This Bachelor Thesis is intended to provide an existing AX25 modem with the necessary mechanisms to decode and evaluate real signals from satellites, in order to exchange photos and telemetry.

Antonio Jiménez Hernández was born in Águilas (Murcia) in 1996. With this Bachelor Thesis complements his education in others specialities and finalizes his Bachelor's Degree in Telecommunications Engineering at the University of Granada (Spain).

Andrés María Roldán Aranda is the academic head of the present project, and the student's tutor. He is professor in the Departament of Electronics and Computers Technologies

Copy for the student / Copia para el alumno

BACHELOR THESIS

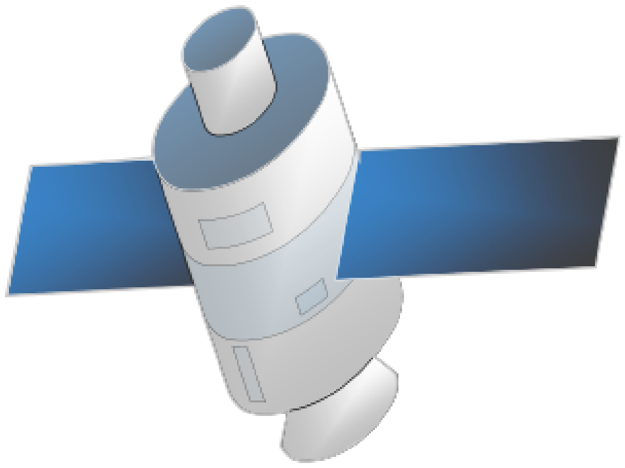
Design and implementation of a cubesat data exchange platform via radio"

Antonio Jiménez Hernández

TELECOMMUNICATIONS ENGINEERING

UNIVERSITY OF GRANADA

Bachelor of Telecommunications Engineering



Bachelor Thesis

Design and implementation of a cubesat data exchange platform via radio

Antonio Jiménez Hernández

2017/2018

Tutor: Andrés María Roldán Aranda

GRADO EN INGENIERÍA DE
TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

"Design and implementation of a cubesat data exchange platform via radio"

CURSO: 2017/2018

Antonio Jiménez Hernández

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

"Design and implementation of a cubesat data exchange platform via radio"

REALIZADO POR:

Antonio Jiménez Hernández

DIRIGIDO POR:

Andrés María Roldán Aranda

DEPARTAMENTO:

Electrónica y Tecnología de los Computadores

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Grado de D. Antonio Jiménez Hernández,

Informa:

que el presente trabajo, titulado:

"Design and implementation of a cubesat data exchange platform via radio"

ha sido realizado y redactado por el mencionado alumno bajo nuestra dirección, y con esta fecha autorizo a su presentación.

Granada, a 20 de junio de 2018

[illegible]

Fdo. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Grado se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 20 de junio de 2018

[illegible] [illegible]

Fdo. Antonio Jiménez Hernández Fdo. Andrés María Roldán Aranda

Desing and implementation of a cubesat data exchange platform via radio

Antonio Jiménez Hernández

PALABRAS CLAVE:

AX25, FX25, Physical layer, Efecto Doppler, telemetría, FEC, Cubesat, Java , Eclipse .

RESUMEN:

El propósito principal de este proyecto es la introducción de técnicas y mecanismos para la correcta recepción de datos vía satélite, así como la introducción de aplicaciones de telemetría para la evaluación del funcionamiento del receptor en el presente proyecto y en continuaciones de éste. Dicho sistema será usado en el proyecto GranaSAT, implementándose tanto en el satélite como en la estación terrestre.

Este trabajo es presentado como el Trabajo Final de Grado del grado de Ingeniería de Telecomunicaciones de la Universidad de Granada.

En este proyecto se ha evaluado el funcionamiento inicial del sistema diseñado por Lucía Castañeda González [**?**], en el cual solo se tienen en cuenta la segunda y tercera capa del modelo OSI. El sistema se diseñó para transmisión vía cable, sin tener en cuenta las características del canal radio. En el presente proyecto se tratarán con más profundidad aspectos relacionados con la primera capa del modelo OSI, introduciendo mecanismos que permitan la correcta transmisión y recepción a través del canal radio.

Design and implementation of data exchange platform via radio for cubesat

Antonio Jiménez Hernández

KEYWORDS:

AX25, FX25, Physical layer, Doppler effect, Telemetry, FEC, Cubesat, Java , Eclipse .

ABSTRACT:

The main purpose of this project is the introduction of techniques and mechanisms for the correct reception of data via satellite, as well as the introduction of telemetry applications for the evaluation of the operation of the receiver, in the present project and in continuations of this. The mentioned system will be used in the project GranaSAT, being implemented in the satellite and in the ground station.

The work is presented as a Degree Final Project for the Degree of Telecommunications Engineering of the University of Granada.

In this project the initial operation of the system designed by Lucía Castañeda González [11] has been evaluated. In that project, it was only taken into account the second and third layers of OSI model. This system was designed for transmission via cable, not taking into account the radio channel characteristics. In the present project, they will be treated more deeply aspects related with the first layer of the OSI model, introducing mechanisms which allow the correct transmission and reception via radio channel.

Agradecimientos

*A mis padres, Antonio y María Ángeles, mi hermano Javier y mis abuelos, Francisco, Francisca, Juan y Damiana, por apoyarme.
A todo el equipo de GranaSAT, en especial a Lucía Castañeda González, por facilitarme los datos necesarios de su proyecto, y a mi tutor, Andrés Roldán Aranda, del que he aprendido mucho durante este proceso.*

Acknowledgements

*To my parents, Antonio and María Ángeles, my brother Javier and my grandparents, Francisco, Francisca, Juan and Damiana, for supporting me.
To all GranaSAT team, specially to Lucía Castañeda González, for issuing all needed data from her project, and to my tutor, Andrés Roldán Aranda, from whom I have learned a lot during this process.*

ACRONYMS

AFC	Automatic Frequency Control
AFSK	Audio Frequency Shift Keying
ARRL	American Radio Relay League
ASCII	American Standard Code for Information Interchange
CRC	Cyclic Redundancy Check
FCS	Frame Check Sequence
FEC	Forward Error Correction
FM	Frequency Modulation
GEO	Geostationary Earth Orbit
ISO	International Standard Organization
LEO	Low Earth Orbit
LFSR	Linear Feedback Shift Register
OSI	Open System Interconnection
PLL	Phase-Locked Loop
RF	Radio Frequency

SSID	Service Set Identifie
UHF	Ultra High Frequency
VCO	Voltage-Controlled Oscillator
VHF	Very High Frequency

GLOSARY

AX25 Protocol used between two amateur radio stations in a pointto-point or networked communications environment. The protocol specifies only link layer and physical layer functions..

Bell 202 The Bell 202 modem was an early (1976) modem standard developed by the Bell Systems. It specifies Audio Frequency Shift Keying (AFSK) to encode and transfer data at a rate of 1200 bits per second, half-duplex.

CRC CITT A Cyclic Redundancy Check is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.CRC CITT is a 16-bits CRC with generator polinomy $G(x) = x^{16} + x^{12} + x^5 + 1$.

Cubesat Name received by satellites whose volume is less than one litre and whose weigth is no more tha 1 kg.

Doppler effect Change in frequency or wavelength of a wave for an observer who is moving relative to the wave source..

FX25 Protocol extension of the AX.25 Link Layer Protocol which includes FEC.

GranaSAT Academic project developed by the University of Granada, whose main target is to launch a cubesat into the space designed by students. It is coordinated by the Professor Andrés María Roldán Aranda. http://granasat.ugr.es/.

INDEX

LIST OF FIGURES

LIST OF VIDEOS

LIST OF TABLES

CHAPTER

1 INTRODUCTION

The present project ends the Telecommunication Engineering Degree. It is a sample of all knowledge acquired during it, and developed with the aim of putting into practice all the concepts learned. The main aim of this project is to provide the existing software, which implemented all the aspects related to the protocol, with a mechanism for correct transmission and reception of data via radio channel, compensating the effects of the channel, as Doppler effect. In addition, telemetry mechanisms were included during the development of this project, such as a time domain signal representation, a spectrum of the signal representation, and an eye diagram of the demodulator output, in order to evaluate the effect of the channel and the transmission and reception of the signal. Finally, the transmission velocity will be increased from 1200 bauds per second, to 9600 bauds per second.

The starting point of this project, is another degree project [11] developed by Lucia Castañeda Gonzalez, which started from a java code developed by Silvan Toledo. Mr Toledo, developed a Java code which was able to receive and transmit AX25 frames. Ms. Castañeda, starting from the Mr Toledo code, implemented an AX25 protocol, as well as a FX25 one, which made the system able to send images and other commands. However, the modem developed by Mr. Toledo was not able to decode signals from satellites, because of that in the present project, demodulator will be improved, to be able to decode real signals from satellites, as well as, the insertion of a telemetry board and the increase of the transmission velocity.

This project is part of a cubesat developed by GranaSAT team, which will be launched into the space in the future.

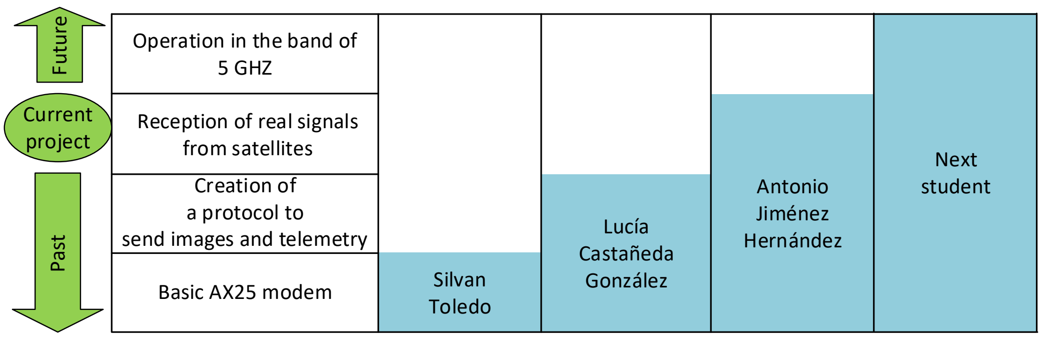

Future
Current project
Past
Operation in the band of 5 GHZ
Reception of real signals from satellites
Creation of a protocol to send images and telemetry
Basic AX25 modem
Silvan Toledo
Lucía Castañeda González
Antonio Jiménez Hernández
Next student

Figure 1.1 – *Project development.*

1.1 State of the art: AX25 implemented in satellites

Nowadays, there are more than 100 satellites transmitting in amateur orbiting around the Earth. Most of them are used as lineal transponders, whose functions are:

- Reception, amplification and retransmission in other frequency band, adapting transmission parameters, as modulation and signal rate to the mentioned band.

- Answer to a required information sent by the ground station.

Most of them which use AX25 are ones which transmit a reduced amount of information, as telemetry data or even sounds and images, used mainly with academical purposes. Examples of them are AO-7,FO-29 or FUNcube. A big amount of satellites transmits in VHF and (145 MHz) for uplink and UHF (435 MHz) for downlink, which is a great chance for GranaSAT to be listened by other ground station. Moreover the band of 145 MHz is used by hams on Earth, so it allows to have access to many transmissions in order to test the receptor capacity before send the satellite.

1.1.1 Cubesat

A Cubesat is a satellite of 10 cm of each side and up to 1 kg of weight. For design it, it is necessary to implement a CPU and a power supply system. Most of they are launched into the LEO orbit, which is the nearest to the Earth, the sourround the Earth several times per day, viewed by the ground station for few minutes. Other satellites as televisions ones, orbit in GEO

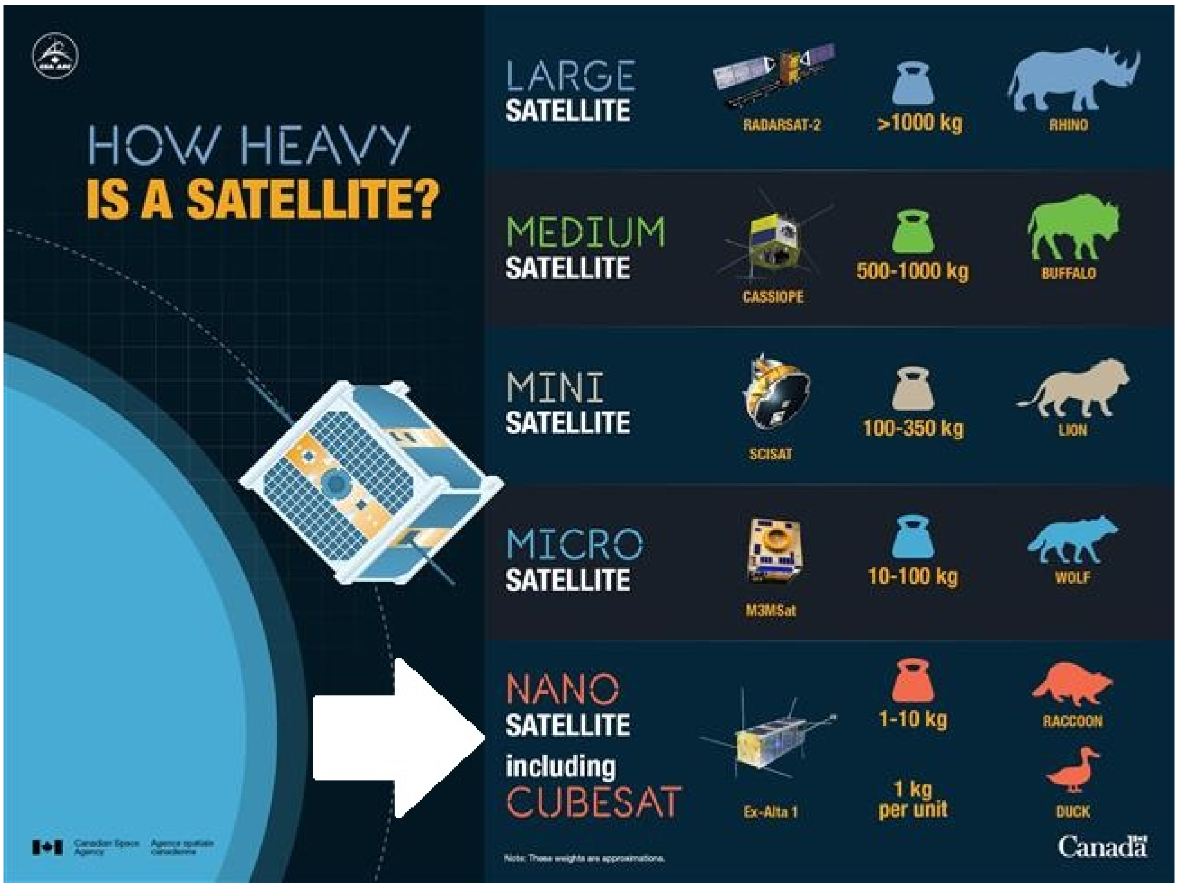


Figure 1.2 – *Dimensions of different satellites [10].*

Main advantages and disadvantages of a cubesat are:

- **Advantages:**
  - **Fast development**. It can be developed in two years.
  - **Cost**. Low prize.
  - **Technology**. Simple, with standard parts available.
  - **Design**. Simple design with short missions. No need to use thermal blankets.
  - **Space debris**. They do not suffer it. They burn up in the atmosphere upon re-entry.
- **Disadvantages:**
  - **Scope**. Reduced, due to the reduced capacity to carry scientific instruments.
  - **Mission duration**. Operational between for a period of 3 to 12 months.

