center) and sessions are saved.

The first thing to change is the flight plan in the variable, changing the default dummy.xml to EETAC.xml.

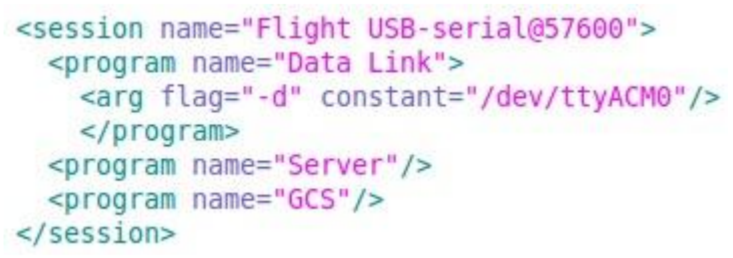


```
<control_panel name="paparazzi control panel">
  <section name="variables">
    <variable name="downlink_serial_port" value="/dev/ttyUSB0"/>
    <variable name="fbw_serial_port" value="/dev/ttyS1"/>
    <variable name="ap_serial_port" value="/dev/ttyS0"/>
    <variable name="ivy_bus" value="127:2010"/>
    <variable name="map" value="muret_UTM.xml"/>
    <variable name="flight_plan" value="flight_plans/EETAC.xml"/>
  </section>
```

**Figure 4.61** Conf_panel xml for AP

As mentioned before, the sessions can be edited and saved in this xml. (Remember that it is also possible to edit and save them from the Paparazzi Center)

In the next figure you can see that it is necessary to delete the last xml line code and edit it from ttyUSB0 to ttyACM0.
This change makes it possible not only to obtain the telemetry through the USB but also to upload the firmware.
If in future an XBEE is purchased, we should select the correct session and edit if necessary.



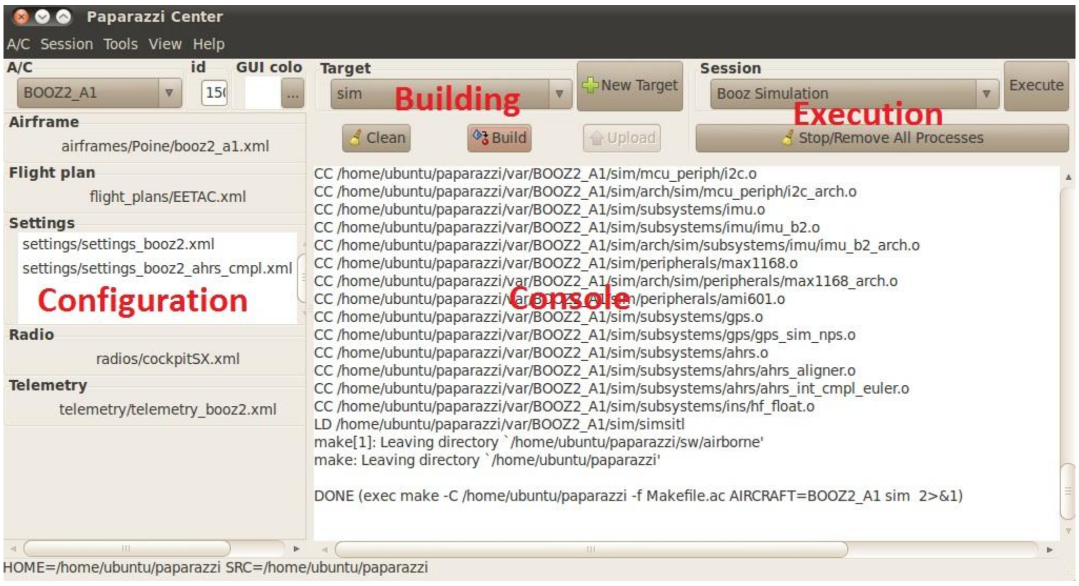
```
<session name="Flight USB-serial@57600">
  <program name="Data Link">
    <arg flag="-d" constant="/dev/ttyACM0"/>
    </program>
  <program name="Server"/>
  <program name="GCS"/>
</session>
```

**Figure 4.62** Session used for AP

# CHAPTER 5. PAPARAZZI CENTER INTERFACE

The Paparazzi developers designed an easy interface to manage the flight and display all the information required for us to monitor how the flight is going.

To launch the PGCS, located in *Places/Home Folder/Paparazzi*, it is just necessary to double-click on the *paparazzi* file. Then a window should appear and we select Run instead of Run in Terminal.
Or use the Launcher created in chapter 4.3.
When launched, a similar type screen should appear without the red labels, which will help us to explain the PGCS.

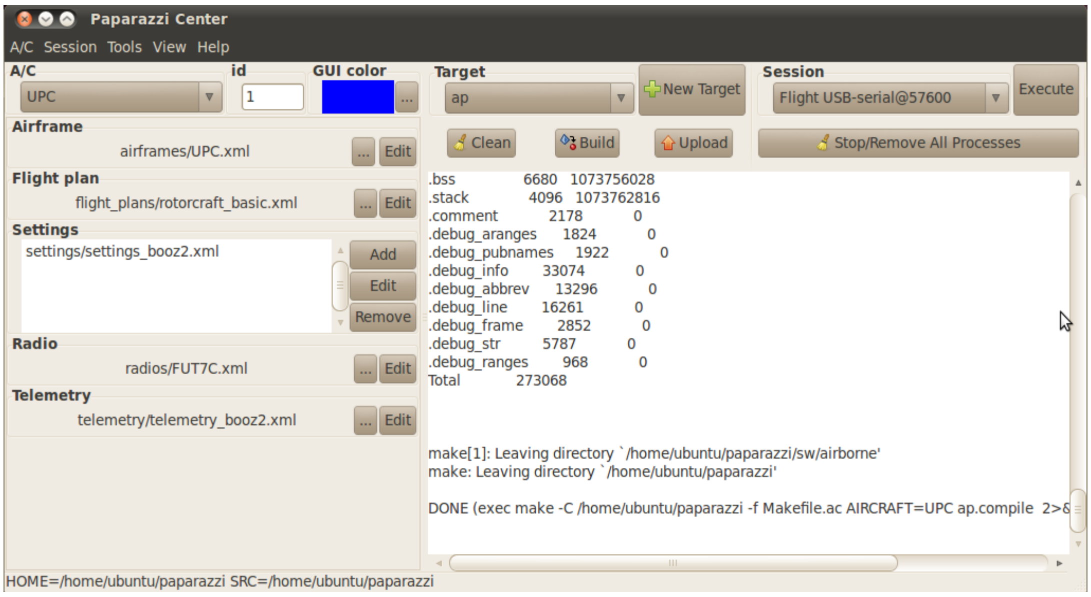


A/C: BOOZ2_A1 id: 15( GUI colo
Target: sim
Session: Booz Simulation
Airframe: airframes/Poine/booz2_a1.xml
Flight plan: flight_plans/EETAC.xml
Settings: settings/settings_booz2.xml, settings/settings_booz2_ahrs_cmpl.xml
Radio: radios/cockpitSX.xml
Telemetry: telemetry/telemetry_booz2.xml

```
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/mcu_periph/i2c.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/arch/sim/mcu_periph/i2c_arch.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/imu.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/imu/imu_b2.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/arch/sim/subsystems/imu/imu_b2_arch.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/peripherals/max1168.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/arch/sim/peripherals/max1168_arch.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/peripherals/ami601.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/gps.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/gps/gps_sim_nps.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/ahrs.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/ahrs/ahrs_aligner.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/ahrs/ahrs_int_cmpl_euler.o
CC /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/subsystems/ins/hf_float.o
LD /home/ubuntu/paparazzi/var/BOOZ2_A1/sim/simsitl
make[1]: Leaving directory `/home/ubuntu/paparazzi/sw/airborne'
make: Leaving directory `/home/ubuntu/paparazzi'

DONE (exec make -C /home/ubuntu/paparazzi -f Makefile.ac AIRCRAFT=BOOZ2_A1 sim  2>&1)
```

HOME=/home/ubuntu/paparazzi SRC=/home/ubuntu/paparazzi

**Figure 5.63** Paparazzi Center ready for Simulation



A/C: UPC id: 1 GUI color
Target: ap
Session: Flight USB-serial@57600
Airframe: airframes/UPC.xml
Flight plan: flight_plans/rotorcraft_basic.xml
Settings: settings/settings_booz2.xml
Radio: radios/FUT7C.xml
Telemetry: telemetry/telemetry_booz2.xml

```
.bss            6680   1073756028
.stack           4096   1073762816
.comment          2178          0
.debug_aranges      1824          0
.debug_pubnames      1922          0
.debug_info       33074          0
.debug_abbrev      13296          0
.debug_line       16261          0
.debug_frame        2852          0
.debug_str        5787          0
.debug_ranges        968          0
Total         273068


make[1]: Leaving directory `/home/ubuntu/paparazzi/sw/airborne'
make: Leaving directory `/home/ubuntu/paparazzi'

DONE (exec make -C /home/ubuntu/paparazzi -f Makefile.ac AIRCRAFT=UPC ap.compile  2>&
```

HOME=/home/ubuntu/paparazzi SRC=/home/ubuntu/paparazzi

**Figure 5.64** Paparazzi Center ready for AP flight

*Note: See that Upload button is only available for AP flight.*

Now, a brief description of each item will be carried out:

## 5.1 Configuration

**A/C:** I selected the Aircraft we are going to use. For simulation we used one which already existed whereas for AP we created one in chapter 4.5.4.

**Airframe:** This is the xml used with the Aircraft. As for the A/C, for simulation we use one which has already been completed for AP flight we have created the UPC xml.

**Flight Plan:** I created one flight plan for our purposes.

**Settings:** It will allow us to manage through the GCS certain Gain values, switch the RC from the GCS, decide the shape of the holding points…

**Radio:** I selected the xml with the proper setup for our flight.
For the Simulation this is not very important, but for the AP flight it has to be the one we edit correctly.

**Telemetry:** It will allow us to see the telemetry we want to check.

When we select one aircraft all the configuration xml files change to the files associated with it. To save the changes it is necessary to save the A/C (A/C drop-down menu and then Save).

Each one of these Configuration xml files can be edited or changed just by clicking on the buttons for those aims located next to the file root.
Also it is possible to edit which xml is associated with the A/C by editing it on the conf.xml as detailed before.

## 5.2 Building

Here are the buttons to compile and flash the code:

**Clean:** To clear the current session in the console.

**Build:** To compile the code.

**Upload:** To flash the code (when it is in AP)

Compilation and flashing agents will appear during their processes so they can be aborted if something goes wrong.

New targets can be added with the New Target button.

## 5.3 Execution

Here you will start the simulation or, if you are in AP, the flight.

**Execute:** When clicking this button the Simulation or the flight agents will begin.

**Session:** Each session that you can select has some associated running agents (i.e.: GCS, Server, Datalink…). More agents can be added by clicking on Tools, selecting them and then saving changes, or by editing the control_panel as seen before.

**Stop/Remove all processes:** By selecting this button you can first stop all the running agents and by clicking again, you can remove them.

## 5.4 Console

All the clean, building, flashing and execution processes will appear in the Console.
Also, the warnings and errors while compiling or executing will be displayed here. The console helped me a lot to solve de problems since appear the details of the errors.

## 5.5 Step by Step process for simulating

The step differences between being simulating or in AP are few.

By default the A/C selected is the Microjet.

The A/C used for simulation is the Booz2_a1, so you select this A/C and then you will see that the configuration files have changed. Then select *sim* in the target drop-down. Later in the session drop-down menu select *Booz Simulation* (the one you have created). Then clean, build and compile. If an error appears you have to solve it before simulation. If everything is okay, click execute and the running agents should start working. Then, you can start simulating and checking all the functions and special features available.

## 5.6 Step by Step process for AP flight

As detailed before, one special A/C (UPC) with all the correct associated files has been created for AP flight.
You only have to select this A/C and select the *ap* Target and the Flight USB-serial@57600 session. This will allow us to flash and get the data through the USB.

Once at this step, click on the clean button. When it finishes, click on the build button. If all was done properly, no error message should appear.
If so, click on the Execute button and the GCS window should show up with the position of the airframe and the flight plan near the EETAC faculty. Then if you go to the Paparazzi Center you should see the running agents (Datalink, Server and GCS).

On the Main Board should be 4 LEDs blinking for a proper AP flight. The green is the RC. The red is supply. The first orange LED following this order is Telemetry. The last LED is the GPS. They are not always blinking.
For instance, if we are indoors, the GPS LED will be off. Or if the RC is switched off the LED also will be off.

Then it is time to start enjoying the flight.

# CHAPTER 6. CONCLUSIONS AND FUTURE LINES

The goal of this project was to build a quadrotor that would be used as a research demonstrator. Therefore, the goal has been successfully completed.

As usually happens in big projects, a lot of problems and doubts arose throughout the realization of this project and these problems allowed me to learn a lot about different areas such as Ubuntu OS, xml code, how a Main Board or an encoder board works, or how the boards interact with each other...

On the one hand, all the hardware integration has been done successfully, dealing with difficult problems such as the encoder problem that can be completely solved, and others that could only be mitigated due to our budget limitations such as the Magnetometer error.

On the other hand, all the software settings were successfully completed and tested. Also, how to interact with the Paparazzi Center was detailed.

So the results are excellent given time and budget limitations.