1.2 Motivation

The main objective of this project is to end the Degree of Telecommunication Engineering, but it is also chosen by other reasons. Participating in a project which implements a system which will be a transmitting and receiving in a real scenario ensures me that I am participating in something real, not just in a theoretical project. Moreover, I was looking for something related with signal processing, which is one of the fields where I am specialized. Furthermore, collaborating with a team of student is a great chance to know about different fields involved in the project.

Because of all these reasons, this modem will be implemented in a Cubesat

1.3 Objectives

The project's text is divided into 6 chapters and which describes each part of the development process.

- Making the modem able to receive real signals from satellites.

- Making the modem able to transmit faster.

- Making the modem able to represent the received signal ans parameters related with it.

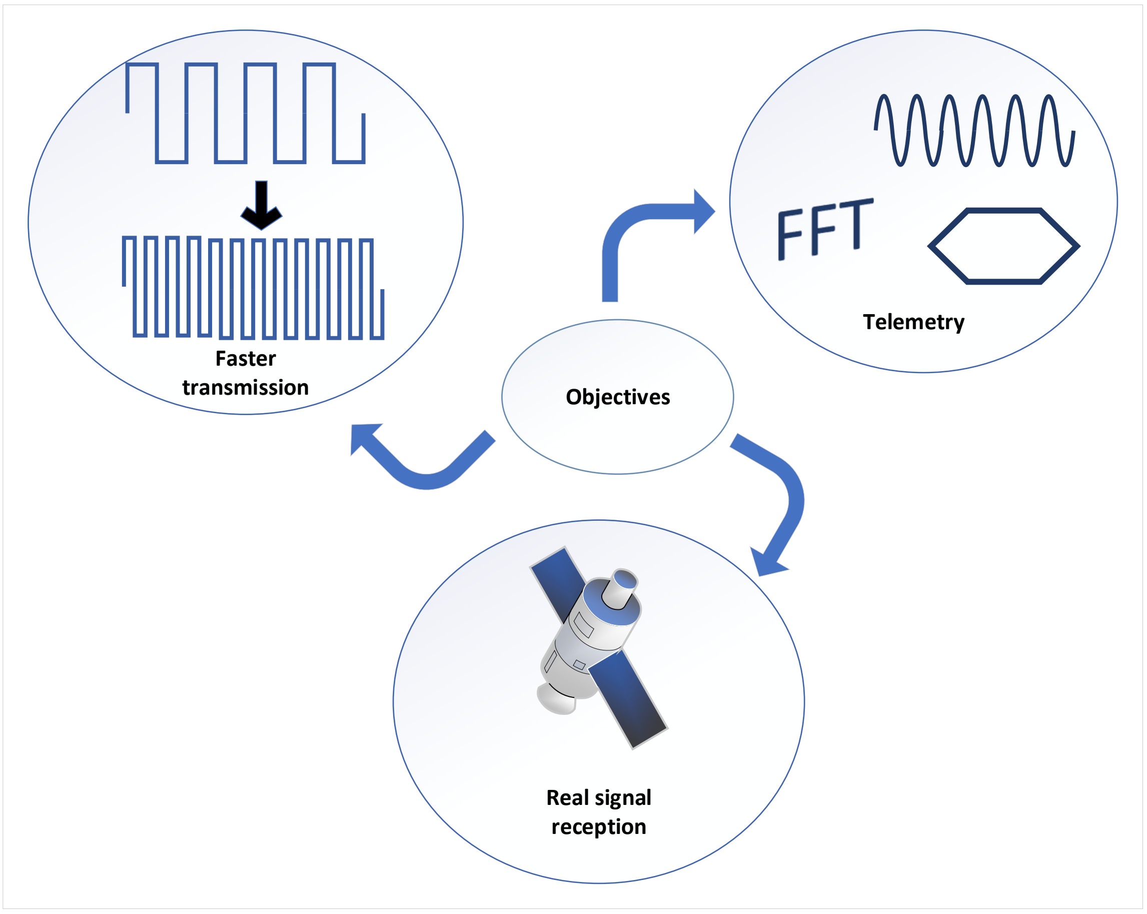


Figure 1.3 – *Objectives of the project.*

CHAPTER

2 SYSTEM REQUIREMENTS

After introduction, requirements to be met by the decoder at the end of the project are explained. It is also explained the software and hardware requirements needed for the correct operation of the developed software. Finally, tests performed and detailed in 5 are listed.

2.1 Requirements

Main requirements are:

- Creation of a telemetry board.
- Changes in the demodulator which allow it to decode signals from satellites.
- Changes in the demodulator which improves the success in signal reception and decoding.
- Creation of a demodulator which allows the decoder to decode signals at 9600 bauds per second.

Secondary requirements are:

- Improvement of the design of the dashboard.
- Improvement of aspects related to the software.

2.2 Possible scenarios

The software is developed for AX25 frames reception and transmission, more concretely, to be part of a ground station which communicates with satellites. Because of that there are two possible scenarios: real one and simulated one. For real scenario it is needed to build a whole ground station to decode signals for satellite. As it is known, this is not affordable for everyone, because of that a simulated scenario will be described.

2.2.1 Real scenario

Device requirements vary depending on if the working mode is reception mode or reception/transmission mode. Here it is going to be explained the reception mode, since for transmission/reception mode is necessary to send a satellite and predefine telecommands, and this part is out of the scope of study of this project.

The diagram of this scenario is shown in figure 2.1

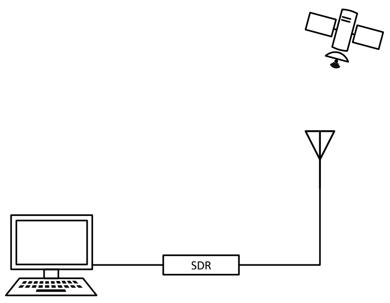


Figure 2.1 – *Real scenario diagram.*

Devices needed are:

- A computer which meets the software requirements detailed below.
- An antenna which allows the connection with the satellite.
- An SDR for RF demodulation.
- A cable able to connect the antenna and the SDR

A budget has been prepared with the prize of all the devices (except computer): Software requirements are:

- An operating system for installing different programs.
- Java Development Kit.

Object	**Prize**
ANTENA DISCONO DIAMOND D3000N	€ 138.02
RF coaxial cable N to MCX connector N male to MCX male St RG178 pigtail cable 15 cm	€ 4.86
Andoer® Mini USB Digital Portátil 2.0 Stick of TV DVB-T + DAB + FM + RTL2832U R820T2	€ 13.99
Total	€ 156.87

Table 2.1 – *Budget for real scenario.*

- Eclipse development environment.
- NetBeans development environment.
- Zadig
- SDRSharp

2.2.2 Simulated scenario

It is also possible use a simulated scenario if there are AX25 frames available, they can be recorded or generated. Let's assume that signal are recorded, devices needed are:

- A computer which meets the software requirements detailed below.
- A external sound card.
- Cable ended in two jack connectors.

Connection is shown in figure 2.2

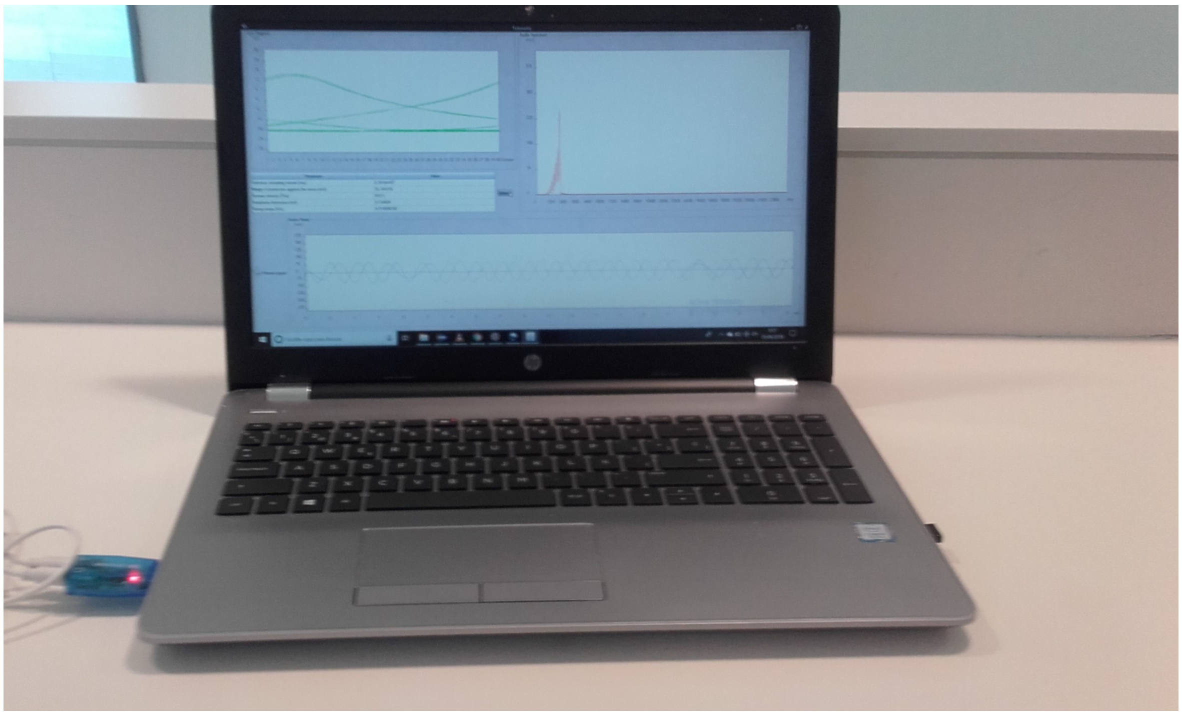


Figure 2.2 – *Simulated scenario diagram.*

Software requirements are:

- An operating system for installing different programs.
- Java Development Kit.
- Eclipse development environment.
- NetBeans development environment.
- VLC for playing recorded signals.

If image is sent, it can be generated and sent by the developed software.

Budget for this scenario is:

Object	**Prize**
SODIAL™ USB 5.1 Sound Card Adapter	€ 0.58
Psylins — Cable of 3,5 mm Male-Male	€ 0.58
Total	€ 1.16

Table 2.2 – *Budget for simulate scenario.*

2.3 Tests performed to the system

To have acknowledged of the starting situation of the decoder, a test is performed at the beginning of it. It is detailed in **??**, where AX25 frames are decoded. After the system analysis in chapter 3, tests will be defined:

- **Evaluation of improvements:** Test performed at the beginning of the project is repeated.
- **Reception of real signals from satellites:** To know the operation of the decoder with real signals from satellites.
- **Reception of signals at 9600 bauds:** To know the operation of the decoder when decoding signals at 9600 bauds per second.

CHAPTER

3

SYSTEM ANALYSIS

In this chapter, it is introduced an analysis of the system based on the requirements exposed in chapter 2. This system will be the starting point of the improvements introduced, detailed in chapter 4.

To know what is needed for the design of each part of the system, a reverse engineering process is developed over the transceiver software, introducing a general approach of the system in **??**, describing and evaluating the demodulator in 3.2, analyzing the Doppler effect in 3.3 and introducing the 9600 bauds standard in 3.4. Finally, a planning of the project will be included in 3.5 and tests will be defined in 3.6

Reverse engineering is the process of getting information from the satellite transceiver (in our case) in order to know how it operates.

The figure 3.1 illustrates an architecture abstraction of the reverse engineering:

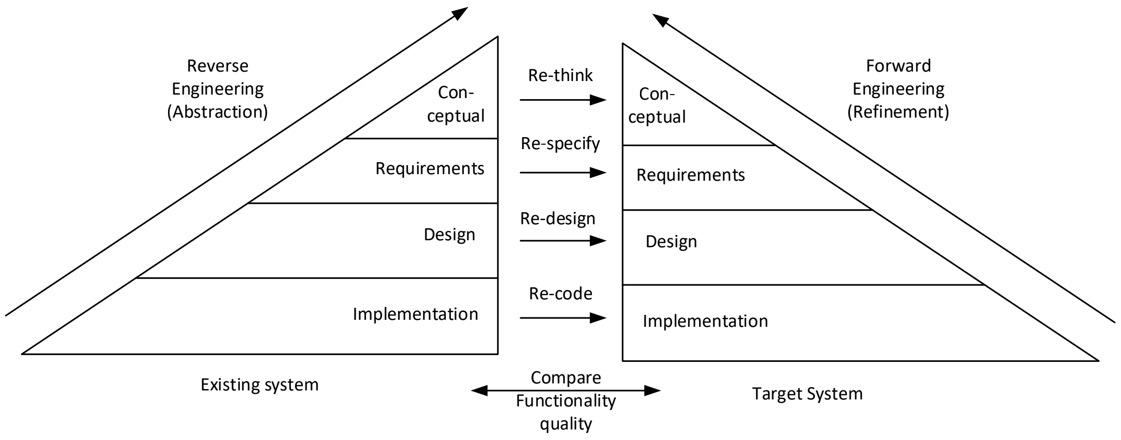


Figure 3.1 – *System design method [3].*

Once the behaviour of the system is completely characterized and evaluated, some telemetry applications 4.3.2 are included in order to better know the behavior of the channel over the physical signal, to make easier the implementation and evaluation of changes performed in the system, related mainly with the physical layer features, which were not considered before the beginning of this project. After that, mentioned changes are introduced in the system (chapter 4) and evaluated in chapter 5

The final system must be able to perform radio transmission, compensating undesirable effects introduced by the radio channel.

3.1 General modem description

In this section, main structure of the system is described. The implementation of the system demodulator is performed with mainly three classes:

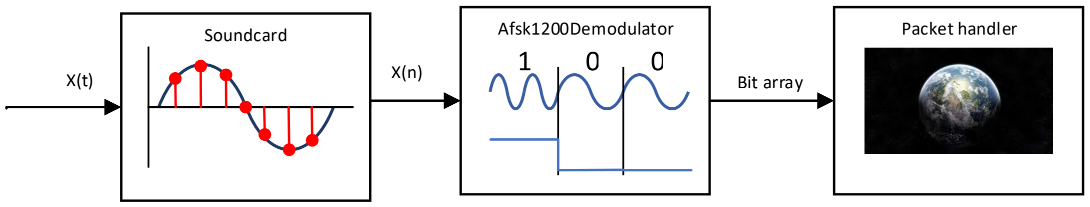


Figure 3.2 – *Receiver implementation diagram [14]*

- **Soundcard**. This class picks the audio samples from the mixer of the physical sound card, it normalizes them and sends to the demodulator.

- **Afsk1200Demodulator**. It is the demodulator itself, it implements the receiving filter, demodulator, and sends data to a handler.

- **Principal**. It is in charge of processing and performing operations, as saving, over receive data.

In the modulator, the operation is performed inversely:

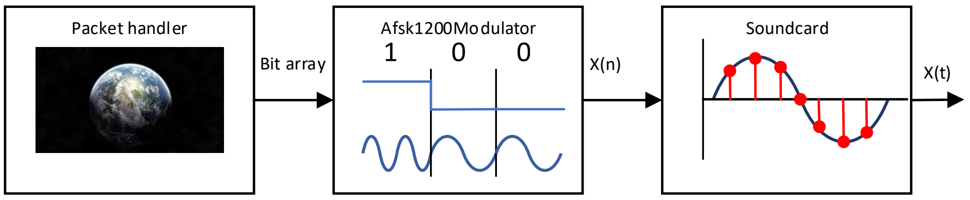


Figure 3.3 – *Transmitter implementation diagram [14]*

- **Principal**. It forms the packets according to the file or message which is being sent.
- **Afsk1200Modulator**. It adds the flags to the message and creates the modulated signal, which is sent to the sound card.
- **Soundcard**. It normalizes the signal and send it into the physical sound card.