Nevertheless, these limitations have greatly reduced the available settings and range of possibilities that could have been accomplished by a microUAV as the one developed in this project.

The following points summarize the future lines that quadrotor could follow:

- Send the IMU board to PPZUAV to fix the MAG and the ATMEGA328.
- Use all the functions of the quadrotor, a wireless Datalink such as a ZigBee module would greatly improve the features and possibilities of the quadrotor.
- A webcam on board the quadrotor would improve greatly the range of possible activities to be done.
- In the future, if more quadrotors are purchased, could be studied how to interact them in swarm.
- New software settings and changes to the current files could be tested and implemented to exploit their benefits.

# REFERENCES

[1] paparazzi.enac.fr [online accessed: April 2010 – November 2011]

[2] A.Barrientos, J.del Cerro, P.Gutiérrez, R.San Martín, A.Martínez, C.Rossi, Vehículos aéreos no tripulados para uso civil. Tecnología y aplicaciones", "Grupo de Robótica y Cibernética, Universidad Politécnica de Madrid

[3] www.eetac.upc.edu EETAC faculty website [online accessed: July 2011]

[4] Proposta de demostrador de Recerca

[5] X-3D-BL User's Manual English v2.1. AscTec GmbH

[6] Google Maps

[7] Sol·licitud per la convocatoria de projectes d'activitats docents Abril 2010

[8] ppzuav.com [online accessed: April 2011 – October 2011]

[9] User Manual PPM Encoder Board ATMEGA168 version 4.2(firmware)

[10] www.ubuntu.com/download/ubuntu/download [online accessed June 2011]

[11] Preliminary Considerations for Classifying Hazards of Unmanned Aircraft Systems. Kelly J. Hayhurst, Jeffrey M. Maddalon, Paul S. Miner, and George N. Szatkowski (Langley Research Center, Hampton, Virginia) Michael L. Ulrey (The Boeing Company, Seattle, Washington) Michael P. DeWalt (Certification Services, Inc., East Sound, Washington) Cary R. Spitzer (AvioniCon, Williamsburg, Virginia). Nasa/TM-2007-214539.

[12] http://electronics.howstuffworks.com/brushless-motor.htm [online accessed October 2011]

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# ANEXOS

**TÍTULO DEL TFC: Setup and Integration of a Quadrotor used for research demonstrator**

**TITULACIÓN: Ingeniería Técnica Aeronáutica, especialidad Aeronavegación**

**AUTOR: Kenneth Llonch González**

**DIRECTOR: Cristina Barrado Muxí**

**CO-DIRECTOR: Jorge Ramírez Alcántara**

**FECHA: 22 de Noviembre de 2011**

# ANEX A. PLANNING

The goal of this project is to design and develop a quadrotor to be used for teaching purposes, carrying out all the phases and processes needed for correct functioning.
Before doing this project it was mandatory to first carry out the safety analysis as shown in SAE ARP 4761 and to develop step by step various early design processes as described in SAE ARP 4754.
During the Design process it was necessary to analyze the functions present in the NASA document [11] to understand which functions the quadrotor must satisfy. The document where this analysis was done is called Aircraft Level Functions and belongs to the ICARUS research group.
The Aircraft Level Functions is divided into 4 main chapters: Aviate, Navigation, Communication and Mitigate.
Our Quadrotor has to satisfy various Aviate functions.
The functions are related to the actions during the flight, the proper preparation on the ground and actions related to system management, for instance:

Convey FP Status
Convey AGT Status
Maintain command and control between control station and UAS

To review in detail all Aviate Functions that the quadrotor must satisfy, see the next diagram:

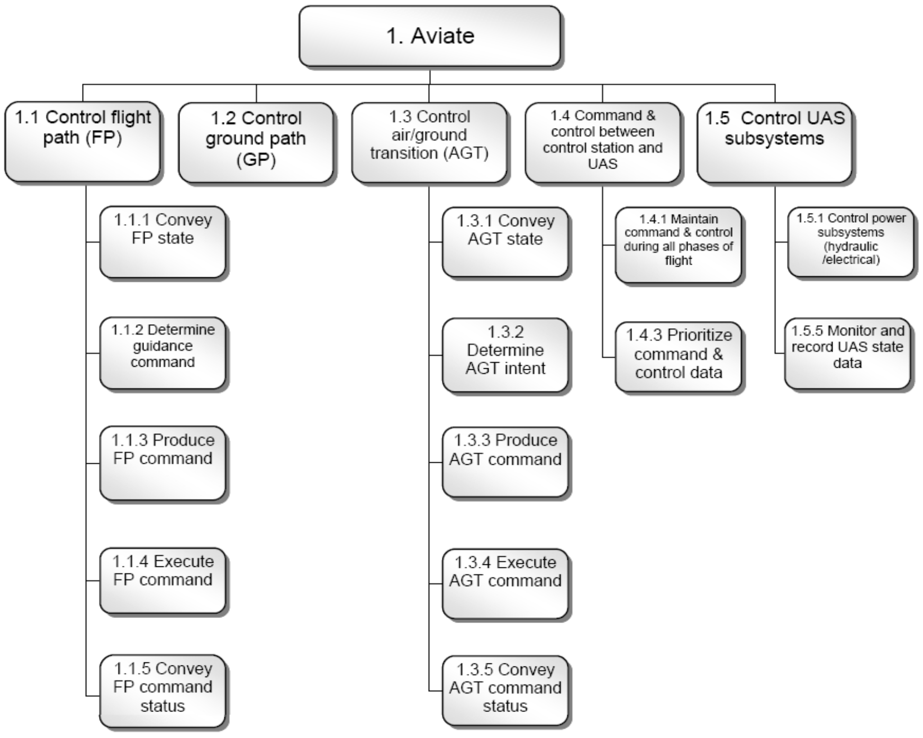

1. Aviate
1.1 Control flight path (FP)
1.2 Control ground path (GP)
1.3 Control air/ground transition (AGT)
1.4 Command & control between control station and UAS
1.5 Control UAS subsystems
1.1.1 Convey FP state
1.1.2 Determine guidance command
1.1.3 Produce FP command
1.1.4 Execute FP command
1.1.5 Convey FP command status
1.3.1 Convey AGT state
1.3.2 Determine AGT intent
1.3.3 Produce AGT command
1.3.4 Execute AGT command
1.3.5 Convey AGT command status
1.4.1 Maintain command & control during all phases of flight
1.4.3 Prioritize command & control data
1.5.1 Control power subsystems (hydraulic /electrical)
1.5.5 Monitor and record UAS state data

**Figure Anex 1** Aviate Diagram

It also has to satisfy some Navigate functions. These functions are related to the management and the execution of the Flight Plan, for instance:

Determine Flight Plan
Produce Navigation Command Status
Determine Navigation Command Status

To review in detail all the Navigate Functions that the quadrotor must satisfy, see the next diagram:

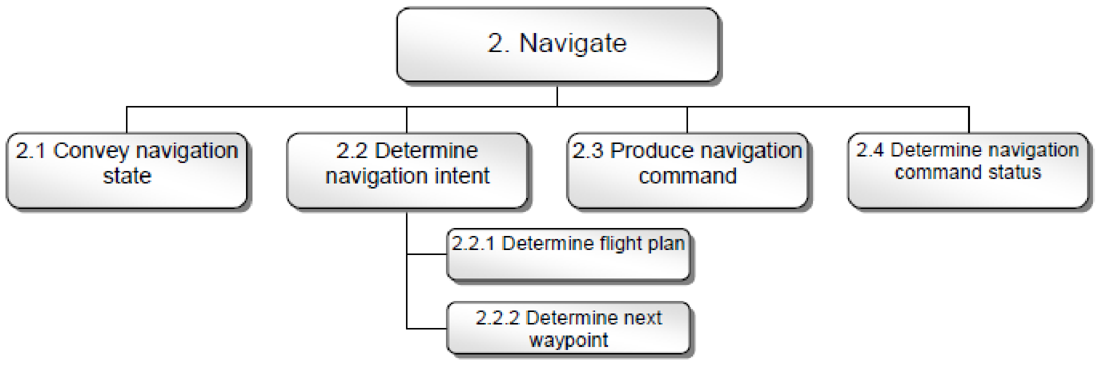

2. Navigate
2.1 Convey navigation state
2.2 Determine navigation intent
2.3 Produce navigation command
2.4 Determine navigation command status
2.2.1 Determine flight plan
2.2.2 Determine next waypoint

**Figure Anex 2** Navigate Diagram

Nonetheless, the Quadrotor does not have to satisfy the Communication functions because it will not fly over 300m so there will not be communication with the ATC.

Finally, the quadrotor has to satisfy some Mitigate functions. The functions are related with avoidance of the ground and vertical objects, avoidance of adverse weather conditions and management of contingencies, for instance:

Avoid Air Traffic
Avoid adverse environmental conditions
Prioritize Mitigation Command

To review in detail all Mitigate Functions that the quadrotor must satisfy, see the next diagram:

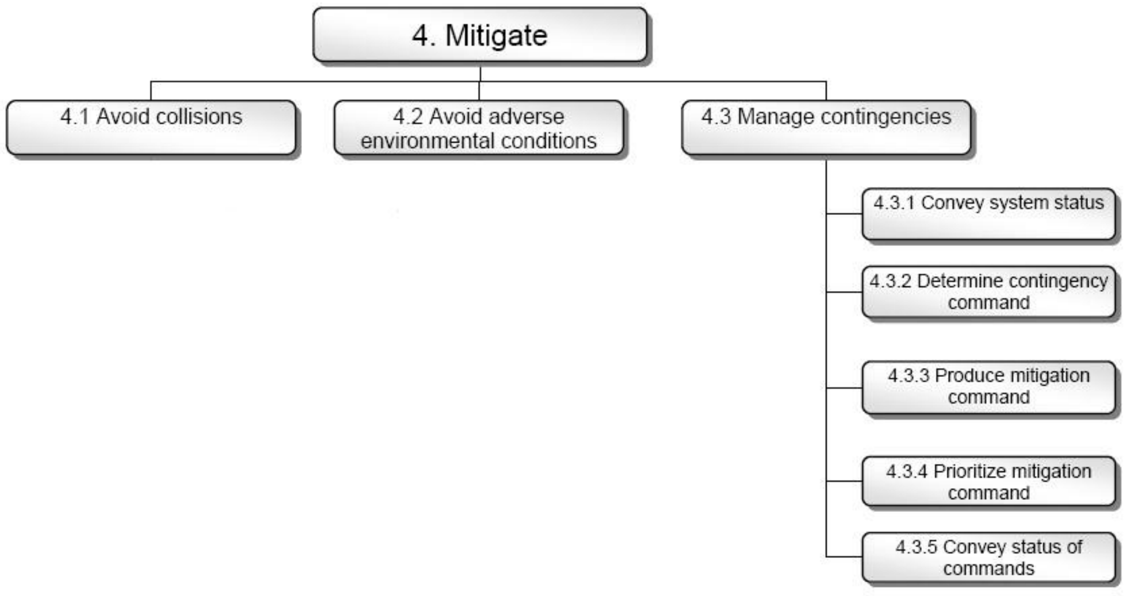

4. Mitigate
4.1 Avoid collisions
4.2 Avoid adverse environmental conditions
4.3 Manage contingencies
4.3.1 Convey system status
4.3.2 Determine contingency command
4.3.3 Produce mitigation command
4.3.4 Prioritize mitigation command
4.3.5 Convey status of commands

**Figure Anex 3** Mitigate Diagram

On the other hand, in the Safety Process there are also some phases to be satisfied before starting the development of the quadrotor.

The Functional Hazard Assessment (also known as Functional Hazard Analysis) is one of the most important phases in the Safety Process.

In the FHA phase, there are two documents. In one of them, all the phases present in the Aircraft Level Functions are classified according to their criticality. Located in the other document are the systems which are classified according to their criticality.

Another important step in the Safety Process are the documents which analyze the problems that may appear in the quadrotor, either the particular problems (PRA), the common causes (CCA) or the problems due to the architecture (ZSA).

# ANEX B. ARCHITECTURE

## B.1 UAV

The UAV is the system which proceeds with the flight. It gets some data from the communication subsystem and its movement is restricted by its own software and by the safety restriction.

In Figure Anex 4 the UAV segment is broken down.
This will be explained in the proceeding chapters.