As it can be seen in figure 3.4, most of the changes are performed in the demodulator, whereas in the modulator it only will be introduced the 9600 transmission mode:

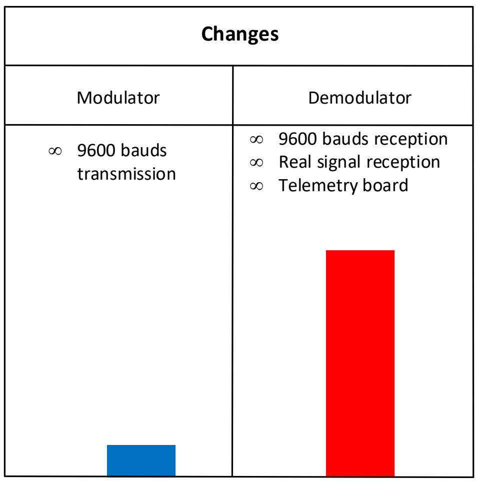


Figure 3.4 – *Changes performed in the modulator and in the demodulator.*

3.2 The demodulator

Since main changes are introduced in the demodulator, it is going to be described more deeply.

3.2.1 Structure

The diagram of the receiver is shown in the figure 3.6:

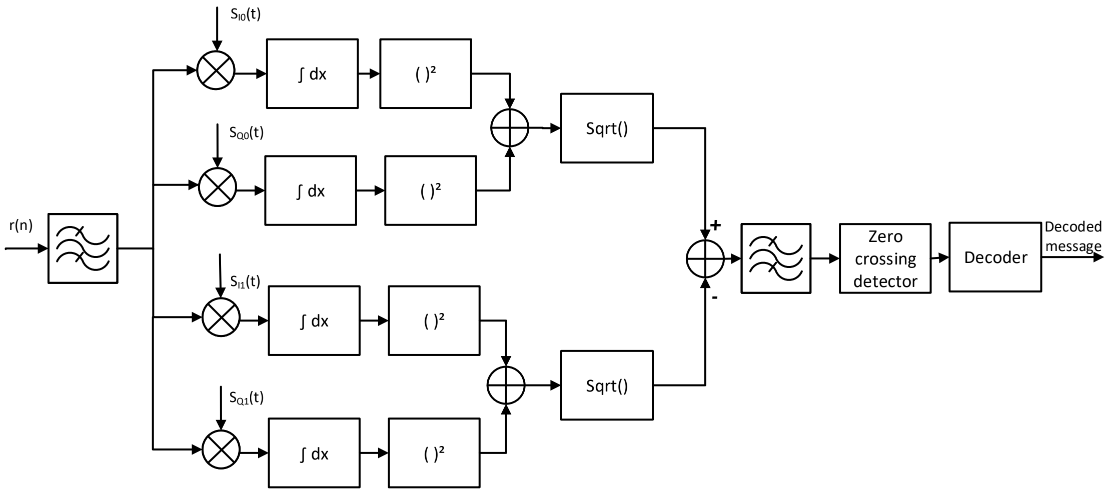


Figure 3.5 – *Block diagram of demodulator in the starting modem.*

It is an optimum no-coherent FSK demodulator. As it is known, sinusoidal' frequency in FSK are orthogonal, so if a symbol is multiplied by the other sinusoidal frequency, the result is 0. This is the basis of this demodulator: The received signal is multiplied by the phase and quadrature components of each symbol frequency. The reason of multiplying by both components is to avoid destructive interference to introduce errors in the detection process. After multiplication, all components are squared, and I/Q components squared are summed, obtaining the energy of I/Q component. The output of the decissor correspond with the symbol of the greatest energy.

To increase the probability of success of the decoder, two demodulators are working in parallel:

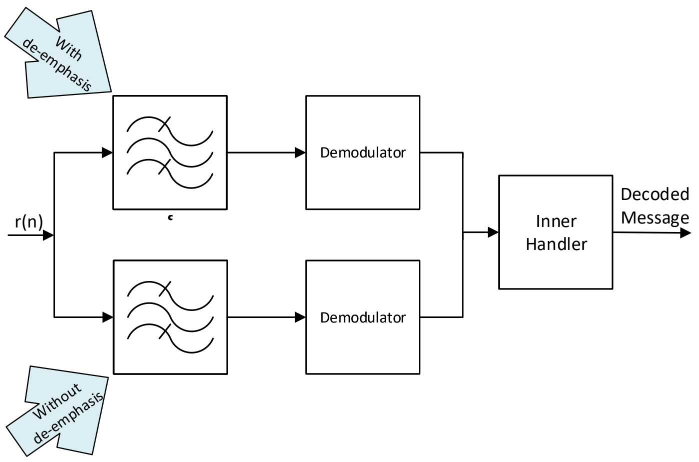
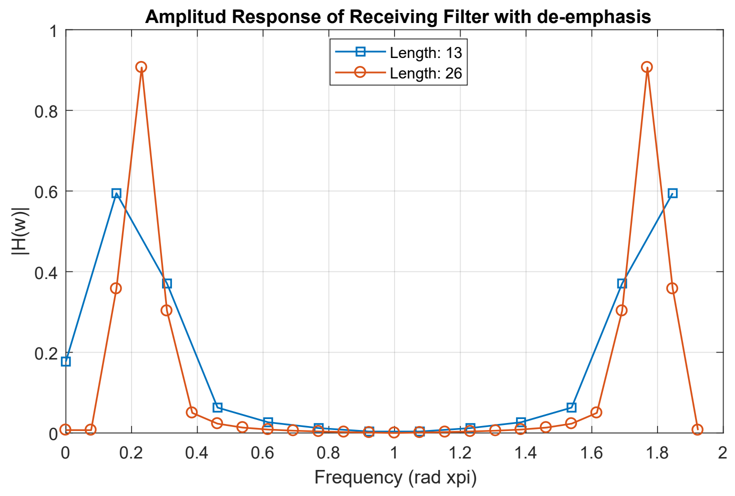
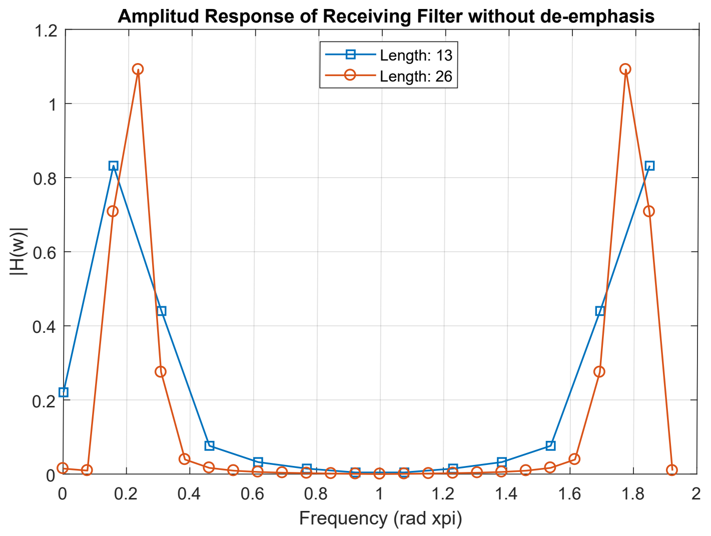


Figure 3.6 – *Block diagram of the multidemodulator system at the starting modem*

Receiver filter is a band-pass filter which can be implemented with or without de-emphais, and both modes with an impulse response of 13 or 26 samples. Filter with de-emphasis are performed with less attenuation for upper-side band. The longest is the impulse response, the narrowest and amplified is the pass-band. The disadvantage of longer impulse response is the requirement of larger memories and the longer delay.



(a) *With de-emphasis*



(b) *Without de-emphasis*

Figure 3.7 – *Receiving filter*

After correlation operation, integrators accumulate the number of samples per symbol, for 16000 audio samples per second (default value for 1200bps mode) the number of samples summed are thirteen.

After subtracting the module of both branches, a low-pass filer which eliminates the higher frequency components of correlation operation is inserted. It can also be implemented with two lengths: eight samples and sixteen samples.

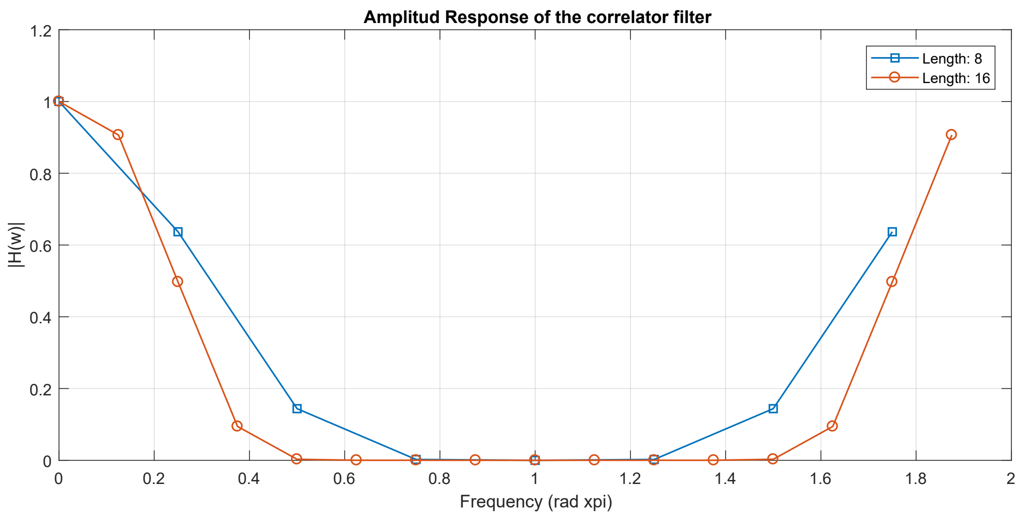


Figure 3.8 – *Correlation filter*

Filters selected are the one with two longer impulse response, because present modem is executed in a computer which has processing power enough to allow long filters use.

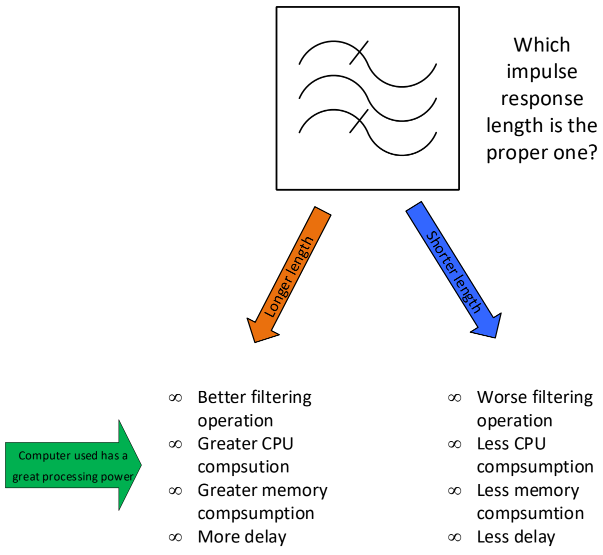

Which impulse response length is the proper one?
Longer length
Shorter length
Computer used has a great processing power
Better filtering operation
Greater CPU compsution
Greater memory compsumption
More delay
Worse filtering operation
Less CPU compsumption
Less memory compsumtion
Less delay

Figure 3.9 – *Filters choosen*

A zero crossing detector is used to detect bits: it counts number of samples between transitions and estimates the number of bits, this is sent to the decoder to form the message.

3.2.2 Demodulator starting test

In order to evaluate the operation of the demodulator before introducing changes, a test will be performed. The decoder will be compared with other two decoders ([2] and [5]), introducing to them frames recorded in an audio file. After that, messages decoded will be saved in a text file and compared. The target of this test is to take acknowledge of the starting operation of the decoder. File with AX25 frames is: ▶

Diagram of the test is:

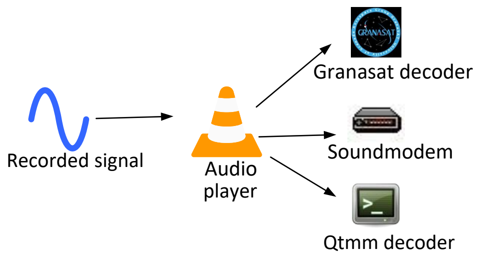


Figure 3.10 – *Diagram of the starting test[2][5][7]*

Results obtained are:

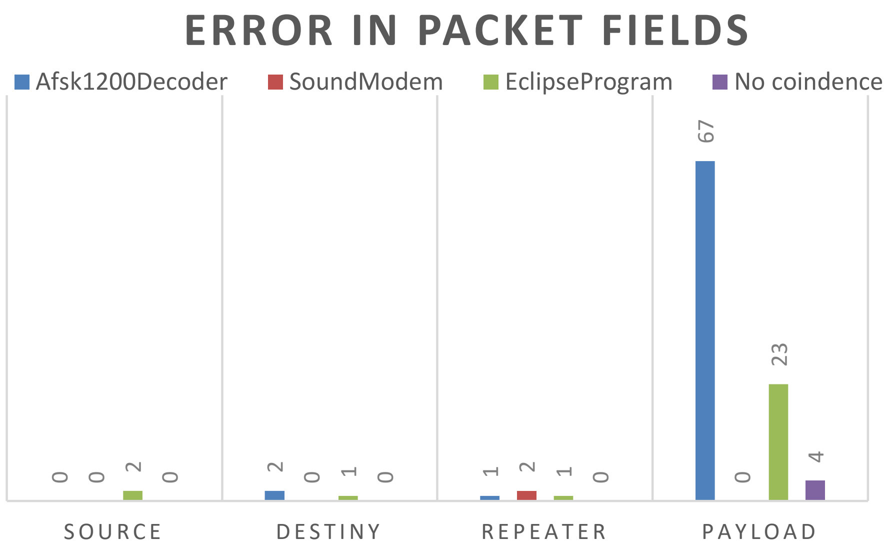


Figure 3.11 – *Corrupted fields in packets*

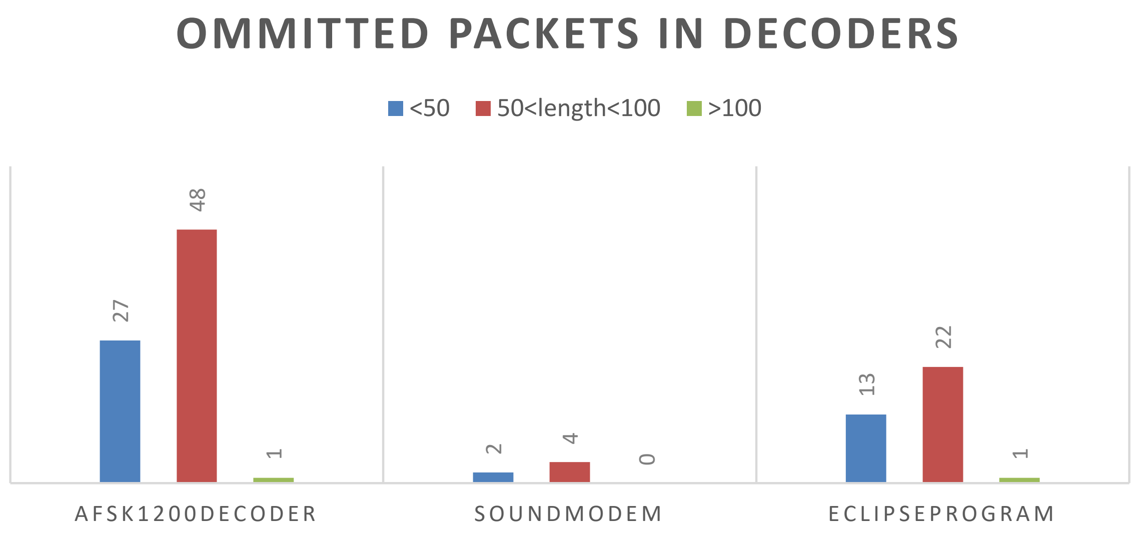


Figure 3.12 – *Omitted packets depending on length*

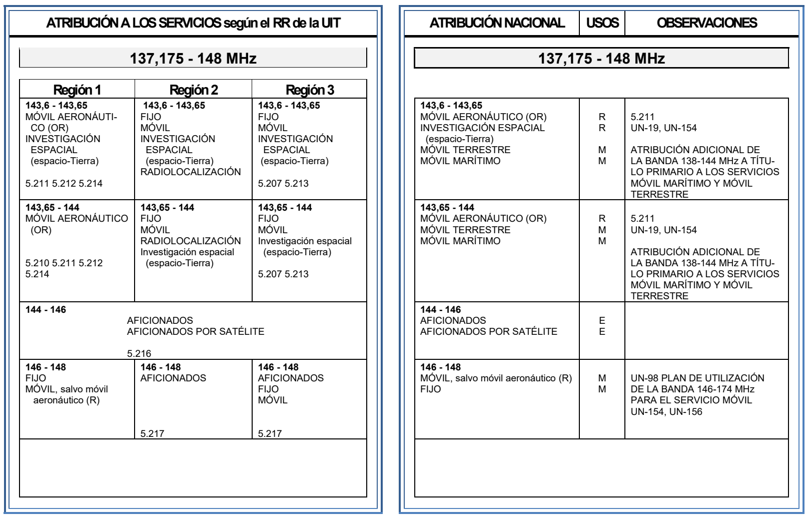
To most corrupted field is the payload. To solve this problem, FEC can be used. Related to the packet length, the minimun number of errors appears with the longest packets.