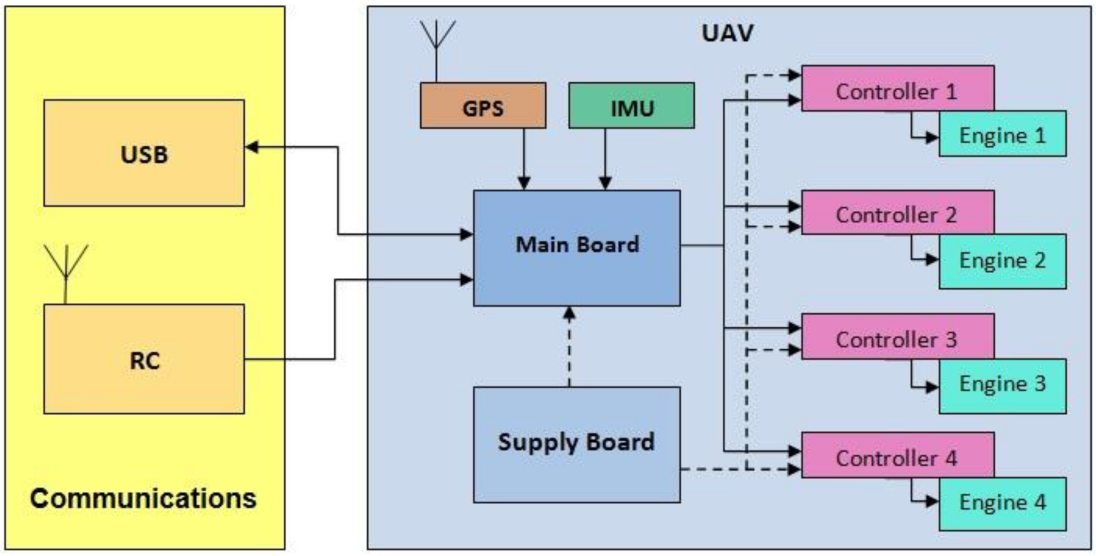


**Figure Anex 4** Connection between UAV and Communication segments

### B.1.1. Airframe

The engines, avionics systems and communication systems are implemented on the Asctec X-CSM airframe.
For a correct positioning, there are various holes in which to place the screws.

In the central core there are 4 tiny holes, which must be perforated slightly in order to enlarge them (i.e. solid or dashed yellow arrows in Figure Anex 5).
On the lower side of each branch, there are 2 holes in which the controller screws can be placed (i.e. red dashed arrows in Figure Anex 5).
At either end of each branch there are 4 holes where the engine can be fixed to. (i.e. green arrows in Figure Anex 5)

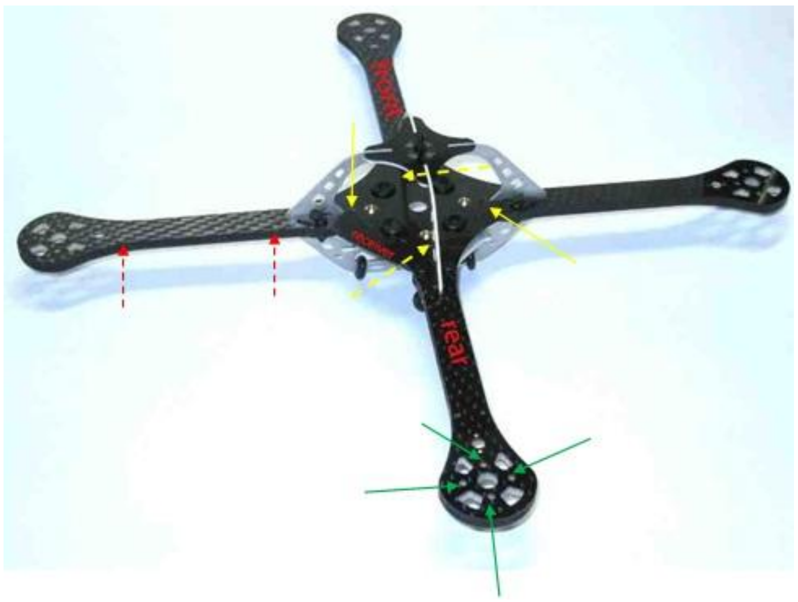


**Figure Anex 5** Detailed holes position

### B.1.2. Avionics

Avionics include all the electronic systems used in the aircraft, some of which interact between each other and the management and/or display of data, which keeps us informed of how the flight is going.

First a brief introduction to the systems in the quadrotor:

There is a Main Board, which has to manage all the information from the other systems and ensure that the flight proceeds as designed. This is our AP.

We also have a GPS that will help us to locate our quadrotor in the Flight Plan. To do that, it will download the ephemerides and with an average error of 1-2 meters, it will tell us where we are flying.

Another important board will be the IMU (Inertial Measurement Unit).
This board contains the gyro, accelerometer and magnetometer which inform the quadrotor about its own movements. This will allow us to verify, that the spins and maneuvers are correct, ensure their accuracy and verify them with the GPS information on the PC.

Finally the engine controllers X-BLDC (Extreme - Brushless Direct Current) are the boards that control thrust to move the quadrotor according to Main Board needs.

Now, some indications about where they are located:

The Main Board is fixed by 4 screws in the central part of the core as seen in Figure Anex 7. Between the Main Board and the airframe surface there are 4 spacer rings of 6mm to avoid vibrations that could damage the board. The Main board gets 5V from the Supply Board through a Picoblade wire of 2 pins.

The Main Board is in charge of managing the movement of the quadrotor when it is flying, and deciding the thrust relations between the engines to get the right torque so it flies correctly.

The IMU is located above the Main Board and is fixed by 4 screws of 25mm, and 4 spacers of 20mm. The screws have 6mm of diameter and a 2mm high head. The connection between both boards is done with a Picoblade wire of 12 pins, although just 10 pins will be used.
For more details see the ICD (Interface Control Document): Main Board – IMU.

As shown in Figure Anex 7, the GPS is located above the core of the airframe to prevent electromagnetic interferences produced in the branches (engines) or by the core (other boards). The GPS will be fixed to a back-plate with four screws (facing up) of 12mm length, 6mm of diameter and a 2mm high head, and spacers of 5mm. The order of items from bottom to top to fix the GPS will be: screw facing-up, backplate, female, spacer, GPS and on the upper face of the GPS another female to fix it.
The connection between the GPS and the Main Board will be completed by an 8 pin Picoblade wire, although just 4 pins will be used.
For more details see the ICD: Main Board – GPS.

One small back-plate with a 3mm diameter hole in the middle will be screwed to the highest screw present in the core. Its function is to allow the GPS to be plugged on the upper side of the core. To do this, 4 holes of 2.5mm will be made on the extremes to fix the GPS to the back-plate.
The attachment of the back-plate to the top of the core will be done with the screw located in top of the core as mentioned previously, so it will not be necessary to have more screws, just turn face-up the one present shown in Figure Anex 6. Between the back-plate and the core we will put a spacer of 5 mm. We will fix the back-plate with the black screw shown in Figure Anex 6, putting it at the upper side of the screw.
Once the back-plate is fixed, it is better to cut off the outermost part.

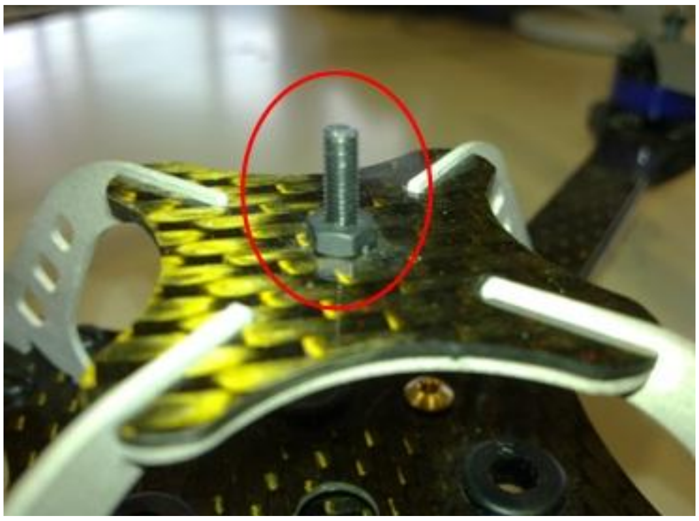


**Figure Anex 6** Screw used to fix the backplate

The engine controllers (X-BLDC), located on the outermost part of each branch, are fixed by 4 autoperforating screws and by adding 4 mm spacers , we avoid the vibrations. It's important to keep in mind that these controllers work properly just with the engines X-BL 52s. If the engines are changed is mandatory to check before flight the correct behavior of each.

The quadrotor obtains from itself data such as position and movement due to the IMU and GPS, which help it to determine the actual position.

To embed the Power Supply into the middle part of the core, it was necessary to enlarge the central hole of the plate located in the middle of the core such that its size was approximately the same size as the Power Supply's micro.

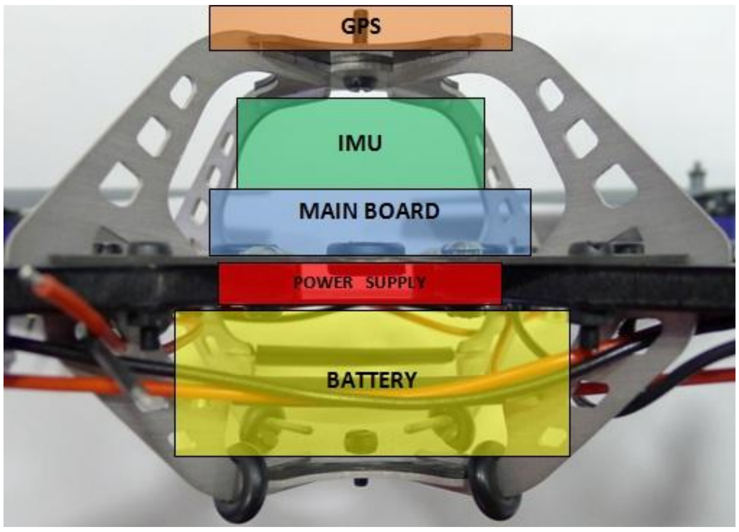


**Figure Anex 7** General vision of the systems in the core

### B.1.3. Energy

The battery is a LiPo with 11.1V and 800mAh.
It's located in the central lower part of the core as shown in Figure Anex 7. To maintain it fixed, a Velcro strip is used. The battery purchased didn't have the corresponding charger so it is necessary to choose one that works fine with this battery. The battery had a wire with a JST-RCY connection to plug it with the Power Supply Board. We removed it to improve the supply of the power supply and the battery through banana connectors.
For more information see the ICD: Power Supply – Battery.

The Power Supply Board is located in the middle of the as you can see in Figure Anex 7. In the Supply Board, 4 wires will be soldered in the upper face of the board and 4 in the lower one. These 8 wires correspond to the voltage and ground of the 4 engines (black are ground and red the supply).
Moreover 2 wires from the battery will be soldered.
For more details see the ICD: Main Board – Power Supply.

### B.1.4. Engines

The engines used in this project are designed by Hacker GmbH, and are called X-BL-52s. (Extreme – Brushless).

First, let’s explain how a DC (Direct Current) motor works:
In a DC motor (see Figure Anex 8), there are permanent magnets on the outside and a spinning armature on the inside. The permanent magnets are stationary, so they are called stators. The armature rotates, so it is called the rotor.

**Figure Anex 8** Typical DC motor

The armature contains an electromagnet. When you run electricity into this electromagnet, it creates a magnetic field in the armature that attracts and repels the magnets in the stator. So the armature spins through 180 degrees. To keep it spinning, you have to change the poles of the electromagnet. The brushes handle this change in polarity.
They make contact with two spinning electrodes attached to the armature and flip the magnetic polarity of the electromagnet as it spins [13].

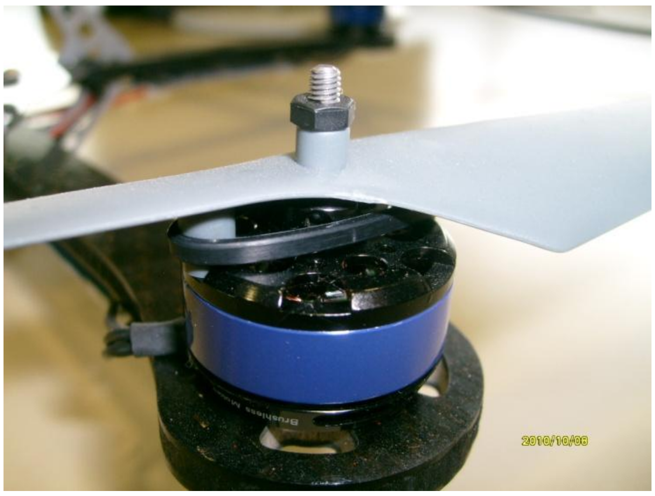


**Figure Anex 9** Hacker Engine

Each engine will be fixed to the 4 holes situated in each end of the branch (green arrows in Figure Anex 5). Two propellers turn clockwise and another two anticlockwise. There are three wires that connect the controller to the engine. The order in which you connect the three wires does not matter.

## B.2 Ground Station

Now that we have reviewed the architecture, we will take a look at the Ground Station systems.

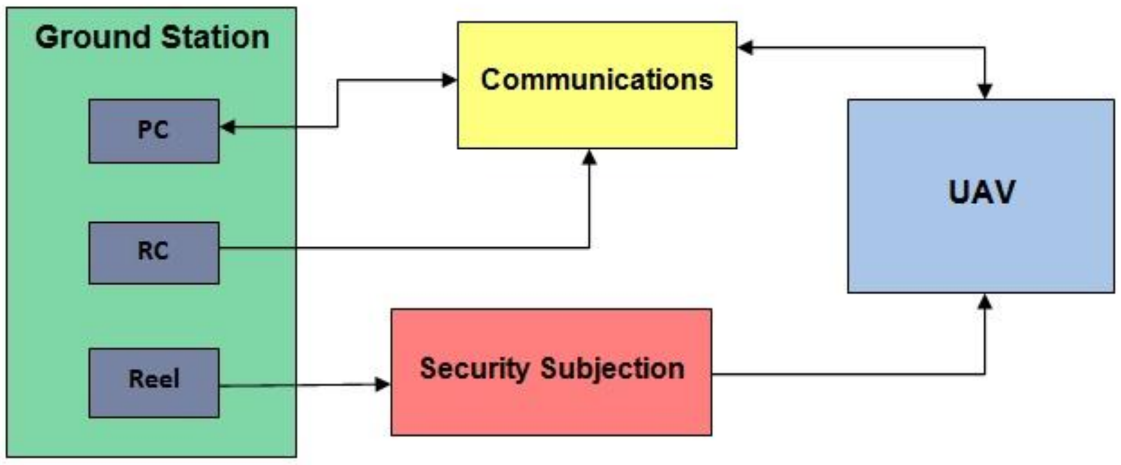


**Figure Anex 10** Ground Station Scheme

The Ground Station is formed by the RC, the computer and a reel.