3.3 Doppler effect

In this section, it is described how Doopler effect affects to the received signal and the way of correcting it.

3.3.1 Frequency Bands of Operation

According to the Spanish National Attribution Frequency Cadre (CNAF)[9], frequency bands assigned to ham transmissions are 144 - 146 MHz in VHF and 430 - 440 MHz in UHF.



ATRIBUCIÓN A LOS SERVICIOS según el RR de la UIT

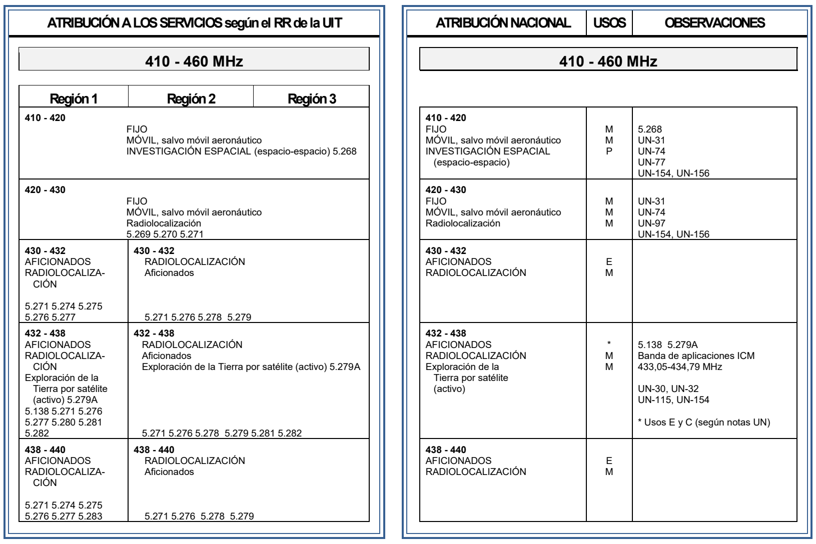
137,175 - 148 MHz

Región 1	Región 2	Región 3
143,6 - 143,65 MÓVIL AERONÁUTI- CO (OR) INVESTIGACIÓN ESPACIAL (espacio-Tierra) 5.211 5.212 5.214	**143,6 - 143,65** FIJO MÓVIL INVESTIGACIÓN ESPACIAL (espacio-Tierra) RADIOLOCALIZACIÓN	**143,6 - 143,65** FIJO MÓVIL INVESTIGACIÓN ESPACIAL (espacio-Tierra) 5.207 5.213
143,65 - 144 MÓVIL AERONÁUTICO (OR) 5.210 5.211 5.212 5.214	**143,65 - 144** FIJO MÓVIL RADIOLOCALIZACIÓN Investigación espacial (espacio-Tierra)	**143,65 - 144** FIJO MÓVIL Investigación espacial (espacio-Tierra) 5.207 5.213
144 - 146 AFICIONADOS AFICIONADOS POR SATÉLITE 5.216		
146 - 148 FIJO MÓVIL, salvo móvil aeronáutico (R)	**146 - 148** AFICIONADOS 5.217	**146 - 148** AFICIONADOS FIJO MÓVIL 5.217

137,175 - 148 MHz

ATRIBUCIÓN NACIONAL	USOS	OBSERVACIONES
143,6 - 143,65 MÓVIL AERONÁUTICO (OR) INVESTIGACIÓN ESPACIAL (espacio-Tierra) MÓVIL TERRESTRE MÓVIL MARÍTIMO	R R M M	5.211 UN-19, UN-154 ATRIBUCIÓN ADICIONAL DE LA BANDA 138-144 MHz A TÍTULO PRIMARIO A LOS SERVICIOS MÓVIL MARÍTIMO Y MÓVIL TERRESTRE
143,65 - 144 MÓVIL AERONÁUTICO (OR) MÓVIL TERRESTRE MÓVIL MARÍTIMO	R M M	5.211 UN-19, UN-154 ATRIBUCIÓN ADICIONAL DE LA BANDA 138-144 MHz A TÍTULO PRIMARIO A LOS SERVICIOS MÓVIL MARÍTIMO Y MÓVIL TERRESTRE
144 - 146 AFICIONADOS AFICIONADOS POR SATÉLITE	E E	
146 - 148 MÓVIL, salvo móvil aeronáutico (R) FIJO	M M	UN-98 PLAN DE UTILIZACIÓN DE LA BANDA 146-174 MHz PARA EL SERVICIO MÓVIL UN-154, UN-156

Figure 3.13 – *VHF band for ham transmission [9]*



ATRIBUCIÓN A LOS SERVICIOS según el RR de la UIT

410 - 460 MHz

Región 1	Región 2	Región 3
410 - 420 FIJO MÓVIL, salvo móvil aeronáutico INVESTIGACIÓN ESPACIAL (espacio-espacio) 5.268		
420 - 430 FIJO MÓVIL, salvo móvil aeronáutico Radiolocalización 5.269 5.270 5.271		
430 - 432 AFICIONADOS RADIOLOCALIZA- CIÓN 5.271 5.274 5.275 5.276 5.277	**430 - 432** RADIOLOCALIZACIÓN Aficionados 5.271 5.276 5.278 5.279	
432 - 438 AFICIONADOS RADIOLOCALIZA- CIÓN Exploración de la Tierra por satélite (activo) 5.279A 5.138 5.271 5.276 5.277 5.280 5.281 5.282	**432 - 438** RADIOLOCALIZACIÓN Aficionados Exploración de la Tierra por satélite (activo) 5.279A 5.271 5.276 5.278 5.279 5.281 5.282	
438 - 440 AFICIONADOS RADIOLOCALIZA- CIÓN 5.271 5.274 5.275 5.276 5.277 5.283	**438 - 440** RADIOLOCALIZACIÓN Aficionados 5.271 5.276 5.278 5.279	

410 - 460 MHz

ATRIBUCIÓN NACIONAL	USOS	OBSERVACIONES
410 - 420 FIJO MÓVIL, salvo móvil aeronáutico INVESTIGACIÓN ESPACIAL (espacio-espacio)	M M P	5.268 UN-31 UN-74 UN-77 UN-154, UN-156
420 - 430 FIJO MÓVIL, salvo móvil aeronáutico Radiolocalización	M M M	UN-31 UN-74 UN-97 UN-154, UN-156
430 - 432 AFICIONADOS RADIOLOCALIZACIÓN	E M	
432 - 438 AFICIONADOS RADIOLOCALIZACIÓN Exploración de la Tierra por satélite (activo)	* M M	5.138 5.279A Banda de aplicaciones ICM 433,05-434,79 MHz UN-30, UN-32 UN-115, UN-154 * Usos E y C (según notas UN)
438 - 440 AFICIONADOS RADIOLOCALIZACIÓN	E M	

Figure 3.14 – *UHF band for ham transmission [9]*

Inside this bands, there are subbands determined for different types of applications. This subbands and canalizations are specified by the International Amateur Radio Union (IARU)

In the table 3.1 is observed that in the VHF band, the subband of operation of satellites is defined from 145.806 to 146 MHz, with a bandwidth of 12 kHz. In the table 3.2 is observed that in the UHF band, the subband of operation of satellites is defined from 435 to 438 MHz, with a bandwidth of 20 kHz.

Frequency	Maximum Bandwidth	Mode	Usage
144.000 144.025	2700 Hz	All Mode	Satellite (downlink only)
144.025 144.100	500 Hz	Telegraphy (EME)	144.050 Telagraphy calling 144.100 Random MS
144.100 144.150	500 Hz	Telegraphy and MGM	144.110-144-160 EME MGM
144.150 144.400	2700 Hz	Telegraphy, MGM and SSB	144.195-144.205 Random MS SSB (m) 144.300 SSB Centre of activity
144.500 144.794	20 kHz	All Mode	144.500 Image mode centre (SSTV, Fax,...) 144.600 Data Centre of activity (MGM, RTTY,..) 144.750 ATV Talk back
144.400 144.490	500Hz	Telegrapy and MGM	Beacons only
144.491 144.493	500 Hz	EMGM	Experimental MGM
144.794 144.9625	12 kHz	MGM Digital communication	144.800 APRS 144.8125 DV internet voice gateway 144.8250 DV internet voice gateway 144.8375 DV internet voice gateway 144.8500 DV internet voice gateway 144.8625 DV internet voice gateway
144.975 145.194	12 kHz	FM / Digital Voice	Repeater input exclusive
145.194 145.206	12 kHz	FM / Digital Voice	Space communication
145.206 145.5625	12 kHz	FM / Digital Voice	145.2375 FM Internet Voice Gateway 145.2875 FM Internet Voice Gateway 145.3375 FM Internet Vocie Gatway 145.375 digital voice calling 145.500 FM calling
145.575 145.7935	12 kHz	FM / Digital Voice	Repeater Output exclusive
145.794 145.806	12 kHz	FM / DIgital Voice	Space communication
145.806 146.000	12kHz	All Mode	Satellite exclusive

Table 3.1 – *Subbands for the band of 144-146 MHz [6].*

Frequency	Maximum bandwith	Mode	Usage
430.000 432.975	20 KHz	All modes	430.025 - 430.375 FM repeater output (1.6 MHz shift) 430.400 - 43.575 digital communications 430.600 . 430.925 digital communications repeater channels 430.925 - 431.025 multimode channels 431.050 - 431.825 Repeater input channel freqs 7.6 MHz shift 431.625 - 431.975 Repeater input channels (1.6MHz shift)
432.000 432.100	500Hz	Telegraphy and MGM	432.000 - 432.500 EME 432.050 Telegraphy Centre of activity 432.088 PSK31 Centre of activity
432.100 432.400	2700Hz	Telegraphy, MGM and SSB	432.200 SSB centre of activity 432.350 Microwave talkback centre of acitivity 432.370 FSK441 centre of activity
432.400 432.490	500 Hz	Telepgraphy and MGM	Beacons exclusive
432.491 432.493	500 Hz	EMGM	Experimental MGM
432.500 432.975	12 KHz	All Modes	432.500 New APRS frequency REPEATER INPUT REGION 1 STANDARD, 25 kHz spacing,2 MHz shift(Channel freq 432.600 - 432.975MHz)
433.000 433.375	12 KHz	FM Digital Voice repeater	REPEATER INPUT REGION 1 STANDARD, 25 kHz spacing, 1.6 MHz shift
433.400 433.575	12 KHz	FM Digital Voice	433.400 SSTV (FM/AFSK) 433.450 Digital Voice calling 433.500 FM calling
433.600 434.000	20 KHz	All modes	433.625 - 433.775 Digital communications channels 434.000 Centre frequency of digital experiments
434.000 434.594	12 KHz	All modes ATV	434.450 Digital communications channels
434.594 434.981	12 KHz	All Modes	
435.000 438.000	20 KHz	Sattelite Service and ATV	ATV Repeater outputs are not permitted in the 435 MHz Band
438.000 440.000	20 KHz	All Modes	438.025 - 438.175 Digital communication channels 438.200 - 438.525 Digital communication repeater channels 438.550 - 438.625 Multi mode 438.650 - 439.425 Repeater output channels (7.6 MHz shift) 439.800 - 439.975 Digital communication link channels

Table 3.2 – *Subands for the band of 430-440 MHz [6].*

3.3.2 Superheterodine Receiver

Once bands of operation are defined, the receiver is characterized. The receiverd used to demodulate the signal from RF to baseband is the ICOM IC-9100. This demodulator can demodulate signals in several bands, but in this section the study is going to be focused in the band of interest in our study. It must be taken into account that the signal is modulated in NFM For UHF and VHF, the receiver is a superheterodyne receiver of double conversion. First stage (Figure 3.15) consists in several filters, a pr amplifier and alternator which adapts the signal for the first IF conversion:

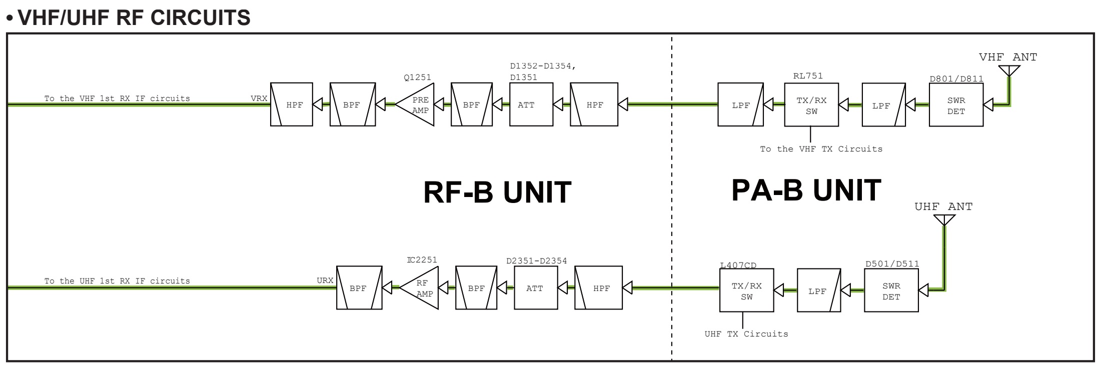


Figure 3.15 – *RF circuits of the receiver [12].*

In the next stage (Figure 3.16), conversions till second IF are made:

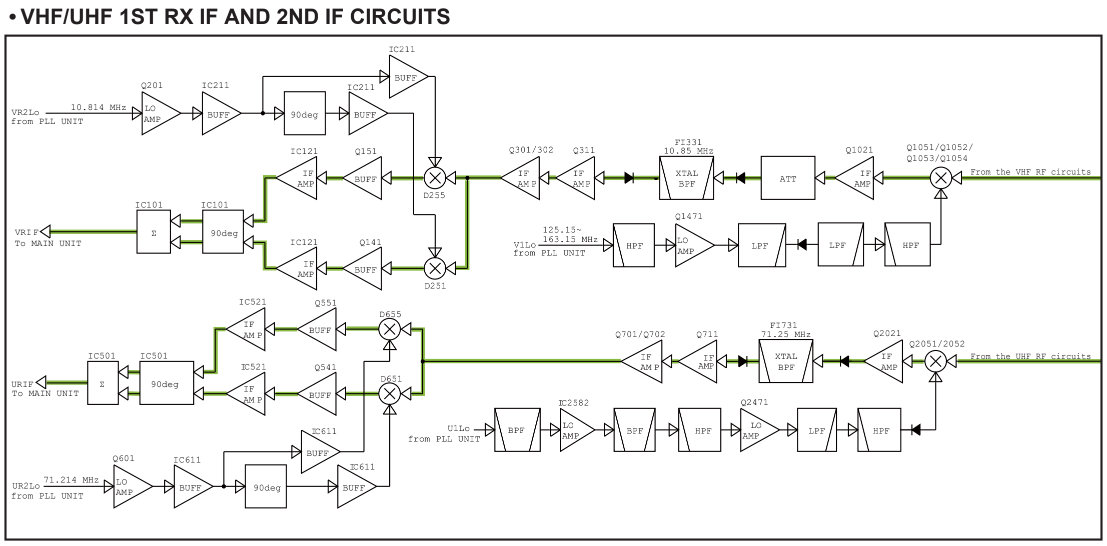


Figure 3.16 – *First and second IF circuits of the receiver [12].*

Finally, digital signal processing is performed (Figure 3.17) to retake the signal:

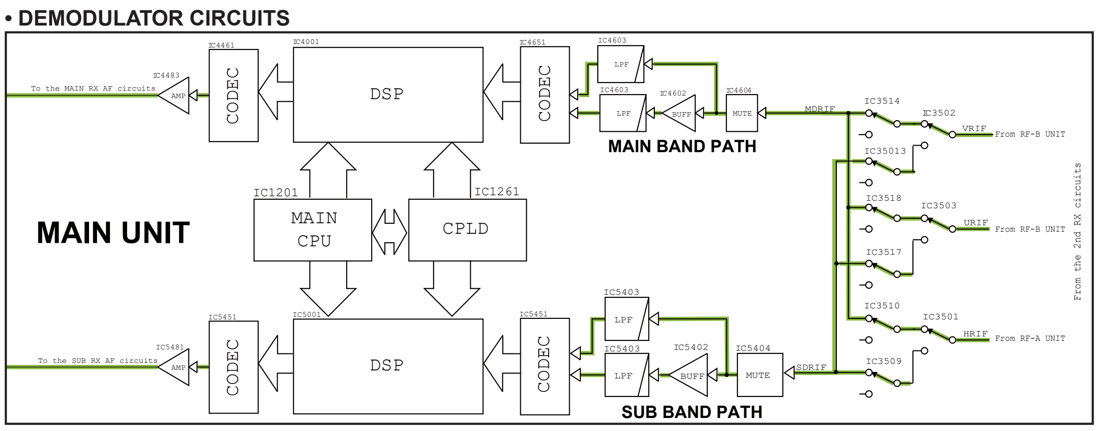


Figure 3.17 – *Demodulator circuits of the receiver [12].*

Intermediate frequencies are summarized in table 3.3

	VHF	UHF
IF 1	10.85 MHz	71.25 MHz
IF 2	36 kHz	

Table 3.3 – *IF frequencies for VHF and UHF.*

3.3.3 Doppler effect in the received signal

In order to know how Doppler effect affects the received signal, a mathematical procedure is developed. With it, it will be possible to understand the effect, and estimate the maximum frequency deviation which supports the system without activating the frequency tuner in the RF receiver. The relative velocity calculation is based in [13].

It must be known the relative velocity of satellite seen from Earth station. For this, the movement of the satellite around the Earth is going to be approximated as a circular uniform movement, to be able to calculate its velocity, as it is represented in figure 3.18.

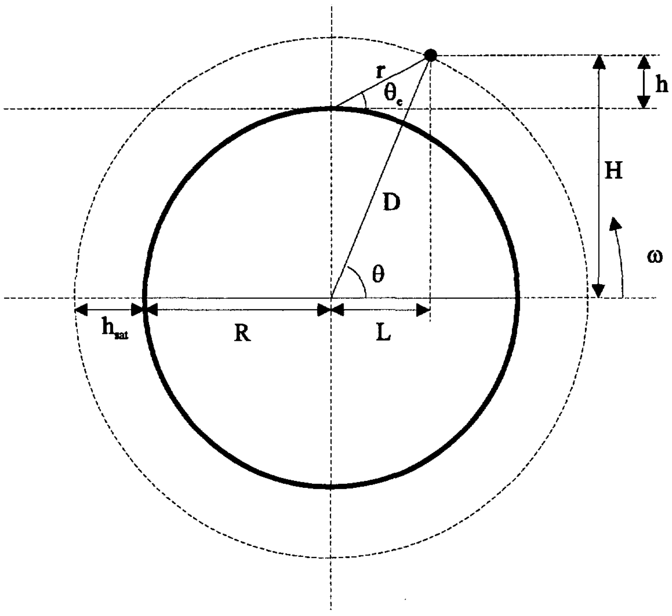


Figure 3.18 – *Orbit of the satellite around the Earth [13].*

Radius of the Earth is R = 6378.14 km. So, relative height of satellite from the horizontal plane which contains the center oh the Earth is expressed as:

$$H = R + h \tag{3.3.1}$$

Therefore, height of the satellite can be written in terms of the angular velocity:

$$H = Dsen(\theta) = Dsen(\omega t) \tag{3.3.2}$$

$$h = Dsen(\theta) - R \tag{3.3.3}$$

where ω is the angular velocity of the satellite. Moreover, the distance of the satellite from the vertical plane which contains the center of the Earth and the ground station can be also expressed in terms of ω:

$$L = Dcos(\theta) = Dcos(\omega t) \tag{3.3.4}$$

Finally, distance of the satellite from the ground station is calculated:

$$r^2 = h^2 + L^2 = D^2 sen^2(\theta) + R^2 - 2DRsen(\omega t) + D^2 cos^2(\omega t) = D^2 + R^2 - 2DRsen(\omega t) \tag{3.3.5}$$

$$r = \sqrt{D^2 + R^2 - 2DRsen(\omega t)} \tag{3.3.6}$$

But what is interesting for this scope of study, it is the relative velocity between the satellite and the ground station, since frequency deviation is:

$$f_D = \frac{v_s}{c} f_t \tag{3.3.7}$$

and received frequency:

$$f_r = \frac{c}{\lambda_r} = \frac{c}{c - v_s} f_t = \frac{1}{1 - \frac{v_s}{c}} f_t \tag{3.3.8}$$

where v_s is the relative velocity between the satellite and the ground station, c is the light velocity and f_t is the transmitted frequency. Applying the relation between distance variation and velocity:

$$v_s = \frac{dr(t)}{dt} = \frac{DR\omega cos(\omega t)}{\sqrt{D^2 + R^2 - 2DRsen(\omega t)}} \tag{3.3.9}$$

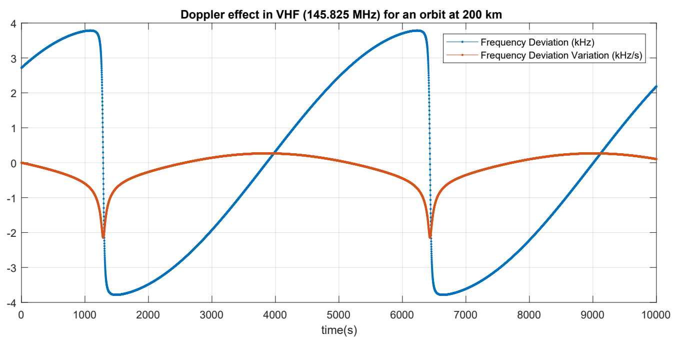
So frequency deviation is:

$$f_D = \frac{v_s}{c} f_t = \frac{DR\omega cos(\omega t)}{c\sqrt{D^2 + R^2 - 2DRsen(\omega t)}} f_t \tag{3.3.10}$$

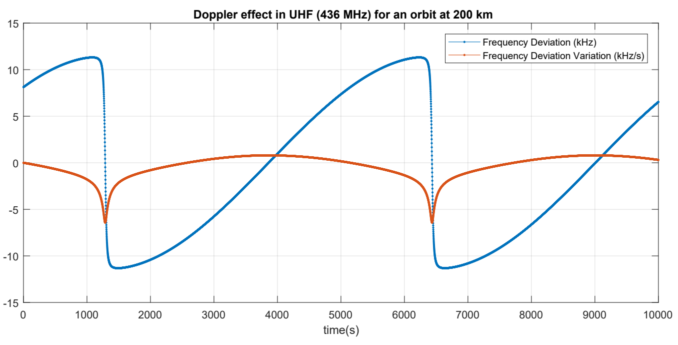
Taking into account that the angular velocity depends on the normal acceleration and the radius:

$$\omega = \frac{a_n}{D} = \frac{g}{D} \tag{3.3.11}$$

where g=9.8 $\frac{m}{s^2}$ is the gravity, it is demonstrated that Doppler effect depends only on satellite orbit and frequency of operation. Simulating this results in Matlab, it is obtained:



(a) *VHF*



(b) *UHF*

Figure 3.19 – *Doppler effect in bands of operation*

In VHF, the maximum frequency deviation is about 3.8 kHz, while in VHF, the maximum frequency deviation is of 12 kHz.

To sum up, a formula which describes how Doppler effect affects to the received signal is calculated, it has also been checked that frequency deviation depends on the orbit radius: the nearer is to the Earth, the greater is the frequency deviation. Furthermore, it increases with the increment of the frequency of operation. In figure 3.19 it is seen that the frequency deviation variation is slow, so in a short time (frame time for example), frequency can be considered constant.

Now let's focus on the received signal. As it was explained in 3.3.2, the signal is modulated with NFM. After two frequency conversions, the signal is processed by a NFM detector. The detector, tuned at f_{IF2}, derivates the phase of the signal to extract the message, which is coded in the instantaneous frequency, but instantaneous frequency is affected by Doppler effect, using 3.3.8:

$$f(t) = \frac{1}{1 - \frac{v_s}{c}}(f_t + m(t)) \tag{3.3.12}$$

where m(t) is the message. Received message is multiplied by a periodic signal (according to 3.3.9). Multiplying in the time domain is equivalent to convolve in the frequency domain. The effect of convolving the spectrum of signal, with the spectrum of a periodic signal whose period tends to zero (as the case of v_s) is a widening of the spectrum. It can be seen with a simple example: an array contains 100 elements, all the elements are 0 but the first and the third one which are one (two impulses very near, as it happens in the spectrum of a periodic signal with a low period, whose spectrum has two impulses centered in the origin). In another array there are ten ones (a pulse), the result of convolving both signals is that the pulse is widened (figure 3.20):

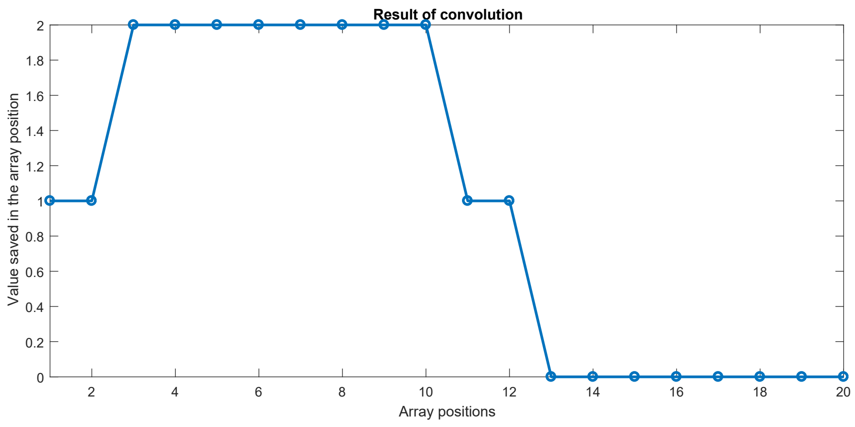


Figure 3.20 – *Example of convolution of a pulse with two near pulses*

For this example it has been considered that the harmonics of the periodic signal are negligible. The effect explained above can be checked with real signals. A real signal has been recorded from ISS station, analyzing its spectrum with Audacity:

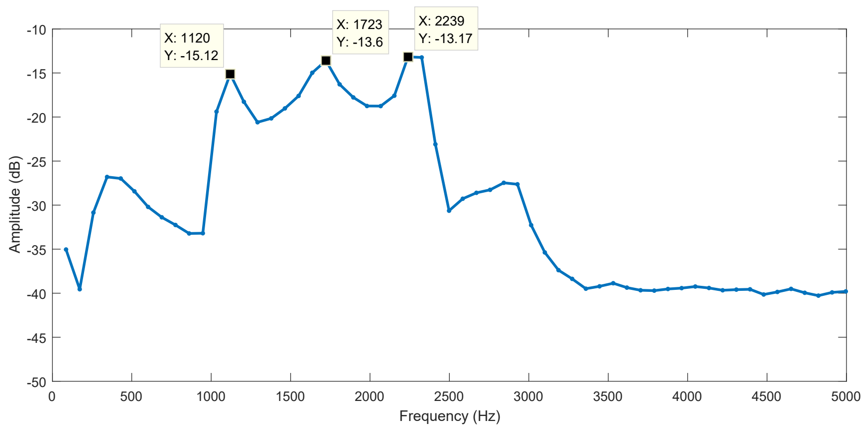


Figure 3.21 – *Spectrum of an ISS signal with Doppler*

It is checked that Doppler effect makes mark and space tones to separate: mark tone is now at 1120 Hz and space tone is at 2239 Hz. Some improvements are added to the demodulator in order to avoid a high decrease of the performance of the demodulator by this effect, they are detailed in 4.1.1 Finally, it is going to be calculated the maximum frequency deviation which is able to support the receiver. To calculate bandwidth of the signal, Carlson's theorem is applied:

$$B_{FSk} = 2(\Delta f + R_s) = 2(500 + 1200) = 3.4kHz \tag{3.3.13}$$

where Δf is frequency deviation, and R_s is the symbol rate. For NFM, the formula can be approximated:

$$B_{NFM} = 2B_{BB} = 2x3.4kHz = 6.8kHz \tag{3.3.14}$$

Now it is going to be supposed that the filters of the heterodyne receiver let pass without distortion the bandwidth corresponding to a channel, specified in 3.3.1, so the extreme of the bandwidth must coincide with the boder oh the channel. In VHF, maximum frequency deviation must be 2.6 kHz and 6.6 kHz in UHF.

3.4 The standard at 9600 bauds

s time goes by, the need of sending larger amounts of data has rised, because of that, 1200 bauds modems started to become obsolete, new researches were headed for increasing the transmission speed of the modem. It was noticed that it was not possible to modulate data bits at 9600 bauds as frequency tones remaining in the legal RF bandwidth. It also does not fulfill the bandwidth requirements of sound devices. Owing to everything exposed, it was decided to introduce the square wave corresponding to bits directly to FM RF modulator.

But it was still not the most efficient way of transmitting high-speed data streams. Steve Goode [1] realized that a square wave is rich in harmonics, which make the bandwidth to be wider, so he proposed the signal to be filtered to eliminate them. In addition to reduce the probability of interference, it also makes easier to design electronics devices in the transmitter and receiver.