The RC contains the switch which allows us to change the flight mode (Manual, Auto1 and Auto2) and the levels to control the quadrotor (i.e. pitch, yaw, roll, trim…).
The RC used is the one included in the FUTK7005 pack and is the one below:

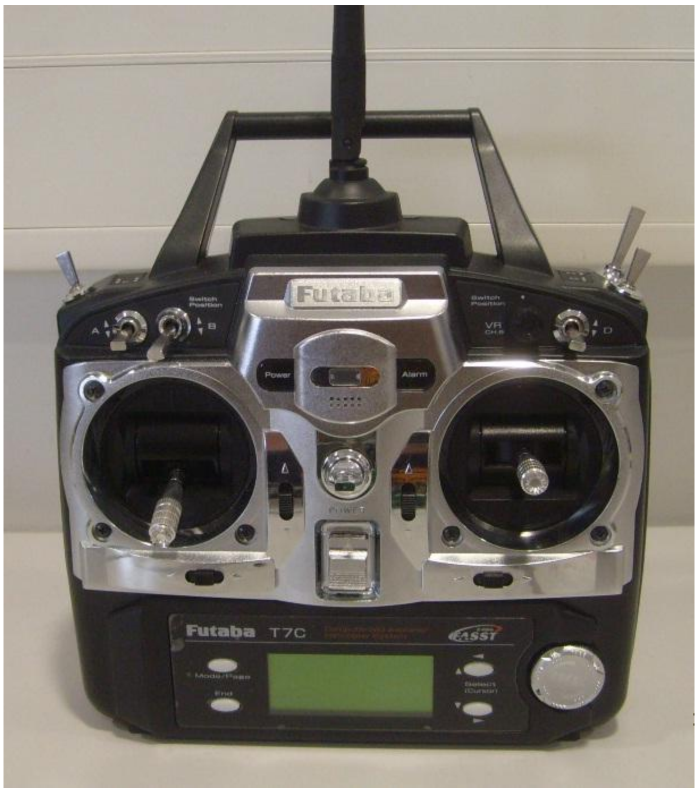


**Figure Anex 11** Futaba 7C 2.4GHz Transmitter

The computer could be any laptop with a USB port to boot the Linux from the USB.

The reel allows us to control physically the maximum distance of the UAV from the center of the OZ.

## B.3 Communications

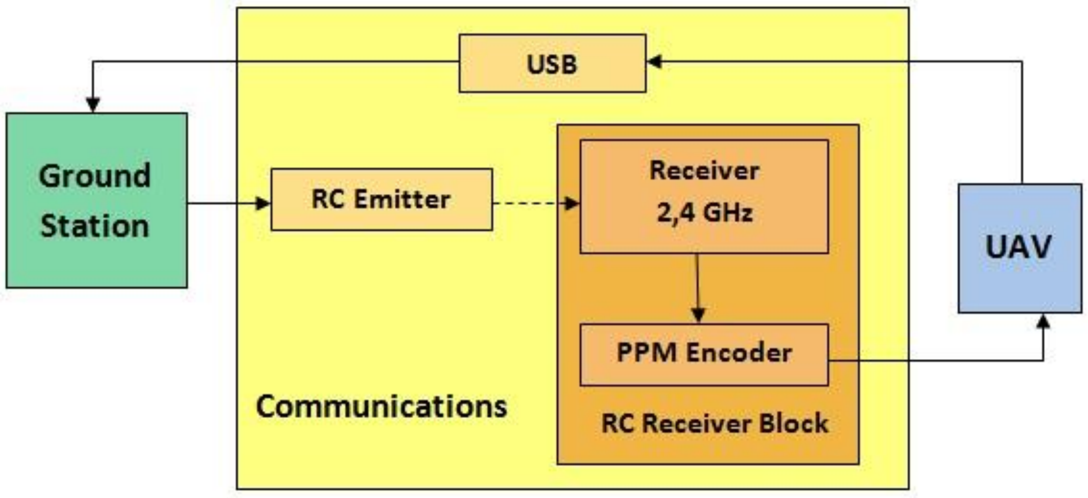


**Figure Anex 12** Communication Diagram

The uplink will be sent from the RC emitter which belongs to the Ground Station segment. The PPM signal sent from the emitter is encoded in FM and modulated at 2.4GHz.
This signal is received by the receiver and it is transmitted by wire to the encoder. Once there, the encoder demodulates the signal into PPM signal so as to satisfy the format input into the Main Board.
The Receiver (+ Encoder) is located on the lateral side of the core as shown below in Figure Anex 13.

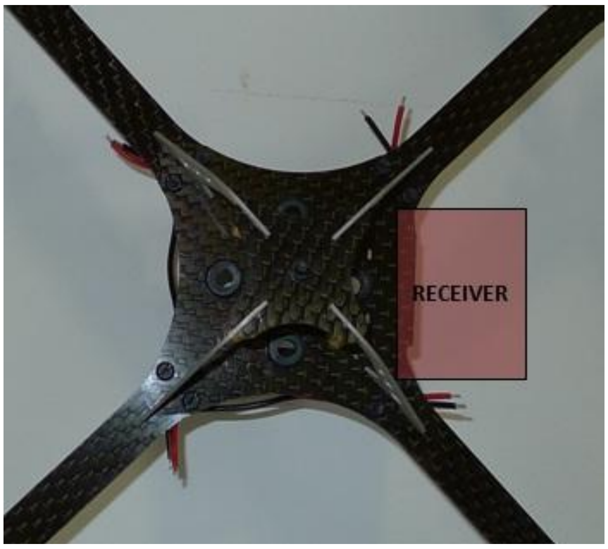


**Figure Anex 13** Upper sight of the core

If we connect the receiver output signal directly to the input of the Main Board it does not work properly.
The next image shows how the PPM signal is formed by a sequence of pulses. The information is not in the width of the pulses. It is in the period between them:

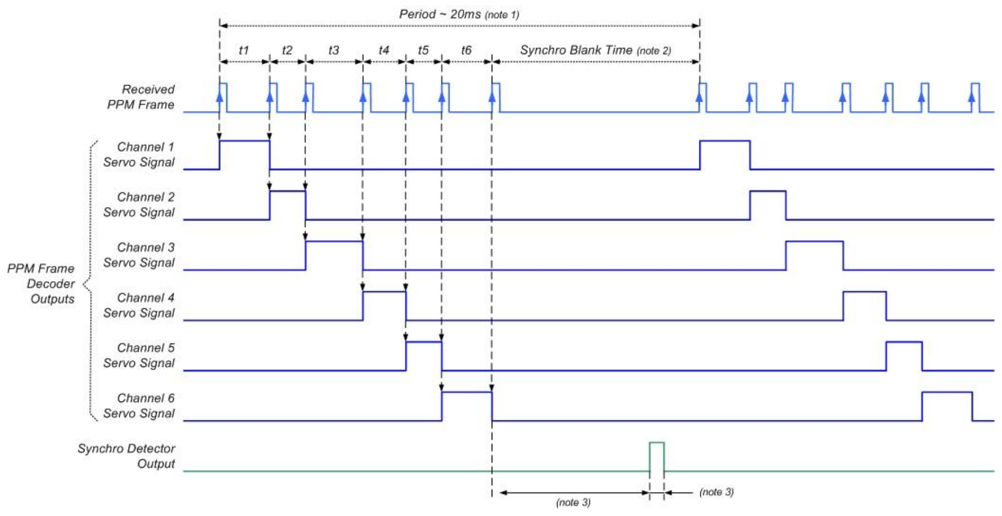


**Figure Anex 14** PPM array

It is important to consider that the emitter and the receiver have 7 channels each, but that the encoder has 8 input channels.
The connection between the encoder and the Main board is carried out by a 3 pin Picoblade wire. For more details see the next ICD: Main Board – Encoder.

## B.4 Security Subjection

In this chapter we are going to see a brief description of the security subjection.

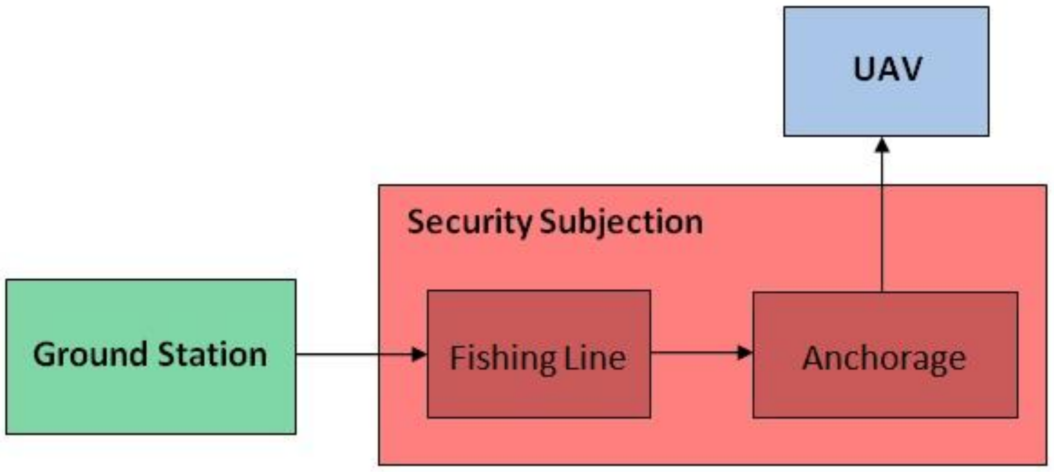


**Figure Anex 15** Security Subjection Scheme

As shown above, the safety subjection is the system that connects the GS and the UAV in a physically. The anchorage and the fishing line are the 2 objects that belong to the Safety Subjection subsystem.
The fishing line will be tied in the lower screw located in the inferior side of the core (see Figure Anex 16). It will prevent the quadrotor from leaving the OZ. The maximum distance of the fishing line is 20m. Bearing in mind that the fishing line will be collected into the reel, which is in the outer side of the OZ, this means that from the reel to the OZ center approximately 25m will be used.

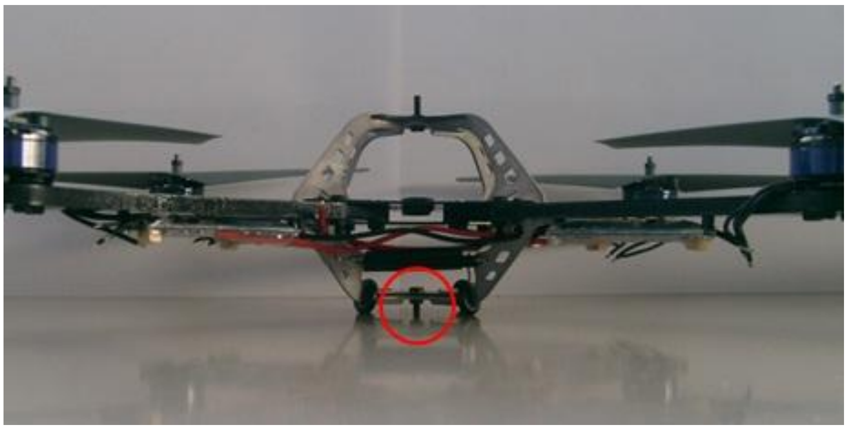


**Figure Anex 16** Screw used to tie the fishing line

The anchorage is formed by picks secured to the floor to maintain the fishing line near the ground.

# ANEX C. XML FILES

All the xml files used during the project will be in this Annex.
This will help to understand the structure and which information contains each file.

To maintain the xml format you see when you open them on Gedit, the files where directly printed to pdf with the Gedit program. This will make easier the interpretation.

The root of the files will appear on the top of the page, so the traceability will be easier.

*Note: The Flight Plan is called EETAC and it includes another file called EETAC_sectors (also enclosed to this project). This file was created from scratch.*

```xml
<!-- Airframe used for Simulation (Booz2_a1.xml) -->

<!-- Few changes from the default one -->

<airframe name="BOOZ2_a1">

  <servos min="0" neutral="0" max="0xff">
    <servo name="FRONT" no="0" min="0" neutral="0" max="255"/>
    <servo name="BACK"  no="1" min="0" neutral="0" max="255"/>
    <servo name="RIGHT" no="2" min="0" neutral="0" max="255"/>
    <servo name="LEFT"  no="3" min="0" neutral="0" max="255"/>
  </servos>

  <commands>
    <axis name="PITCH"  failsafe_value="0"/>
    <axis name="ROLL"   failsafe_value="0"/>
    <axis name="YAW"    failsafe_value="0"/>
    <axis name="THRUST" failsafe_value="0"/>
  </commands>

<!-- not needed anymore - in subsystem -->
  <section name="ACTUATORS_MKK" prefix="ACTUATORS_MKK_">
    <define name="NB" value="4"/>
    <define name="ADDR" value="{ 0x52, 0x54, 0x56, 0x58 }"/>
  </section>
<!--  -->

  <section name="SUPERVISION" prefix="SUPERVISION_">
    <define name="MIN_MOTOR" value="2"/>
    <define name="MAX_MOTOR" value="210"/>
    <define name="TRIM_A" value="0"/>
    <define name="TRIM_E" value="0"/>
    <define name="TRIM_R" value="0"/>
    <define name="NB_MOTOR" value="4"/>
    <define name="SCALE" value="256"/>
    <define name="ROLL_COEF"  value="{    0,    0, -256,  256}"/>
    <define name="PITCH_COEF" value="{  256, -256,    0,    0}"/>
    <define name="YAW_COEF"   value="{ -256, -256,  256,  256}"/>
    <define name="THRUST_COEF" value="{ 256,  256,  256,  256}"/>
  </section>

  <section name="IMU" prefix="IMU_">

    <define name="GYRO_P_NEUTRAL" value="33924"/>
    <define name="GYRO_Q_NEUTRAL" value="33417"/>
    <define name="GYRO_R_NEUTRAL" value="32809"/>
    <define name="GYRO_P_SENS" value=" 1.01" integer="16"/>
    <define name="GYRO_Q_SENS" value=" 1.01" integer="16"/>
    <define name="GYRO_R_SENS" value=" 1.01" integer="16"/>

    <define name="ACCEL_X_NEUTRAL" value="32081"/>
    <define name="ACCEL_Y_NEUTRAL" value="33738"/>
    <define name="ACCEL_Z_NEUTRAL" value="32441"/>
    <define name="ACCEL_X_SENS" value="2.50411474" integer="16"/>
    <define name="ACCEL_Y_SENS" value="2.48126183" integer="16"/>
    <define name="ACCEL_Z_SENS" value="2.51396167" integer="16"/>

    <define name="MAG_X_CHAN" value="4"/>
    <define name="MAG_Y_CHAN" value="0"/>
    <define name="MAG_Z_CHAN" value="2"/>

    <define name="MAG_X_NEUTRAL" value="2358"/>
    <define name="MAG_Y_NEUTRAL" value="2362"/>
    <define name="MAG_Z_NEUTRAL" value="2119"/>
```

```
<!--    <define name="MAG_X_SENS" value="-4.94075530" integer="16"/>-->
<!--    <define name="MAG_Y_SENS" value=" 5.10207664" integer="16"/>-->
    <define name="MAG_X_SIGN" value="-1"/>
    <define name="MAG_Y_SIGN" value=" 1"/>
    <define name="MAG_Z_SIGN" value="-1"/>

    <define name="MAG_Z_SENS" value="4.90788848" integer="16"/>
    <define name="MAG_45_HACK" value="1"/>
<!--    <define name="MAG_X_SENS" value="-4.94075530 * sqrt(2)/2" integer="16"/> -->
<!--    <define name="MAG_Y_SENS" value=" 5.10207664 * sqrt(2)/2" integer="16"/> -->
    <define name="MAG_X_SENS" value=" 3.4936416" integer="16"/>
    <define name="MAG_Y_SENS" value=" 3.607713"  integer="16"/>

   <define name="BODY_TO_IMU_PHI"   value="RadOfDeg(4.)"/>
   <define name="BODY_TO_IMU_THETA" value="RadOfDeg(3.)"/>
   <define name="BODY_TO_IMU_PSI"   value="RadOfDeg(0.)"/>
<!--
   <define name="BODY_TO_IMU_PHI"   value="RadOfDeg(0.)"/>
   <define name="BODY_TO_IMU_THETA" value="RadOfDeg(0.)"/>
   <define name="BODY_TO_IMU_PSI"   value="RadOfDeg(0.)"/>
-->

  </section>

  <section name="STABILIZATION_RATE" prefix="STABILIZATION_RATE_">

    <define name="SP_MAX_P" value="10000"/>
    <define name="SP_MAX_Q" value="10000"/>
    <define name="SP_MAX_R" value="10000"/>

    <define name="GAIN_P" value="-400"/>
    <define name="GAIN_Q" value="-400"/>
    <define name="GAIN_R" value="-350"/>

  </section>

  <section name="STABILIZATION_ATTITUDE" prefix="STABILIZATION_ATTITUDE_">

    <!-- setpoints -->
    <define name="SP_MAX_PHI"     value="RadOfDeg(45.)"/>
    <define name="SP_MAX_THETA"   value="RadOfDeg(45.)"/>
    <define name="SP_MAX_R"       value="RadOfDeg(90.)"/>
    <define name="DEADBAND_R"     value="250"/>

    <!-- reference -->
    <define name="REF_OMEGA_P"  value="RadOfDeg(800)"/>
    <define name="REF_ZETA_P"   value="0.9"/>
    <define name="REF_MAX_P"    value="RadOfDeg(300.)"/>
    <define name="REF_MAX_PDOT" value="RadOfDeg(7000.)"/>

    <define name="REF_OMEGA_Q"  value="RadOfDeg(800)"/>
    <define name="REF_ZETA_Q"   value="0.9"/>
    <define name="REF_MAX_Q"    value="RadOfDeg(300.)"/>
    <define name="REF_MAX_QDOT" value="RadOfDeg(7000.)"/>

    <define name="REF_OMEGA_R"  value="RadOfDeg(500)"/>
    <define name="REF_ZETA_R"   value="0.9"/>
    <define name="REF_MAX_R"    value="RadOfDeg(180.)"/>
    <define name="REF_MAX_RDOT" value="RadOfDeg(1800.)"/>

    <!-- feedback -->
    <define name="PHI_PGAIN"  value="-400"/>
    <define name="PHI_DGAIN"  value="-300"/>
    <define name="PHI_IGAIN"  value="-100"/>

    <define name="THETA_PGAIN"  value="-400"/>
    <define name="THETA_DGAIN"  value="-300"/>
    <define name="THETA_IGAIN"  value="-100"/>

    <define name="PSI_PGAIN"  value="-380"/>
    <define name="PSI_DGAIN"  value="-320"/>
    <define name="PSI_IGAIN"  value="-75"/>

    <!-- feedforward -->
    <define name="PHI_DDGAIN"   value=" 300"/>
    <define name="THETA_DDGAIN" value=" 300"/>
    <define name="PSI_DDGAIN"   value=" 300"/>

  </section>

  <section name="INS" prefix="INS_">
    <define name="BARO_SENS" value="15." integer="16"/>
  </section>

  <section name="GUIDANCE_V" prefix="GUIDANCE_V_">
   <define name="MIN_ERR_Z"   value="POS_BFP_OF_REAL(-10.)"/>
   <define name="MAX_ERR_Z"   value="POS_BFP_OF_REAL( 10.)"/>
   <define name="MIN_ERR_ZD"  value="SPEED_BFP_OF_REAL(-10.)"/>
   <define name="MAX_ERR_ZD"  value="SPEED_BFP_OF_REAL( 10.)"/>
   <define name="MAX_SUM_ERR" value="2000000"/>
   <define name="HOVER_KP"    value="-500"/>
   <define name="HOVER_KD"    value="-200"/>
   <define name="HOVER_KI"    value="-100"/>
   <!-- 1.5m/s for full stick : BOOZ_SPEED_I_OF_F(1.5) / (MAX_PPRZ/2) -->
   <define name="RC_CLIMB_COEF" value ="163"/>
   <!-- BOOZ_SPEED_I_OF_F(1.5) * 20% -->
   <define name="RC_CLIMB_DEAD_BAND" value ="160000"/>
<!--   <define name="INV_M" value ="0.2"/> -->
  </section>
 <section name="GUIDANCE_H" prefix="GUIDANCE_H_">
   <define name="PGAIN" value="-100"/>
   <define name="DGAIN" value="-100"/>
   <define name="IGAIN" value="-0"/>
 </section>

 <section name="BAT">
   <define name="MILLIAMP_PER_PERCENT" value="0.86"/>
   <define name="CATASTROPHIC_BAT_LEVEL" value="9.3" unit="V"/>
   <define name="BATTERY_SENS"   value="0.48" integer="16"/>
 </section>
 <section name="AUTOPILOT">
   <define name="MODE_MANUAL" value="AP_MODE_ATTITUDE_DIRECT"/>
   <define name="MODE_AUTO1"  value="AP_MODE_ATTITUDE_Z_HOLD"/>
   <define name="MODE_AUTO2"  value="AP_MODE_NAV"/>
 </section>

 <section name="FMS">

 </section>

 <section name="MISC">
   <define name="FACE_REINJ_1"  value="1024"/>
 </section>

 <section name="SIMULATOR" prefix="NPS_">
   <define name="ACTUATOR_NAMES"  value="{&quot;front_motor&quot;, &quot;back_motor&quot;,
&quot;right_motor&quot;, &quot;left_motor&quot;}"/>
   <define name="INITIAL_CONDITITONS" value="&quot;reset00&quot;"/>
   <define name="SENSORS_PARAMS" value="&quot;nps_sensors_params_booz2_a1.h&quot;"/>
 </section>

  <firmware name="rotorcraft">
    <target name="ap" board="booz_1.0"/>
    <target name="sim" board="pc">
      <subsystem name="fdm"         type="nps"/>
    </target>
    <subsystem name="radio_control" type="ppm"/>
    <subsystem name="telemetry"     type="transparent"/>
    <subsystem name="actuators"     type="mkk"/>
    <subsystem name="imu"           type="b2_v1.0"/>
    <subsystem name="gps"           type="ublox"/>
    <subsystem name="stabilization"   type="euler"/>
    <subsystem name="ahrs"          type="int_cmpl_euler"/>
    <subsystem name="ins"           type="hff"/>
  </firmware>

  <firmware name="booz_test_progs">
    <target name="test_telemetry"     board="booz_1.0"/>
    <target name="test_baro"          board="booz_1.0"/>
    <target name="test_rc_spektrum"   board="booz_1.0"/>
    <target name="test_rc_ppm"        board="booz_1.0"/>
    <target name="test_esc_mkk_simple" board="booz_1.0"/>
    <target name="test_actuators_mkk" board="booz_1.0"/>
    <target name="test_ami601"        board="booz_1.0"/>
  </firmware>

</airframe>
```

```xml
<!-- Flight Plan xml-->

<!-- In this project, the Flight Plan was called EETAC since the flight will be carried out
in that faculty of the UPC (Universitat Politècnica de Catalunya from Spain).-->

<!DOCTYPE flight_plan SYSTEM "flight_plan.dtd">

<flight_plan alt="15" ground_alt="0" lat0="41.274955" lon0="1.984608" max_dist_from_home="20"
name="EETAC" security_height="5">
  <waypoints>
    <waypoint alt="10.0" name="HOME" x="2.8" y="3.6"/>
    <waypoint alt="10.0" name="wp1" x="-14.7" y="-1.0"/>
    <waypoint alt="10.0" name="wp2" x="16.4" y="6.8"/>
    <waypoint alt="10.0" name="wp3" x="-3.5" y="19.4"/>
    <waypoint alt="10.0" name="wp4" x="5.5" y="-11.9"/>
  </waypoints>
  <includes>
    <include name="EETAC" procedure="EETAC_sectors.xml"/>
  </includes>
  <exceptions>
    <exception cond="Or(! InsideEETAC(GetPosX(), GetPosY()), GetPosAlt() > ground_alt + 30)"
deroute="Stay Home"/>
  </exceptions>
  <blocks>
    <block name="Stay Home">
      <stay wp="HOME"/>
    </block>
    <block name="Puntos">
      <go wp="wp1"/>
      <go wp="wp3"/>
      <go wp="wp4"/>
      <go wp="wp2"/>
    </block>
    <block name="casa">
      <go wp="HOME"/>
    </block>
  </blocks>
</flight_plan>
```

```
<!-- Flight Plan sectors-->

<!-- This file was created as a "support" of the flight plan.
 Here are detailed the waypoints that belongs to the exception in the Flight Plan.
 Remember that waypoints with an underscore "_" before will only be displayed in edition module -->

<!DOCTYPE procedure SYSTEM "flight_plan.dtd">
<procedure>

  <waypoints>
    <waypoint name="_1" x="25.1" y="-3.1"/>
    <waypoint name="_2" x="16.5" y="-16.0"/>
    <waypoint name="_3" x="1.8" y="-20.0"/>
    <waypoint name="_4" x="-11.1" y="-13.9"/>
    <waypoint name="_5" x="-16.6" y="-4.4"/>
    <waypoint name="_6" x="-17.0" y="11.6"/>
    <waypoint name="_7" x="-5.2" y="22.8"/>
    <waypoint name="_8" x="7.6" y="24.0"/>
    <waypoint name="_9" x="18.8" y="20.6"/>
    <waypoint name="_10" x="25.2" y="10.5"/>
   </waypoints>

  <sectors>
    <sector name="EETAC" color="red" >
      <corner name="_1"/>
      <corner name="_2"/>
      <corner name="_3"/>
      <corner name="_4"/>
      <corner name="_5"/>
      <corner name="_6"/>
      <corner name="_7"/>
      <corner name="_8"/>
      <corner name="_9"/>
      <corner name="_10"/>

    </sector>
  </sectors>

</procedure>
```

```xml
<!-- Airframe xml for AP flight -->

<!-- The airframe file has been named UPC. We based our changes on the existing airframe booz2_g1 -->


<airframe name="BOOZ2_G1">

  <modules main_freq="512">
      <!--load name="sys_mon.xml"/-->
  </modules>

  <firmware name="rotorcraft">
    <define name="USE_INS_NAV_INIT"/>
    <define name="USE_ADAPT_HOVER"/>
    <define name="NO_FUCKING_STARTUP_DELAY"/>

    <target name="ap" board="booz_1.0">
      <define name="FAILSAFE_GROUND_DETECT"/>
      <define name="USE_GPS_ACC4R"/>
      <define name="BOOZ_START_DELAY" value="3"/>
    </target>
    <target name="sim" board="pc">
      <subsystem name="fdm"         type="nps"/>