He also realized that when many equal bits were sent in a row, a significant DC component was introduced in the wave form, and it needed to be passed to RF without distortion, that made design of modulators and demodulators more complicated. He decided to randomize data before sending it, to reduce the amount of '1's and '0's in a row. His solution was based on a 17-bit linear feedback shift register (LFSR) which implements a scrambling polinomy of $1 + x^{12} + x^{17}$

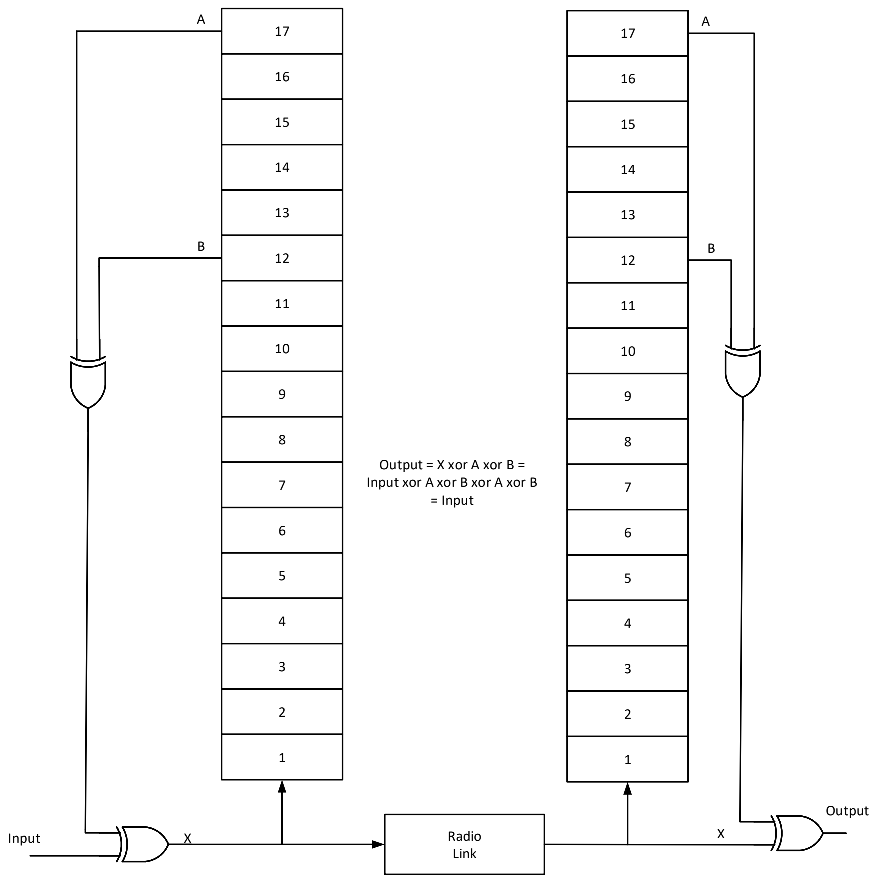


Figure 3.22 – *Scrambler/Descrambler [1]*

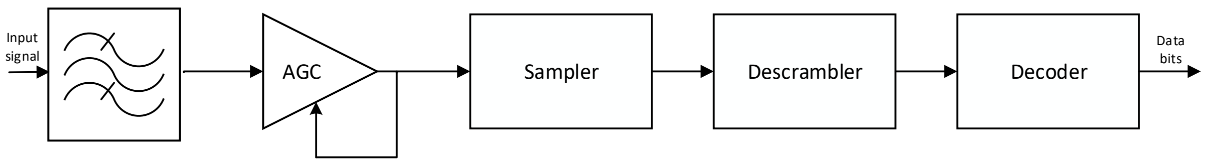


Figure 3.23 – *Demodulator at 9600 bauds.*

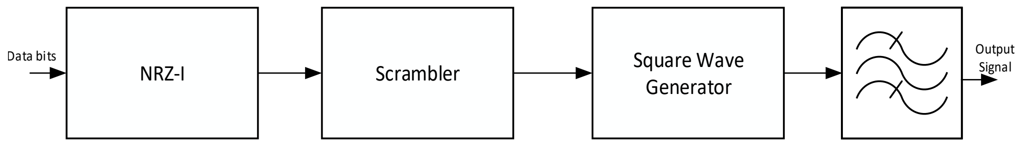


Figure 3.24 – *Modulator at 9600 bauds.*

3.5 Project planning

After the analysis of the system, planning has been done:

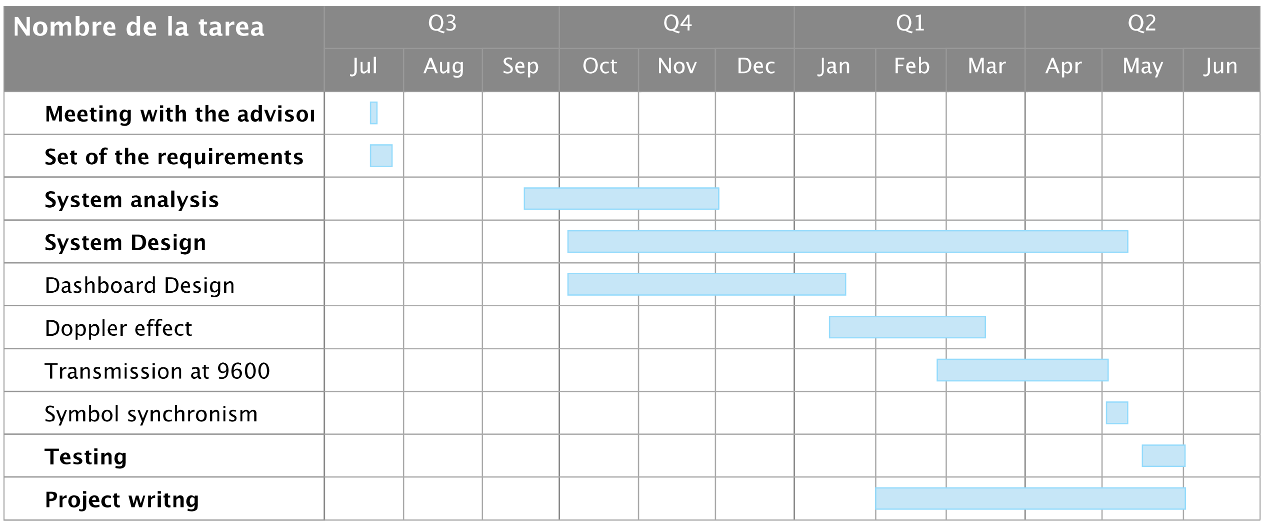


Figure 3.25 – *Planning of the project.*

3.6 Tests definition

Tests which are detailed in chapter 5 are defined here:

- Repetition of the test performed in 3.2.2.
- Reception of signals at 9600 bauds.
- Reception of signals from satellites.
- FEC evaluation, FEC change is explained in 4.1.

CHAPTER

4 SYSTEM DESIGN

In this chapter are explained the changes introduced into the software, trying to meet the requirements detailed in the chapter 2. Once the improvements have been introduced, in chapter 5 tests will be performed in order to have evidence about the requirements met.

4.1 FEC

During system analysis, it was noticed that FEC was implement in a quite inefficient way: To calculate the FEC of a byte, it separates the byte into an array of eight int variables, to calculate FEC, which is unnecessary, because in java, when binary operators are used with tho integers, the operation is performed bit wise, with no necessity of separation. With this change, it is used 5 times less memory, moreover, execution time is shortened.

To illustrate the change, parity matrixes are showed before the change:

```
static int [][] FECCode={
        {1, 1, 1, 0, 1, 1, 0, 0},
        {1, 1, 0, 1, 0, 0, 1, 1},
        {1, 0, 1, 1, 1, 0, 1, 0},
        {0, 1, 1, 1, 0, 1, 0, 1}
    };
int [][] FEC_parity={
        {1, 0, 0, 0},
        {0, 1, 0, 0},
        {0, 0, 1, 0},
        {0, 0, 0, 1}
    };
static int [][] FECWithParity={
        {1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0},
        {1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0},
        {1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0},
        {0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1}
    };
```

And after the change:

```
static int [] FECCode={
        0xEC,//11101100
        0xD3,//11010011
        0xBA,//10111010
        0x75// 01110101
      };
int [] FEC_parity={
      0x8,//1000
      0x4,//0100
      0x2,//0010
      0x1// 0001
    };
static int [] FECWithParity={
      0xEC8,//111011001000
      0xD34,//110100110100
      0xBA2,//101110100010
      0x751// 011101010001
    };
```

It is checked the space saved and the processing time consumption reduction, moreover functions to separate an integer in an array of binary integers ('0s' and '1s'), is eliminated.

4.1.1 Modification in the demodulator

For improving the demodulator performance, and with the intend of providing it with mechanisms to be able to receive real signals, some changes have been introduced. To introduce the changes, three other demodultors has been analyzed: FoxTelem [4], Direwolf [8] and qtmm [2]. The process of reverse engineering has been applied to this decoders, extracting their more interesting features and integrating them in the present system:

4.1.2 New structure

Several demodulators working in parallel can improve the probability of succeed when decoding a message, because of that, structure of decoding has been modified as described in figure 4.1

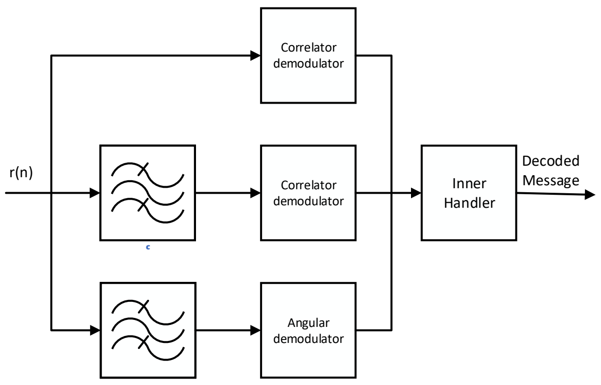


Figure 4.1 – *New demodulator structure*

Both types of demodulators used are described in 4.1.2.1 and 4.1.2.2

4.1.2.1 Angular demodulator

As it is known, a signal can be expressed in terms of its phase and quadrature components:

$$x(t) = x_I(t)cos(w_c t) - x_Q(t)sin(w_c t) \tag{4.1.1}$$

Starting from phase and quadrature components, module and phase of $x(t)$ can be obtained:

$$|x(t)| = \sqrt{x_I^2(t) + x_Q^2(t)} \tag{4.1.2}$$

$$\angle x(t) = arctan\frac{x_Q(t)}{x_I(t)} \tag{4.1.3}$$

AFSK modulation can be seen as a FM modulation carrying a square wave as a message in its instantaneous frequency, for that reason, deriving the angle of $x(t)$, the message signal can be extracted:

$$\frac{d\angle x(t)}{dt} = \frac{1}{1+(\frac{x_Q(t)}{x_I(t)})^2}\frac{d\frac{x_Q(t)}{x_I(t)}}{dt} \tag{4.1.4}$$

$$\frac{d\frac{x_Q(t)}{x_I(t)}}{dt} = \frac{\frac{dx_Q(t)}{dt}x_I(t) - \frac{dx_I(t)}{dt}x_Q(t)}{x_I^2(t)} \tag{4.1.5}$$

Combining 4.1.4 and 4.1.5, it is obtained:

$$\frac{d\angle x(t)}{dt} = \frac{\frac{dx_Q(t)}{dt}x_I(t) - \frac{dx_I(t)}{dt}x_Q(t)}{x_Q^2(t) + x_I^2(t)} \tag{4.1.6}$$

Using the central difference method, it can be transformed in a discrete version:

$$\left.\frac{d\angle x(t)}{dt}\right|_{t=nT} = \frac{x_I(n-1)(x_Q(n) - x_Q(n-2)) - x_Q(n-1)(x_I(n) - x_I(n-2))}{2(x_I^2(n-1) + x_Q^2(n-1))} \tag{4.1.7}$$

The block diagram of the angular demodulator is represented in 4.2

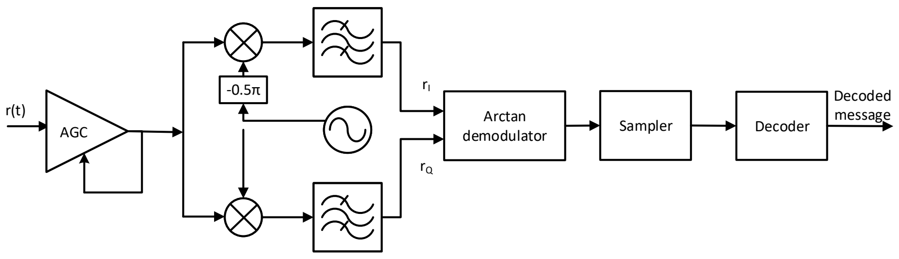


Figure 4.2 – *Angular demodulator*

To get the phase and the quadrature components, ortogonality property of sine and cosine wave are used:

$$cos(\omega t)cos(\omega t) = 1 - sen^2(\omega t) \tag{4.1.8}$$

$$sin(\omega t)sin(\omega t) = 1 - cos^2(\omega t) \tag{4.1.9}$$

$$sin(\omega t)cos(\omega t) = 0 \tag{4.1.10}$$

High frequency components are eliminated with a low-pass filter. When square signal is extracted, it is corrected with an AGC module, explained in 4.1.2.3. Both filters described in 3.7 are used as receiving filters.

After filtering, AGC (described in 4.1.2.3) is applied, because experience has demonstrated that for this kind of demodulator, AGC is useful at the beginning.

```
float angular(float filt) {
//Automatic Frequency Gain
  filt=agc0(filt,0.00022f,0.495f);
//Extraction of phase and quadrature component
  u1I[j_cd]=filt*(float)Math.cos(phase_1700);
  u1Q[j_cd]=filt*(float)Math.sin(phase_1700);

//Eliminating double frequency component
  float xIf=Filter.filter(u1I, j_cd, cd_filter);
  float xQf=Filter.filter(u1Q, j_cd, cd_filter);

//Update carrier phase
  phase_1700+=phase_inc_1700;
  if (phase_1700 > (float) 2.0*Math.PI) phase_1700 -= (float) 2.0*Math.PI;

//Update input of decissor
  xI[2]=xI[1];
  xQ[2]=xQ[1];
  xI[1]=xI[0];
  xQ[1]=xQ[0];
  xI[0]=xIf;
  xQ[0]=xQf;

//Update filter variable
  j_cd++;
  if (j_cd==cd_filter.length) j_cd=0;

  return(xI[1]*(xQ[0]-xQ[2])-xQ[1]*(xI[0]-xI[2]))/(2*(xI[1]*xI[1]+xQ[1]*xQ[1]));
}
```

The input *filt*, is a filtered sample of the received signal, *cd_filter* is used from starting demodulator (3.8), xI and xQ save three samples of phase and quadrature components, needed for calculating the output. This method returns the input of the sampler previous of decissor.

4.1.2.2 New correlator demodulator

The block diagram of the new correlator demodulator (figure 4.1) shows the changes introduced:

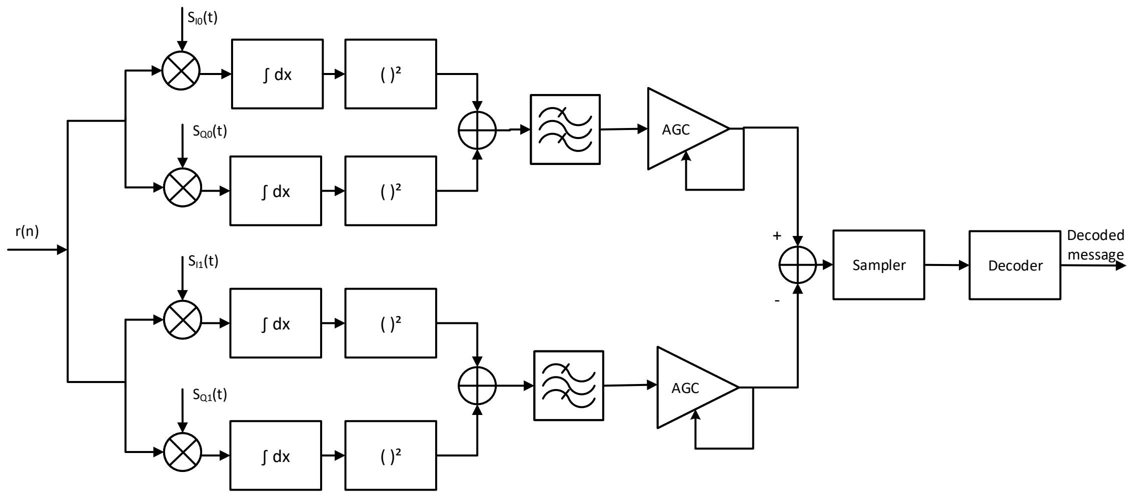


Figure 4.3 – *New correlator demodulator*

One of the most important changes introduced to the initial demodulator is that receiving filter has been removed in one of them, and the other is a filter with de-emphasis. During the design of the new demodulator, it was checked that receiving filters make the modulator not to demodulate signals affected by Doppler effect, and in other real signals, it was observed that 2200 Hz sinudoid was attenuated. Moreover, square root blocks have been removed.