      <!--define name="NPS_NO_SUPERVISION"/-->
    </target>

    <subsystem name="radio_control" type="ppm"/>
    <subsystem name="telemetry"     type="transparent_usb"/>
    <subsystem name="actuators"     type="asctec_v2"/>
    <subsystem name="imu"           type="b2_v1.1"/>
    <subsystem name="gps"           type="ublox"/>
    <subsystem name="stabilization"   type="euler"/>
    <subsystem name="ahrs"          type="int_cmpl_euler"/>
    <subsystem name="ins"           type="hff"/>
  </firmware>

  <firmware name="booz2_test_progs">
    <target name="test_telemetry"   board="booz_1.0"/>
    <target name="test_baro"        board="booz_1.0"/>
    <target name="test_rc_spektrum" board="booz_1.0"/>
    <target name="test_rc_ppm"      board="booz_1.0"/>
    <target name="test_micromag"      board="booz_1.0"/>
  </firmware>

  <servos min="0" neutral="0" max="0xff">
    <servo name="FRONT"  no="0" min="0" neutral="0" max="255"/>
    <servo name="BACK"   no="1" min="0" neutral="0" max="255"/>
    <servo name="LEFT"    no="2" min="0" neutral="0" max="255"/>
    <servo name="RIGHT" no="3" min="0" neutral="0" max="255"/>
  </servos>

  <commands>
    <axis name="PITCH"  failsafe_value="0"/>
    <axis name="ROLL"   failsafe_value="0"/>
    <axis name="YAW"    failsafe_value="0"/>
    <axis name="THRUST" failsafe_value="0"/>
  </commands>

  <section name="SUPERVISION" prefix="SUPERVISION_">
    <define name="MIN_MOTOR" value="2"/>
    <define name="MAX_MOTOR" value="200"/>
    <define name="TRIM_A" value="0"/>
    <define name="TRIM_E" value="0"/>
    <define name="TRIM_R" value="0"/>
    <define name="NB_MOTOR" value="4"/>
    <define name="SCALE" value="256"/>
    <define name="ROLL_COEF"  value="{    0,    0, -256,  256}"/>
    <define name="PITCH_COEF" value="{  256, -256,    0,    0}"/>
    <define name="YAW_COEF"   value="{ -256, -256,  256,  256}"/>
    <define name="THRUST_COEF" value="{ 256,  256,  256,  256}"/>
  </section>

  <section name="IMU" prefix="IMU_">

    <define name="GYRO_P_NEUTRAL" value="33150"/>
    <define name="GYRO_Q_NEUTRAL" value="33300"/>
    <define name="GYRO_R_NEUTRAL" value="31450"/>


    <define name="GYRO_P_SENS" value="1.00" integer="16"/>
    <define name="GYRO_Q_SENS" value="1.00" integer="16"/>
    <define name="GYRO_R_SENS" value="1.00" integer="16"/>

    <define name="ACCEL_X_SENS" value="2.562148" integer="16"/>
    <define name="ACCEL_Y_SENS" value="2.560994" integer="16"/>
    <define name="ACCEL_Z_SENS" value="2.563054" integer="16"/>

    <define name="ACCEL_X_NEUTRAL" value="32785"/>
    <define name="ACCEL_Y_NEUTRAL" value="32537"/>
    <define name="ACCEL_Z_NEUTRAL" value="32534"/>

    <define name="MAG_X_NEUTRAL" value="-12"/>
    <define name="MAG_Y_NEUTRAL" value="-10"/>
    <define name="MAG_Z_NEUTRAL" value="-11"/>

    <define name="MAG_X_SENS" value="22.008352" integer="16"/>
    <define name="MAG_Y_SENS" value="21.79885" integer="16"/>
    <define name="MAG_Z_SENS" value="14.675745" integer="16"/>

    <define name="BODY_TO_IMU_PHI"   value="RadOfDeg(1.3)"/>
    <define name="BODY_TO_IMU_THETA" value="RadOfDeg(-2.6)"/>
    <define name="BODY_TO_IMU_PSI"   value="RadOfDeg(0.)"/>

  </section>

  <section name="STABILIZATION_RATE" prefix="STABILIZATION_RATE_">

    <define name="SP_MAX_P" value="10000"/>
    <define name="SP_MAX_Q" value="10000"/>
    <define name="SP_MAX_R" value="10000"/>

    <define name="GAIN_P" value="-400"/>
    <define name="GAIN_Q" value="-400"/>
    <define name="GAIN_R" value="-350"/>

  </section>

  <section name="STABILIZATION_ATTITUDE" prefix="STABILIZATION_ATTITUDE_">

    <!-- setpoints -->
    <define name="SP_MAX_PHI"     value="RadOfDeg(45.)"/>
    <define name="SP_MAX_THETA"   value="RadOfDeg(45.)"/>
    <define name="SP_MAX_R"       value="RadOfDeg(90.)"/>
    <define name="DEADBAND_R"     value="250"/>

     <!-- reference -->
    <define name="REF_OMEGA_P"  value="RadOfDeg(800)"/>
    <define name="REF_ZETA_P"   value="0.85"/>
    <define name="REF_MAX_P"    value="RadOfDeg(300.)"/>
    <define name="REF_MAX_PDOT" value="RadOfDeg(7000.)"/>
    <define name="REF_OMEGA_Q"  value="RadOfDeg(800)"/>
    <define name="REF_ZETA_Q"   value="0.85"/>
    <define name="REF_MAX_Q"    value="RadOfDeg(300.)"/>
    <define name="REF_MAX_QDOT" value="RadOfDeg(7000.)"/>

    <define name="REF_OMEGA_R"  value="RadOfDeg(500)"/>
    <define name="REF_ZETA_R"   value="0.85"/>
    <define name="REF_MAX_R"    value="RadOfDeg(90.)"/>
    <define name="REF_MAX_RDOT" value="RadOfDeg(900.)"/>

   <!-- feedback -->
    <define name="PHI_PGAIN"  value="-2000"/>
    <define name="PHI_DGAIN"  value="-400"/>
    <define name="PHI_IGAIN"  value="-200"/>

    <define name="THETA_PGAIN"  value="-2000"/>
    <define name="THETA_DGAIN"  value="-400"/>
    <define name="THETA_IGAIN"  value="-200"/>

<!-- I changed the PSI_PGain from -2000 to 0 due to MAG error-->
<!-- I changed the PSI_DGain from -400 to 0 due to MAG error-->
<!-- I changed the PSI_IGain from -10 to 0 due to MAG error-->

    <define name="PSI_PGAIN"  value="0"/>
    <define name="PSI_DGAIN"  value="0"/>
    <define name="PSI_IGAIN"  value="0"/>

    <!-- feedforward -->
<!-- I changed the PSI_DDGain from 300 to 0 due to MAG error-->
    <define name="PHI_DDGAIN"   value=" 300"/>
    <define name="THETA_DDGAIN" value=" 300"/>
    <define name="PSI_DDGAIN"   value=" 0"/>

  </section>

  <section name="INS" prefix="INS_">
    <define name="BARO_SENS" value="15." integer="16"/>
    <define name="UNTILT_ACCEL" value="1"/>
    <define name="SONAR_SENS" value="2.146" integer="16"/>
  </section>

  <section name="GUIDANCE_V" prefix="GUIDANCE_V_">
    <define name="MIN_ERR_Z"    value="POS_BFP_OF_REAL(-10.)"/>
    <define name="MAX_ERR_Z"    value="POS_BFP_OF_REAL( 10.)"/>
    <define name="MIN_ERR_ZD"   value="SPEED_BFP_OF_REAL(-10.)"/>
    <define name="MAX_ERR_ZD"   value="SPEED_BFP_OF_REAL( 10.)"/>
    <define name="MAX_SUM_ERR"  value="2000000"/>
    <define name="REF_MIN_ZDD"  value="-1.5*9.81"/>
    <define name="REF_MAX_ZDD"  value=" 0.5*9.81"/>
    <define name="REF_MIN_ZD"   value="-1.5"/>
    <define name="REF_MAX_ZD"   value=" 1.5"/>
    <define name="HOVER_KP"     value="-150"/>
    <define name="HOVER_KD"     value="-80"/>
    <define name="HOVER_KI"     value="0"/>
    <!-- 1.5m/s for full stick : SPEED_BFP_OF_REAL(1.5) / (MAX_PPRZ/2) -->
    <define name="RC_CLIMB_COEF" value ="163"/>
    <!-- SPEED_BFP_OF_REAL(1.5) * 20% -->
    <define name="RC_CLIMB_DEAD_BAND" value ="160000"/>
    <!-- <define name="INV_M" value="0.118"/> -->
  </section>


  <section name="GUIDANCE_H" prefix="GUIDANCE_H_">
    <define name="PGAIN" value="-50"/>
    <define name="DGAIN" value="-100"/>
    <define name="IGAIN" value="-15"/>
    <define name="NGAIN" value="-0"/>
    <!-- feedforward -->
    <define name="AGAIN" value="100"/>
  </section>

  <section name="BAT">
    <define name="MILLIAMP_PER_PERCENT" value="0.86"/>
    <define name="CATASTROPHIC_BAT_LEVEL" value="9.3" unit="V"/>
    <define name="BATTERY_SENS"   value="0.183" integer="16"/>
  </section>

 <section name="AUTOPILOT">
   <define name="MODE_MANUAL" value="AP_MODE_ATTITUDE_DIRECT"/>
   <define name="MODE_AUTO1"  value="AP_MODE_ATTITUDE_Z_HOLD"/>
   <define name="MODE_AUTO2"  value="AP_MODE_KILL"/>
 </section>

  <section name="FMS">
    <define name="BOOZ_FMS_TIMEOUT" value="0"/>
  </section>

  <section name="CAM" prefix="BOOZ_CAM_">
    <define name="ON" value="LED_ON(CAM_SWITCH_LED)"/>
    <define name="OFF" value="LED_OFF(CAM_SWITCH_LED)"/>
    <define name="TILT_NEUTRAL" value="1500"/>
    <define name="TILT_MAX" value="1000"/>
    <define name="TILT_MIN" value="2300"/>
    <define name="TILT_ANGLE_MIN" value="RadOfDeg(-90.)" unit="rad"/>
    <define name="TILT_ANGLE_MAX" value="RadOfDeg( 10.)" unit="rad"/>
    <define name="PAN_NEUTRAL" value="0"/>
    <define name="PAN_MIN" value="0"/>
    <define name="PAN_MAX" value="25736"/> <!-- 360 deg (2^12) -->
    <define name="SetPwm(_v)" value="BoozSetPwm1Value(_v)"/>
  </section>

  <section name="DROP">
    <define name="DROP_SERVO_CLOSED" value="2120"/>
    <define name="DROP_SERVO_OPEN" value="1060"/>
    <!--define name="BoozDropPwm(_v)" value="BoozSetPwm0Value(_v)"/-->
  </section>

  <section name="MISC">
    <define name="BOOZ_ANALOG_BARO_THRESHOLD" value="800"/>
    <define name="FACE_REINJ_1"  value="1024"/>
    <define name="DEFAULT_CIRCLE_RADIUS" value="10."/>
    <!--define name="IMU_MAG_OFFSET" value="-5.8"/-->
  </section>

  <section name="GCS">
    <define name="ALT_SHIFT_PLUS_PLUS" value="5"/>
    <define name="ALT_SHIFT_PLUS" value="1"/>
    <define name="ALT_SHIFT_MINUS" value="-1"/>
  </section>

  <section name="SIMULATOR" prefix="NPS_">
    <define name="ACTUATOR_NAMES"  value="{&quot;front_motor&quot;, &quot;back_motor&quot;,
&quot;right_motor&quot;, &quot;left_motor&quot;}"/>
      <define name="INITIAL_CONDITITONS" value="&quot;reset00&quot;"/>
    <define name="SENSORS_PARAMS" value="&quot;nps_sensors_params_booz2_a1.h&quot;"/>
  </section>

</airframe>
```

```xml
<!-- Reset00 xml -->

<!-- Remember to set up correctly the coordinates in this file.  -->

<?xml version="1.0"?>
<initialize name="reset00">
  <ubody unit="FT/SEC"> 0.0 </ubody>
  <vbody unit="FT/SEC"> 0.0 </vbody>
  <wbody unit="FT/SEC"> 0.0 </wbody>
  <phi   unit="DEG"> 0.0 </phi>
  <theta unit="DEG"> 0.0 </theta>
  <psi   unit="DEG"> 0.0 </psi>
<!-- latitude in geocentric coordinates!! -->
  <latitude  unit="DEG">41.084512</latitude>
  <!--<latitude  unit="DEG">  37.6136 </latitude> -->
  <longitude unit="DEG">1.984611</longitude>
<!-- altitude is above ground level AGL -->
<!--  <altitude  unit="M"> 0.11 </altitude> -->
<!-- altitudeMSL is above sea level ASL -->
  <altitudeMSL unit="M"> 2.5 </altitudeMSL>
<!--  <altitude  unit="M">95 </altitude> -->
  <winddir unit="DEG"> 0.0 </winddir>
  <vwind unit="FT/SEC"> 0.0 </vwind>
</initialize>
```

```xml
<!-- Conf xml -->

<!-- Here you can see the files associated with the simulation (BOOZ2_A1) and with AP flight (UPC) -->

<conf>
  <aircraft
   name="BOOZ2_A1"
   ac_id="150"
   airframe="airframes/Poine/booz2_a1.xml"
   radio="radios/cockpitSX.xml"
   telemetry="telemetry/telemetry_booz2.xml"
   flight_plan="flight_plans/EETAC.xml"
   settings=" settings/settings_booz2.xml settings/settings_booz2_ahrs_cmpl.xml"
   gui_color="white"
  />
  <aircraft
   name="BOOZ2_A7"
   ac_id="159"
   airframe="airframes/Poine/booz2_a7.xml"
   radio="radios/cockpitSX.xml"
   telemetry="telemetry/telemetry_booz2.xml"
   flight_plan="flight_plans/dummy.xml"
   settings=" settings/settings_booz2.xml"
   gui_color="white"
  />
  <aircraft
   name="BOOZ2_A8"
   ac_id="160"
   airframe="airframes/Poine/booz2_a8.xml"
   radio="radios/cockpitSX.xml"
   telemetry="telemetry/telemetry_test_passthrough.xml"
   flight_plan="flight_plans/dummy.xml"
   settings=" settings/settings_test_passthrough.xml"
   gui_color="white"
  />
  <aircraft
   name="EasyStar_ETS"
   ac_id="8"
   airframe="airframes/easystar_ets_example.xml"
   radio="radios/cockpitSX.xml"
   telemetry="telemetry/default.xml"
   flight_plan="flight_plans/versatile.xml"
   settings=" settings/tuning.xml settings/infrared.xml"
   gui_color="red"
  />
  <aircraft
   name="Hexa_LisaL"
   ac_id="153"
   airframe="airframes/Poine/h_hex.xml"
   radio="radios/cockpitSX.xml"
   telemetry="telemetry/telemetry_booz2.xml"
   flight_plan="flight_plans/dummy.xml"
   settings=" settings/settings_booz2.xml"
   gui_color="white"
  />
  <aircraft
   name="LISA_ASCTEC_PIOTR"
   ac_id="161"
   airframe="airframes/esden/lisa_asctec.xml"
   radio="radios/cockpitSX.xml"
   telemetry="telemetry/telemetry_booz2.xml"
   flight_plan="flight_plans/dummy.xml"
   settings=" settings/settings_booz2.xml settings/settings_booz2_ahrs_cmpl.xml"
   gui_color="white"
  />
  <aircraft
   name="Microjet"
   ac_id="5"
   airframe="airframes/microjet_example.xml"
   radio="radios/cockpitMM.xml"
   telemetry="telemetry/default.xml"
   flight_plan="flight_plans/basic.xml"
   settings=" settings/basic_infrared.xml"
   gui_color="#6293ba"
  />
  <aircraft
   name="Tiny_IMU"
   ac_id="7"
   airframe="airframes/example_twog_analogimu.xml"
   radio="radios/cockpitSX.xml"
   telemetry="telemetry/default_fixedwing_imu.xml"
   flight_plan="flight_plans/versatile.xml"
   settings=" settings/tuning_ins.xml"
   gui_color="blue"
  />
  <aircraft
   name="Twinjet"
   ac_id="6"
   airframe="airframes/twinjet_example.xml"
   radio="radios/cockpitMM.xml"
   telemetry="telemetry/default.xml"
   flight_plan="flight_plans/versatile.xml"
   settings=" settings/tuning.xml settings/infrared.xml"
   gui_color="#ba6293"
  />
  <aircraft
   name="UPC"
   ac_id="1"
   airframe="airframes/UPC.xml"
   radio="radios/FUT7C.xml"
   telemetry="telemetry/telemetry_booz2.xml"
   flight_plan="flight_plans/EETAC.xml"
   settings=" settings/settings_booz2.xml"
   gui_color="blue"
  />
</conf>
```

```
<!-- Control Panel xml-->

<!-- Here are the settings for all the sessions and the available running agents -->

<?xml version="1.0"?>
<control_panel name="paparazzi control panel">
  <section name="variables">
    <variable name="downlink_serial_port" value="/dev/ttyUSB0"/>
    <variable name="fbw_serial_port"      value="/dev/ttyS1"/>
    <variable name="ap_serial_port"       value="/dev/ttyS0"/>
    <variable name="ivy_bus"              value="127:2010"/>
    <variable name="map"                  value="muret_UTM.xml"/>
    <variable name="flight_plan"          value="flight_plans/EETAC.xml"/>
  </section>
  <section name="programs">
    <program name="Server" command="sw/ground_segment/tmtc/server">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Simulator Booz" command="var/BOOZ2_A1/sim/simsitl"/>
    <program name="Data Link" command="sw/ground_segment/tmtc/link">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="GCS" command="sw/ground_segment/cockpit/gcs">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Flight Plan Editor" command="sw/ground_segment/cockpit/gcs -edit">
    </program>
    <program name="Messages" command="sw/ground_segment/tmtc/messages">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Messages (Python)" command="sw/ground_segment/python/messages_app/messagesapp.py"/>
    <program name="Settings" command="sw/ground_segment/tmtc/settings">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Settings (Python)" command="sw/ground_segment/python/settings_app/settingsapp.py"/>
    <program name="GPSd position display" command ="sw/ground_segment/tmtc/gpsd2ivy"/>
    <program name="Log Plotter" command ="sw/logalizer/plot"/>
    <program name="Real-time Plotter" command ="sw/logalizer/plotter"/>
    <program name="Log File Player" command="sw/logalizer/play">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Simulator" command="sw/simulator/launchsitl">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Joystick" command="sw/ground_segment/joystick/input2ivy">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Hardware in the Loop" command="sw/simulator/simhitl">
      <arg flag="-fbw" variable="fbw_serial_port"/>
      <arg flag="-ap" variable="ap_serial_port"/>
    </program>
    <program name="Environment Simulator" command="sw/simulator/gaia">
      <arg flag="-b" variable="ivy_bus"/>
    </program>
    <program name="Http Server" command="sw/ground_segment/tmtc/boa"/>
    <program name="Plot Meteo Profile" command="sw/logalizer/plotprofile"/>
    <program name="Weather Station" command="sw/ground_segment/misc/davis2ivy">
      <arg flag="-b" variable="ivy_bus"/>
      <arg flag="-d" constant="/dev/ttyUSB1"/>
    </program>
 </section>

  <section name="sessions">
   <session name="Flight USB-serial@9600">
```

```
      <program name="Data Link">
        <arg flag="-d" constant="/dev/ttyUSB0"/>
      </program>
      <program name="Server"/>
      <program name="GCS"/>
    </session>

    <session name="Simulation Booz">
      <program name="GCS"/>
      <program name="Server">
        <arg flag="-n"/>
      </program>
      <program name="Simulator Booz"/>
    </session>

   <session name="Flight USB-serial@57600">
     <program name="Data Link">
       <arg flag="-d" constant="/dev/ttyACM0"/>
       <arg flag="-s" constant="57600"/>
     </program>
     <program name="Server"/>
     <program name="GCS"/>
   </session>
   <session name="Flight USB-XBee-API@57600">
      <program name="Data Link">
        <arg flag="-d" constant="/dev/paparazzi/xbee"/>
        <arg flag="-transport" constant="xbee"/>
        <arg flag="-s" constant="57600"/>
      </program>
      <program name="Server"/>
      <program name="GCS"/>
    </session>
    <session name="HITL">
      <program name="Hardware in the Loop">
        <arg flag="-a" constant="HITL"/>
        <arg flag="-noground"/>
        <arg flag="-boot"/>
      </program>
      <program name="GCS"/>
      <program name="Data Link">
        <arg flag="-s" constant="57600"/>
      </program>
      <program name="Server"/>
    </session>
    <session name="Booz simulation : AHRS">
     <program name="Messages">
        <arg flag="-c" constant="telemetry"/>
     </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="rate : p"/>
        <arg flag="-g" constant="0+450:830x450"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_RATE_ATTITUDE:p'"/>
        <arg flag="-c" constant="'*:telemetry:IMU_GYRO:gp:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_RATE_LOOP:est_p:57.3'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="rate : q"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_RATE_ATTITUDE:q'"/>
        <arg flag="-c" constant="'*:telemetry:IMU_GYRO:gq:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_RATE_LOOP:est_q:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_RATE_LOOP:sp_q:57.3'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="rate : r"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_RATE_ATTITUDE:r'"/>
        <arg flag="-c" constant="'*:telemetry:IMU_GYRO:gr:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_RATE_LOOP:est_r:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_RATE_LOOP:sp_r:57.3'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="attitude : phi"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_RATE_ATTITUDE:phi'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_ATT_LOOP:est_phi:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_ATT_LOOP:sp_phi:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_MEASURE:phi:57.3'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="attitude : theta"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_RATE_ATTITUDE:theta'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_ATT_LOOP:est_theta:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_ATT_LOOP:sp_theta:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_MEASURE:theta:57.3'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="attitude : psi"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_RATE_ATTITUDE:psi'"/>
<!--        <arg flag="-c" constant="'*:telemetry:BOOZ_ATT_LOOP:est_psi:57.3'"/> -->
<!--        <arg flag="-c" constant="'*:telemetry:BOOZ_ATT_LOOP:sp_psi:57.3'"/> -->
        <arg flag="-c" constant="'*:telemetry:AHRS_MEASURE:psi:57.3'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="speed : u v w"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_SPEED_POS:u'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_SPEED_POS:v'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_SPEED_POS:w'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="position : x"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_SPEED_POS:x'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_HOV_LOOP:est_x'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_HOV_LOOP:sp_x'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="position : y"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_SPEED_POS:y'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_HOV_LOOP:est_y:'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_HOV_LOOP:sp_y:'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="position : z"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_SPEED_POS:z'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_VERT_LOOP:est_z'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_VERT_LOOP:sp_z'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="ahrs : bp bq br"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_STATE:bp:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_STATE:bq:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_STATE:br:57.3'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_GYRO_BIAS:bp'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_GYRO_BIAS:bq'"/>
        <arg flag="-c" constant="'*:telemetry:BOOZ_SIM_GYRO_BIAS:br'"/>
    </program>
    <program name="Real-time Plotter">
        <arg flag="-t" constant="ahrs : covariances"/>
        <arg flag="-u" constant="0.1"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_phi_phi'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_phi_bp'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_bp_bp'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_theta_theta'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_theta_bq'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_bq_bq'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_psi_psi'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_psi_br'"/>
        <arg flag="-c" constant="'*:telemetry:AHRS_EULER_COV:p_br_br'"/>
    </program>
    </session>
  </section>
</control_panel>
```

```xml
<!-- Radio xml for Simulation -->

<?xml version="1.0"?>

<!DOCTYPE radio SYSTEM "radio.dtd">
<radio name="cockpitSX (easy)" data_min="900" data_max="2100" sync_min ="5000" sync_max ="15000"
pulse_type="POSITIVE">
 <channel ctl="D" function="ROLL"     max="2050" neutral="1500" min="950" average="0"/>
 <channel ctl="C" function="PITCH"    min="2050" neutral="1500" max="950" average="0"/>
 <channel ctl="B" function="YAW"      min="2050" neutral="1500" max="950" average="0"/>
 <channel ctl="A" function="THROTTLE" min="1223" neutral="1223" max="2050" average="0"/>
 <channel ctl="G" function="UNUSED"    min="2050" neutral="1500" max="950" average="1"/>
 <channel ctl="E" function="GAIN1"     min="2050" neutral="1496" max="948" average="1"/>
 <channel ctl="PHAUX2" function="MODE"    min="2050" neutral="1500" max="948" average="1"/>
</radio>
```

```
<!-- FUT7C xml -->

<!-- Radio Settings for the MEEKPE Encoder Board. Remember to add three empty channels if you use the
 ATMEGA328 Encoder-->


<!DOCTYPE radio SYSTEM "radio.dtd">
<radio name="T7chp" data_min="900" data_max="2100" sync_min="5000" sync_max="15000" pulse_type="POSITIVE">
  <channel ctl="right_stick_horiz" function="ROLL" min="1939" neutral="1525" max="1114" average="0"/>
  <channel ctl="right_stick_vert" function="PITCH" min="1112" neutral="1518" max="1937" average="0"/>
  <channel ctl="left_stick_vert" function="THROTTLE" min="1940" neutral="1940" max="1115" average="0"/>
  <channel ctl="left_stick_horiz" function="YAW" min="1940" neutral="1520" max="1115" average="0"/>
  <channel ctl="switch_E" function="MODE" min="966" neutral="1526" max="2079" average="10"/>
</radio>
```

```
<!-- Settings xml-->

<!-- File used in Simulation and AP flight -->


<!DOCTYPE settings SYSTEM "settings.dtd">

<settings>
  <dl_settings>

    <dl_settings NAME="Misc">
       <dl_setting var="telemetry_mode_Main_DefaultChannel" min="0" step="1" max="11" module="telemetry"
shortname="telemetry" values="Default|PPM|Raw|Scaled|AHRS|Rate|Attitude|Vertical|Horizontal|Aligner|
HS_att_roll|Tune_hover">
         <key_press key="d" value="0"/>
         <key_press key="v" value="7"/>
         <key_press key="h" value="8"/>
       </dl_setting>
       <dl_setting var="autopilot_mode_auto2" min="0" step="1" max="12" module="autopilot"
shortname="auto2" values="Fail|Kill|Rate|Att|Rate_rcC|Att_rcC|Att_C|Rate_Z|Att_Z|Hover|Hover_C|Hover_Z|
Nav"/>
      <dl_setting var="kill_throttle" min="0" step="1" max="1" module="autopilot" values="Resurrect|Kill"
handler="KillThrottle"/>
      <dl_setting var="autopilot_power_switch" min="0" step="1" max="1" module="autopilot" values="OFF|
ON" handler="SetPowerSwitch">
        <strip_button name="POWER ON" icon="on.png" value="1"/>
        <strip_button name="POWER OFF" icon="off.png" value="0"/>
      </dl_setting>
      <dl_setting var="autopilot_rc" min="0" step="1" max="1" module="autopilot" values="RC OFF|RC ON">
        <strip_button name="RC ON" value="1"/>
        <strip_button name="RC OFF" value="0"/>
      </dl_setting>
    </dl_settings>

    <dl_settings NAME="Rate Loop">
      <dl_setting var="stabilization_rate_gain.p" min="-1000" step="1" max="-1" module="stabilization/
stabilization_rate" shortname="gain p"/>
      <dl_setting var="stabilization_rate_gain.q" min="-1000" step="1" max="-1" module="stabilization/
stabilization_rate" shortname="gain q"/>
      <dl_setting var="stabilization_rate_gain.r" min="-1000" step="1" max="-1" module="stabilization/
stabilization_rate" shortname="gain r"/>
    </dl_settings>


    <dl_settings NAME="Att Loop">
      <dl_setting var="stabilization_gains.p.x" min="-4000" step="1" max="-1"   module="stabilization/
stabilization_attitude" shortname="pgain phi" />
      <dl_setting var="stabilization_gains.d.x" min="-4000" step="1" max="-1"   module="stabilization/
stabilization_attitude" shortname="dgain p"/>
      <dl_setting var="stabilization_gains.i.x" min="-300"  step="1" max="0"    module="stabilization/
stabilization_attitude" shortname="igain phi" handler="SetKiPhi"/>
      <dl_setting var="stabilization_gains.dd.x" min="0"    step="1" max="1000" module="stabilization/
stabilization_attitude" shortname="ddgain p"/>
      <dl_setting var="stabilization_gains.p.y" min="-4000" step="1" max="-1"   module="stabilization/
stabilization_attitude" shortname="pgain theta"/>
      <dl_setting var="stabilization_gains.d.y" min="-4000" step="1" max="-1"   module="stabilization/
stabilization_attitude" shortname="dgain q"/>
      <dl_setting var="stabilization_gains.i.y" min="-300"  step="1" max="0"    module="stabilization/
stabilization_attitude" shortname="igain theta"/>
      <dl_setting var="stabilization_gains.dd.y" min="0"    step="1" max="500"  module="stabilization/
stabilization_attitude" shortname="ddgain q"/>
      <dl_setting var="stabilization_gains.p.z" min="-4000" step="1" max="-1"   module="stabilization/
stabilization_attitude" shortname="pgain psi"/>
      <dl_setting var="stabilization_gains.d.z" min="-4000" step="1" max="-1"   module="stabilization/
```

```
stabilization_attitude" shortname="dgain r"/>
      <dl_setting var="stabilization_gains.i.z" min="-300"  step="1" max="0"    module="stabilization/
stabilization_attitude" shortname="igain psi"/>
      <dl_setting var="stabilization_gains.dd.z" min="0"    step="1" max="2000"  module="stabilization/
stabilization_attitude" shortname="ddgain r"/>
    </dl_settings>

    <dl_settings NAME="Vert Loop">
      <dl_setting var="guidance_v_kp" min="-600" step="1" max="0"   module="guidance/guidance_v"
shortname="kp"/>
      <dl_setting var="guidance_v_kd" min="-600" step="1" max="0"   module="guidance/guidance_v"
shortname="kd"/>
      <dl_setting var="guidance_v_ki" min="-300" step="1" max="0"   module="guidance/guidance_v"
shortname="ki" handler="SetKi" />
      <dl_setting var="guidance_v_z_sp" min="-5" step="0.5" max="3" module="guidance/guidance_v"
shortname="sp" unit="2e-8m" alt_unit="m" alt_unit_coef="0.00390625"/>
      <dl_setting var="ins_vf_realign" min="0" step="1" max="1" module="subsystems/ins"
shortname="vf_realign" values="OFF|ON"/>
   </dl_settings>

   <dl_settings NAME="Horiz Loop">
      <dl_setting var="guidance_h_pos_sp.x" MIN="-10" MAX="10" STEP="1"  module="guidance/guidance_h"
shortname="sp_x_ned" unit="1/2^8m" alt_unit="m" alt_unit_coef="0.00390625"/>
      <dl_setting var="guidance_h_pos_sp.y" MIN="-10" MAX="10" STEP="1"  module="guidance/guidance_h"
shortname="sp_y_ned" unit="1/2^8m" alt_unit="m" alt_unit_coef="0.00390625"/>
      <dl_setting var="guidance_h_psi_sp" MIN="-180" MAX="180" STEP="5"  module="guidance/guidance_h"
shortname="sp_psi" unit="1/2^20r" alt_unit="deg" alt_unit_coef="0.000054641513360"/>
      <dl_setting var="guidance_h_pgain" min="-400" step="1" max="0" module="guidance/guidance_h"
shortname="kp"/>
      <dl_setting var="guidance_h_dgain" min="-400" step="1" max="0" module="guidance/guidance_h"
shortname="kd"/>
      <dl_setting var="guidance_h_igain" min="-400" step="1" max="0" module="guidance/guidance_h"
shortname="ki" handler="SetKi"/>
      <dl_setting var="guidance_h_ngain" min="-400" step="1" max="0" module="guidance/guidance_h"
shortname="kn"/>
      <dl_setting var="guidance_h_again" min="-400" step="1" max="0" module="guidance/guidance_h"
shortname="ka"/>
      <dl_setting var="ins_hf_realign" min="0" step="1" max="1" module="subsystems/ins"
shortname="hf_realign" values="OFF|ON"/>
   </dl_settings>

   <dl_settings NAME="NAV">
     <dl_setting var="flight_altitude" MIN="0" STEP="0.1" MAX="400" module="navigation" unit="m"
handler="SetFlightAltitude"/>
     <dl_setting var="nav_heading" MIN="0" STEP="1" MAX="360" module="navigation" unit="1/2^12r"
alt_unit="deg" alt_unit_coef="0.0139882"/>
     <dl_setting var="nav_radius" MIN="-150" STEP="0.1" MAX="150" module="navigation" unit="m"/>
   </dl_settings>


  </dl_settings>
</settings>
```

```
<!-- Settings ahrs xml -->

<!-- File used in Simulation -->

<!DOCTYPE settings SYSTEM "settings.dtd">

<settings>
  <dl_settings>

    <dl_settings NAME="Filter">
       <dl_setting var="ahrs_impl.reinj_1" min="512" step="512" max="262144" module="subsystems/ahrs/
ahrs_int_cmpl_euler" shortname="reinj_1"/>
    </dl_settings>

  </dl_settings>
</settings>
```

```
<!-- Telemetry xml -->

<!-- This file is used for Simulation and AP flight -->


<?xml version="1.0"?>
<!DOCTYPE telemetry SYSTEM "telemetry.dtd">
<telemetry>


  <process name="Main">

    <mode name="default">
      <message name="DL_VALUE"          period="1.1"/>
      <message name="ROTORCRAFT_STATUS"       period="1.2"/>
      <message name="ROTORCRAFT_FP"          period="0.25"/>
      <message name="ALIVE"             period="2.1"/>
      <message name="INS_REF"     period="5.1"/>
      <message name="ROTORCRAFT_NAV_STATUS"  period="1.6"/>
      <message name="WP_MOVED"          period="1.3"/>
      <message name="BOOZ2_CAM"         period="1."/>
      <message name="GPS_INT"         period=".25"/>
      <message name="INS"          period=".25"/>
    </mode>

    <mode name="ppm">
      <message name="BOOZ2_CMD"           period=".05"/>
      <message name="PPM"                 period="0.5"/>
      <message name="RC"                  period="0.5"/>
      <message name="BOOZ2_RADIO_CONTROL" period="0.5"/>
      <message name="ROTORCRAFT_STATUS"       period="1"/>
    </mode>

    <mode name="raw_sensors">
      <message name="ROTORCRAFT_STATUS"       period="1.2"/>
      <message name="DL_VALUE"          period="0.5"/>
      <message name="ALIVE"             period="2.1"/>
      <message name="IMU_ACCEL_RAW"     period=".05"/>
      <message name="IMU_GYRO_RAW"      period=".05"/>
      <message name="IMU_MAG_RAW"       period=".05"/>
      <message name="BARO_RAW"          period=".1"/>
    </mode>

    <mode name="scaled_sensors">
      <message name="ROTORCRAFT_STATUS"       period="1.2"/>
      <message name="DL_VALUE"          period="0.5"/>
      <message name="ALIVE"             period="2.1"/>
      <message name="IMU_GYRO_SCALED"        period=".075"/>
      <message name="IMU_ACCEL_SCALED"       period=".075"/>
      <message name="IMU_MAG_SCALED"         period=".1"/>
    </mode>

    <mode name="ahrs">
      <message name="ROTORCRAFT_STATUS"       period="1.2"/>
      <message name="DL_VALUE"          period="0.5"/>
      <message name="ALIVE"             period="2.1"/>
<!--      <message name="BOOZ2_ALIGNER"     period=".1"/> -->
      <message name="FILTER"      period=".5"/>
<!--      <message name="BOOZ2_AHRS_QUAT"   period=".25"/> -->
      <message name="BOOZ2_AHRS_EULER"  period=".1"/>
<!--      <message name="BOOZ2_AHRS_RMAT"   period=".5"/> -->
    </mode>

    <mode name="rate_loop">
```

```
      <message name="ROTORCRAFT_STATUS"       period="1.2"/>
      <message name="DL_VALUE"          period="0.5"/>
      <message name="ALIVE"             period="2.1"/>
      <message name="RATE_LOOP"   period=".02"/>
    </mode>

    <mode name="attitude_loop">
      <message name="ROTORCRAFT_STATUS"             period="1.2"/>
      <message name="DL_VALUE"                period="0.5"/>
      <message name="ALIVE"                   period="0.9"/>
      <message name="STAB_ATTITUDE"     period=".03"/>
      <message name="STAB_ATTITUDE_REF" period=".03"/>
    </mode>

    <mode name="vert_loop">
      <message name="ROTORCRAFT_STATUS"             period="1.2"/>
      <message name="DL_VALUE"                period="0.5"/>
      <message name="ALIVE"                   period="0.9"/>
      <message name="VFF"               period=".05"/>
      <message name="VERT_LOOP"         period=".05"/>
<!--      <message name="BOOZ2_CMD"               period=".05"/> -->
      <message name="INS"               period=".05"/>
      <message name="INS_REF"           period="5.1"/>
    </mode>

    <mode name="h_loop">
      <message name="ALIVE"                   period="0.9"/>
      <message name="HOVER_LOOP"        period="0.062"/>
      <message name="STAB_ATTITUDE"     period=".4"/>
      <message name="HFF_DBG"           period=".2"/>
      <!--<message name="STAB_ATTITUDE_REF" period=".4"/>-->
      <message name="ROTORCRAFT_FP"               period="0.8"/>
      <message name="ROTORCRAFT_STATUS"             period="1.2"/>
      <message name="ROTORCRAFT_NAV_STATUS"       period="1.6"/>
          <message name="HFF_GPS"           period=".03"/>
      <message name="INS_REF"           period="5.1"/>
    </mode>

    <mode name="aligner">
      <message name="ALIVE"                   period="0.9"/>
      <message name="FILTER_ALIGNER"    period="0.02"/>
    </mode>

    <mode name="hs_att_roll">
      <message name="ROTORCRAFT_STATUS"                 period="1.2"/>
      <message name="ALIVE"                       period="0.9"/>
      <message name="DL_VALUE"                    period="0.5"/>
<!--      <message name="STAB_ATTITUDE_HS_ROLL" period="0.02"/> -->
    </mode>

    <mode name="tune_hover">
      <message name="DL_VALUE"          period="1.1"/>
      <message name="ROTORCRAFT_STATUS"       period="1.2"/>
      <message name="ALIVE"             period="2.1"/>
      <!--<message name="BOOZ2_SONAR"       period="0.1"/>-->
      <!--<message name="BOOZ2_TUNE_HOVER"              period=".1"/>-->
      <!-- <message name="BOOZ2_GPS"              period=".20"/> -->
      <!--<message name="INS2"              period=".05"/>
      <message name="INS3"              period=".20"/>-->
      <message name="INS_REF"           period="5.1"/>
    </mode>


  </process>

</telemetry>
```