After correlation, low pass filtering and AGC is applied to both branches, which improves performace.

```
float correlator(float filt) {
//Multiply sample by phase and quadrature component of mark and space tones
  c0_real[j_corr] = filt*(float) Math.cos(phase_f0);
  c0_imag[j_corr] = filt*(float) Math.sin(phase_f0);

  c1_real[j_corr] = filt*(float) Math.cos(phase_f1);
  c1_imag[j_corr] = filt*(float) Math.sin(phase_f1);

//Update phase of mark and space tone
  phase_f0 += phase_inc_f0;
  if (phase_f0 > (float) 2.0*Math.PI) phase_f0 -= (float) 2.0*Math.PI;
  phase_f1 += phase_inc_f1;
  if (phase_f1 > (float) 2.0*Math.PI) phase_f1 -= (float) 2.0*Math.PI;

//Accumulation of energy of the branch f0
  float cr = sum(c0_real,j_corr);
  float ci = sum(c0_imag,j_corr);
  c0[j_cd] = (float)(cr*cr + ci*ci);
//Accumulation of energy of the branch f1
  cr = sum(c1_real,j_corr);
  ci = sum(c1_imag,j_corr);
  c1[j_cd] = (float)(cr*cr + ci*ci);

//Filtering energy
  float c0f= Filter.filter(c0,j_cd,cd_filter);
  float c1f= Filter.filter(c1,j_cd,cd_filter);

//Applying AGC
  c0f=agc0(c0f,0.00022f,0.495f);
  c1f=agc1(c1f,0.00022f,0.495f);

//Update filter variables
  j_cd++;
  if (j_cd==cd_filter.length) j_cd=0;

  j_corr++;
  if(j_corr==c0_real.length)j_corr=0;

  return c0f-c1f;
}
```

Variable *filt* is the received sample. This method returns the input of the sampler which feeds the decissor.

4.1.2.3 AGC

The main goal of AGC is that it is able to amplify the signal between two constant values without changing the shape of the signal, making it to be beteen -0.5 and 0.5.

```
float agc(float sample, float slowDecay, float fastAttack) {
    if (sample >= peak) {
      peak = sample * fastAttack + peak * (1.0f - fastAttack);
    }
    else {
      peak = sample * slowDecay + peak * (1.0f - slowDecay);
    }

    if (sample <= valley) {
      valley = sample * fastAttack + valley * (1.0f - fastAttack);
    }
    else  {
      valley = sample * slowDecay + valley * (1.0f - slowDecay);
    }

    if (peak > valley) {
      return ((sample - 0.5f * (peak + valley)) / (peak -valley));
    }
    return (0f);
  }
```

peak and *valley* must be global variables, since they carry information about the maximum and the minimum value of the signal. *slowDecay* and *fastAttack* indicates the amount of energy delivered by the input sample and the previous value of *peak* and *valley*. *fastAttack* must be greater than *slowDecay*, since signals tend to increase faster than decreasing. Finally, sample is corrected if *peak* is greater than *valley*

4.1.2.4 Symbol synchronism

Due to the fact that Doppler effect widens the spectrum of the signal, summed to the unknown delays which can affect to the signal, symbol synchronisms must be introduced. Before, bits where decoded counting the samples between crossings by zero, this technique is too rudimentary.

```
void samp(int sample) {
//Is there a transition?
  if (previous_fdiff != sample) {
//If it is lower than symbol period ceneter, phase is increased
  if (sphase <  (0x8000 -  (SPHASEINC/2)))
      sphase += SPHASEINC/8;
//If not it is decreased
  else
      sphase -=  SPHASEINC/8;
  }
//update symbol phase
  sphase +=  SPHASEINC;
//If it is the end of the period, signal is sampled
  if (sphase  >= 0x10000) {
      sphase &= 0xffff;
      hdlcDecod(sample);
  }
}
```

When there is a transition, symbol phase must correspond to the center of the symbol period, because center of the bit will be located half of a symbol period later. The algorithm checks after each sample if the symbol phase is near of sampling ($phase < 1 - \frac{1}{samplesperbit}$) when a transition occurs, the symbol phase is corrected, and later updated normally. When a symbol period is accumulated, the signal is sampled.

4.2 9600 bauds

Since many amateur satellites transmit at 9600 bauds, one of the improvements of the system is giving to the software the capability of transmitting both 9600 and 1200 bauds. For this task, the process of reverse engineering has been applied to direwolf **??**in order to understand how AX.25 works at high speed.

4.2.1 Implementation

The blocks described above has been implemented as follows:

4.2.1.1 Receiving/transmitting filter

The receiving filter must fulfill some specifications:

- Passband edge at 4.8 kHz.
- Stopband edge at 10 kHz.
- Ripple of 0.1 in the passband
- Ripple of 0.5 in the stopband

To implement this filter the window method has been used, using as window a kaiser window. Kaiser window has been chosen because β parameter give us one extra freedom degree. Using the matlab function kaiserord, for a sampling frequency of 48 kHz, the result obtained is:

- $w_c = 0.3083$
- $\beta = 1$
- Impulse response length = 8

As it can be seen in figure 4.4, the chosen window is a compromise between attenuation of side lobes and transition band. As it is known, a narrow window higher attenuation of side lobes but slower transition. On the other hand, wider windows provide lower attenuation of side lobes and a shorter transition band

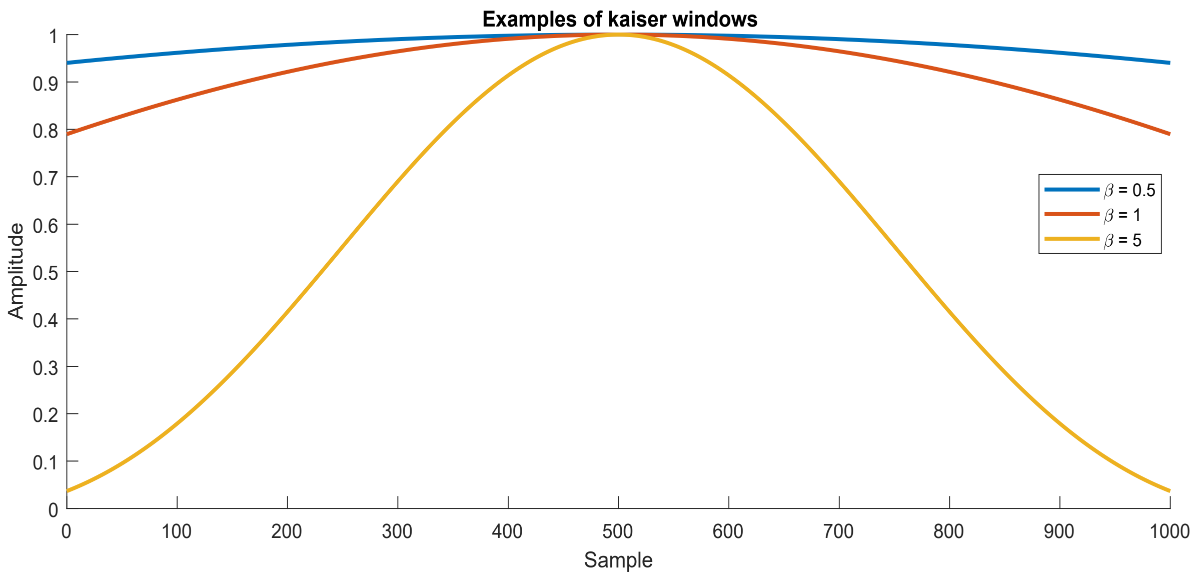
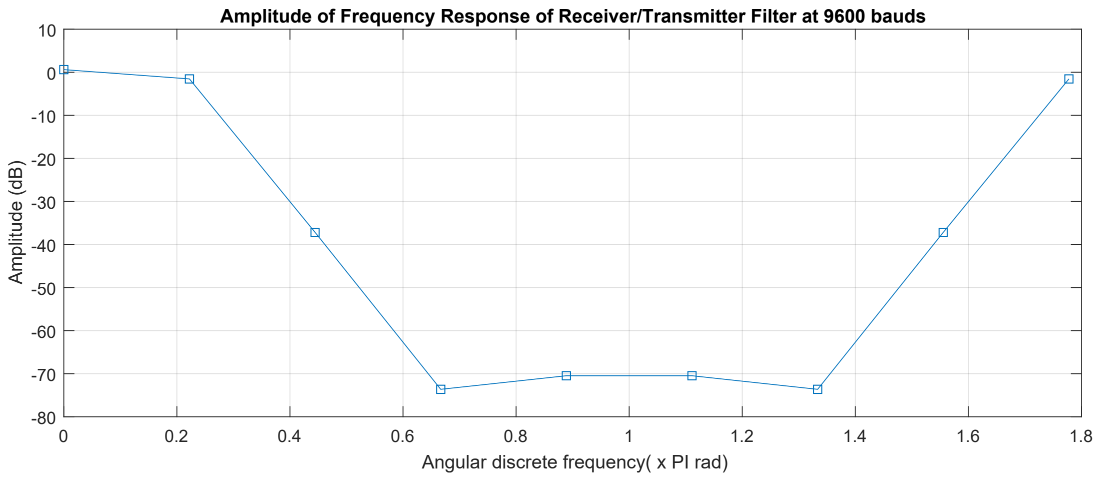
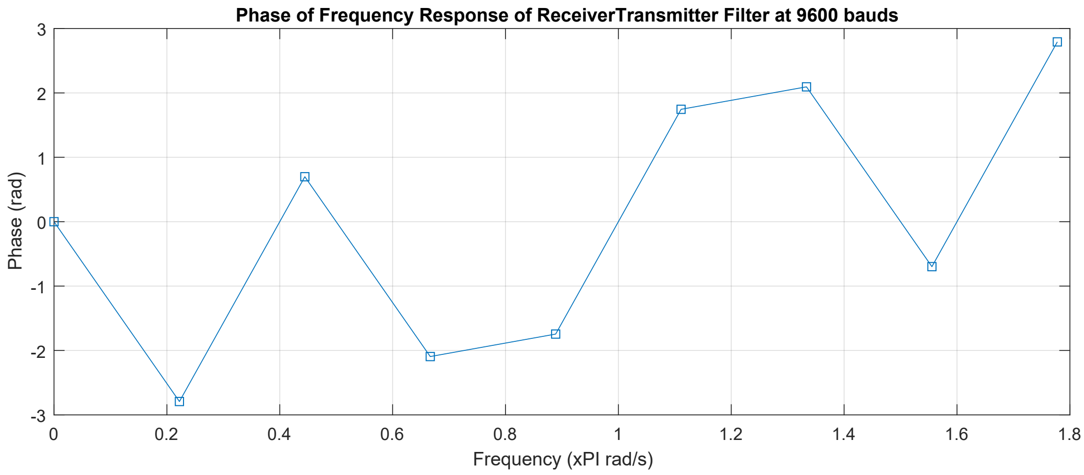


Figure 4.4 – *Example of kaiser windows.*

To implement the filter the fir1 matlab funcion has been used, obtaining the following impulse response:



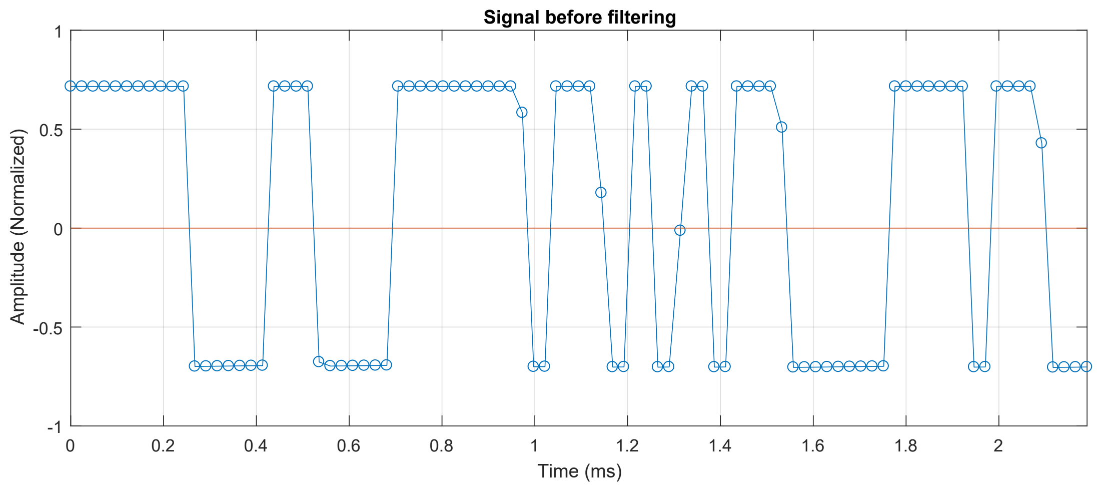
(a) *Amplitude Response*



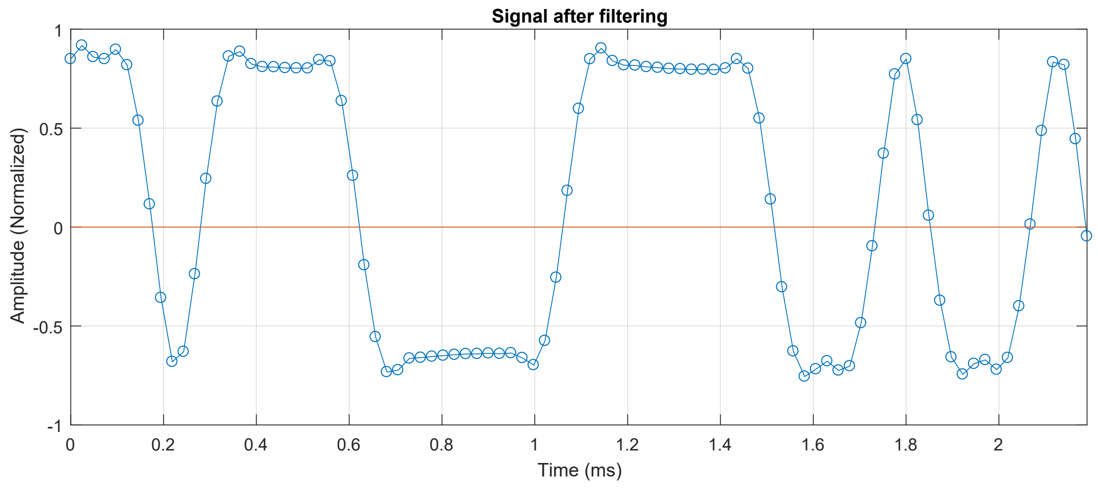
(b) *Phase Response*

Figure 4.5 – *Impulse Response of Receiving Filter at 9600 mode*

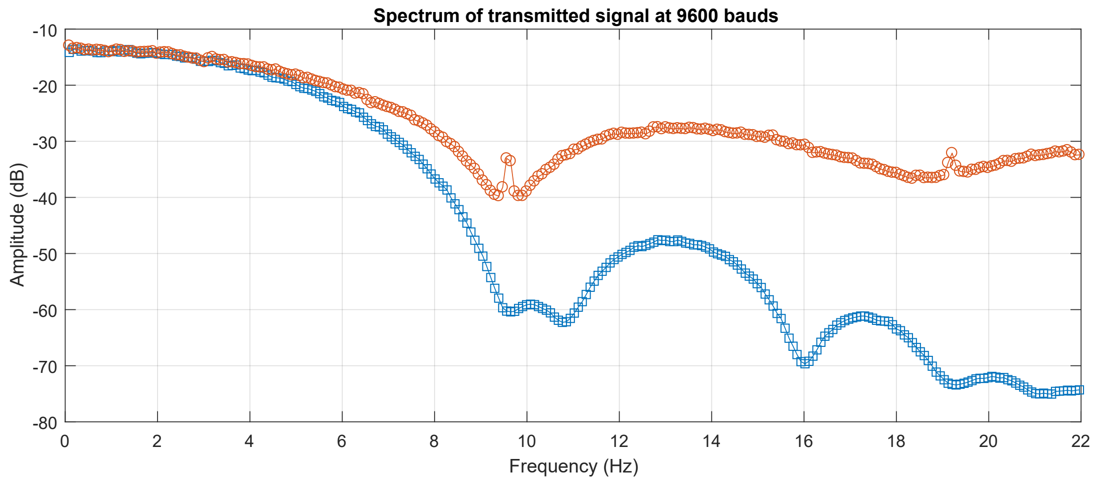
The effect of the filter in the wave form can be seen in the following figures:



(a) *Time Domain without filtering*



(b) *Time Domain filtered*



(c) *Frequency Domain*

Figure 4.6 – *Wave form at 9600 mode*

4.2.1.2 Scrambler

To implement the scrambler, the bit-wise property of java binary operator have been used. The LFSR has been implemented by an int variable, using the following code lines:

```
int b = ((scramble >> 16) ^ (scramble >> 11) ^ (symbol)) & 1;
scramble = (scramble << 1) | (b & 1);
```

and also the descrambler:

```
int b = ( (descramble >> 11) ^ (descramble >> 16) ^ sample ) & 1;
descramble = (descramble << 1) | sample;
}
```

4.2.1.3 class Modulator9600

To develop the class Modulator9600, the class Afsk1200Modulator has been used as starting point, making the main changes in the method generateSymbolSamples(int symbol, float[] s, int position).

This method generates the audio samples corresponding a symbol. It has an argument the symbol to be transmitted, an array 's', to save the audio samples, and the starting postion where samples must be saved in the array.

At 1200 bauds, this method saves samples corresponding to a tone of a determined frequency, corresponding to a '0' or a '1'. It saves the phase of the tone, to make the signal phase continuous, what implies a constant envelope, so suitable to radio transmission.

```
private int generateSymbolSamples(int symbol, float[] s, int position) {
  int count = 0;
//Accumulate samples for a symbol period
  while (tx_symbol_phase < (float) (2.0*Math.PI)) {
//Save sample
    s[position] = (float) Math.sin(tx_dds_phase);
//Update tone phase
    if (symbol==0) tx_dds_phase += phase_inc_f0;
    else           tx_dds_phase += phase_inc_f1;
//Update symbol phase
    tx_symbol_phase += phase_inc_symbol;

    if (tx_dds_phase   > (float) (2.0*Math.PI)) tx_dds_phase   -= (float) (2.0*Math.PI);

    position++;
    count++;
  }
//Reset symbol phase to avoid overflow
  tx_symbol_phase -= (float) (2.0*Math.PI);

  return count;
}
```

At 9600 bauds, instead of tones, a filtered square wave is generated, with the filter showed in 4.2.1.1.

```
private int generateSymbolSamples(int symbol, float[] s, int position) {
  int count=0;
//Scrambler
  int b = ((scramble >> 16) ^ (scramble >> 11) ^ (symbol)) & 1;
  scramble = (scramble << 1) | (b & 1);
//Number of samples to e added
  while (count < samples_per_bit) {

//Sample is added
  if (b==0) f_in[j_f]= -0.5f;
  else      f_in[j_f]=  0.5f;

//Sampled are filtered
  s[position] = Filter.filter(f_in, j_f, f9600);

  j_f++;
```

```
    if(j_f == f_in.length)j_f=0;

    position++;
    count ++;

    }

    return count;
}
```

4.2.1.4 class Demodulator9600

This class is also developed taking as starting point its analog at 1200 bauds. In this case, the main change is in the method addSamplesPrivate(float[] s, int n). This class decodes the message coded in the first 'n' samples of the array 's'

The demodulator implemented in the 1200 mode, was explained in 4.1.1

```
protected void addSamplesPrivate(float[] s, int n) {
int i = 0;
//For each sample
while (i<n) {

   float   sample;
//Interpolate if sample rate is 8000
   if (interpolate) {
     if (interpolate_original) {
       sample = s[i];
       interpolate_last = sample;
       interpolate_original = false;
       i++;
     } else {
       sample = 0.5f * (s[i] + interpolate_last);
       interpolate_original = true;
     }
   } else {
     sample = s[i];
     i++;
   }


   float fdiff;
//If this demodulator is an angular one
   if(emphasis==0 || emphasis==6) {

//Filtering
     u1[j_td]=sample;
     x[j_td]  = Filter.filter(u1, j_td, td_filter);
//Angular demodulation
     fdiff=angular(x[j_td]);

//Save samples for telemetry
     if (s_count < signal.length && emphasis==6) {
       signal[s_count]=sample;
       filteredSignal[s_count]=x[j_td];
       s_count++;
     }
   }

//If this demodulator is a correlator
   else
   fdiff=correlator(sample);


   int b=0;

   if (fdiff > 0)b=1;
//Sampling
  samp(b);

//Save samples for eye diagram
   if (cont < eye.length && emphasis==6) {
     eye[cont]=fdiff;
     cont++;
   }

   previous_fdiff = b;

   j_td++;
   if (j_td==td_filter.length) j_td=0;

  }
}
```

In the case of 9600, no demodulator are needed, just a low-pass filter, an AGC and a sampler, where scrambler is introduced.

```
protected void addSamplesPrivate(float[] s, int n) {
  int i=0;

  while (i<n) {

//Signal is filtered
    u1[j_td]= s[i];

    float diff  = Filter.filter(u1, j_td, f9600);

//AutomaticGainControl
    float fdiff=agc(diff,0.00012f,0.08f);

//Save samples for telemetry
    if (s_count  <signal.length) {
      signal[s_count]=s[i];
      filteredSignal[s_count]=diff;
      s_count++;
    }

    i++;

//Save samples for EyePanel
    if(cont < eye.length) {
    eye[cont]=fdiff;
    cont++;
    }

    int b = 0;
    if (fdiff > 0) b=1;

//Sampling
    samp(b);

    previous_fdiff = b;

//Update filter pointer
    j_td++;  //Index for convoution
    if (j_td==f9600.length) j_td=0;
  }
}
```

Descrambler is introduced in the sampling function:

```
void samp(int sample) {
//Is there a transition?
  if (previous_fdiff != sample) {
//If it is lower than symbol period ceneter, phase is increased
  if (sphase <  (0x8000 -  (SPHASEINC/2)))
      sphase += SPHASEINC/8;
//If not it is decreased
  else
      sphase -=  SPHASEINC/8;
  }
//update symbol phase
  sphase +=  SPHASEINC;
//If it is the end of the period, signal is sampled
  if (sphase  >= 0x10000) {
      sphase &= 0xffff;
//Descramble
    int b = ( (descramble >> 11) ^ (descramble >> 16) ^ sample ) & 1;
    descramble = (descramble << 1) | sample;

      hdlcDecod(b);
  }
}
```

4.3 Dashboard

Dashboard designed by Lucía Castañeda González in her project [11], has been modified, reordering the components and including a new LookAndFeel, making it more attractive4.3.1. In addition, a Telemetry button has been included. When it is clicked on it, telemetry window is displayed. In it is possible to find and eye diagram 4.3.5, a spectrum plot 4.3.4, a representation of the time domain signal 4.3.3 and some parameters extracted from the eye diagram 4.3.6

4.3.1 Main Board

The dashboard found at the beginning of the project was:

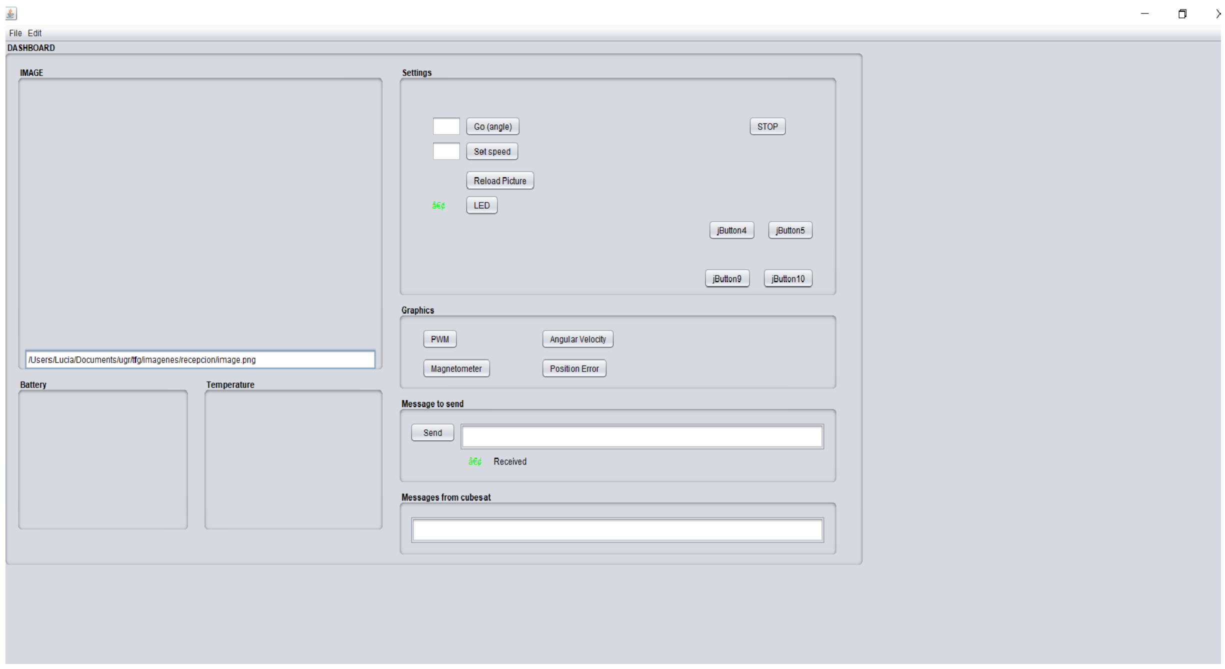
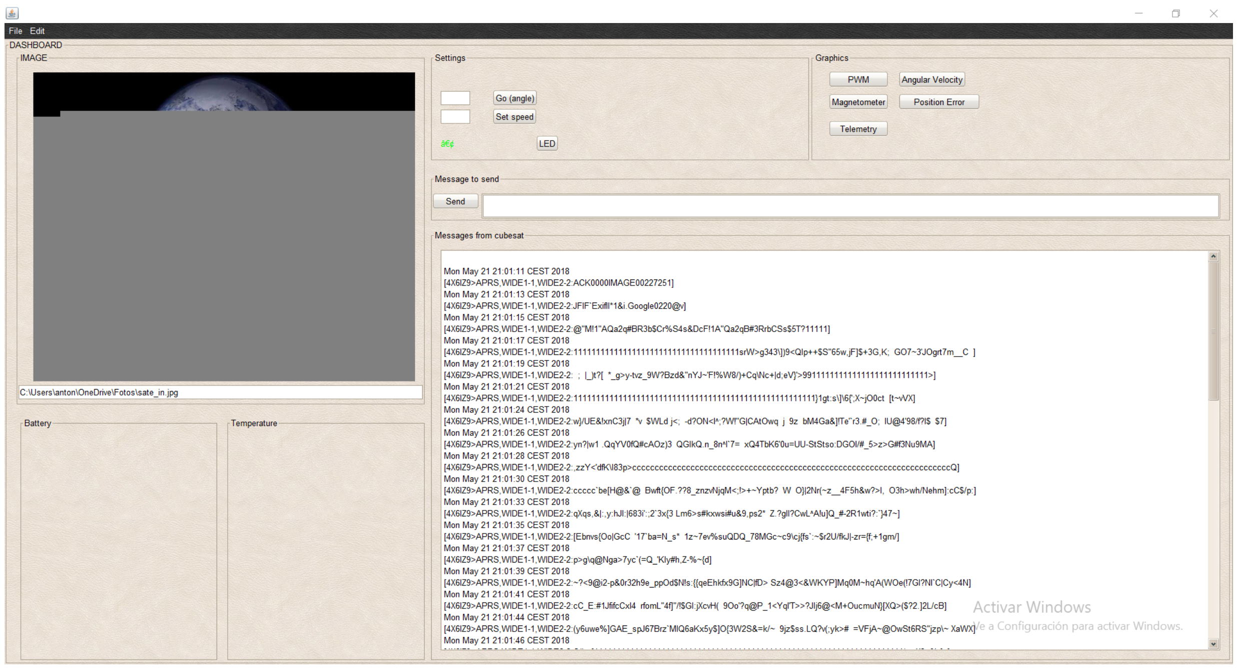


Figure 4.7 – *Old dashboard [11].*

New one is:



```
C:\Users\anton\OneDrive\Fotos\sate_in.jpg
```

```
Mon May 21 21:01:11 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:ACK0000IMAGE00227251]
Mon May 21 21:01:13 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:JFIF`Exifll*1&i.Google0220@v]
Mon May 21 21:01:15 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:@"M!1"AQa2q#BR3b$Cr%S4s&DcF!1A"Qa2qB#3RrbCSs$5T?11111]
Mon May 21 21:01:17 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:1111111111111111111111111111111111srW>g343\])9<Qlp++$S"65w,jF]$+3G,K;  GO7~3'JOgrt7m__C  ]
Mon May 21 21:01:19 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:  ;  |_)t?[  *_g>y-tvz_9W?Bzd&"nYJ~'F!%W8/)+Cq\Nc+|d;eV}'>991111111111111111111111111>]
Mon May 21 21:01:21 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:1111111111111111111111111111111111111111111111111111}1gt:s\]\6{';X~jO0ct  [t~vVX]
Mon May 21 21:01:24 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:w}/UE&!xnC3j|7  *v  $WLd j<;  -d?ON<I^;?W!"G|CAtOwq  j  9z  bM4Ga&]!Te`r3.#_O;  IU@4'98/f?I$  $7]
Mon May 21 21:01:26 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:yn?|w1 .QqYV0fQ#cAOz)3  QGlkQ.n_8n^l`7=  xQ4TbK6'0u=UU-StStso:DGOI/#_5>z>G#f3Nu9MA]
Mon May 21 21:01:28 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:,zzY<'dfK\I83p>ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccQ]
Mon May 21 21:01:30 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:ccccc`be[H@&`@  Bwft{OF.??8_znzvNjqM<;!>+~Yptb?  W  O}|2Nr(~z__4F5h&w?>l,  O3h>wh/Nehm]:cC$/p:]
Mon May 21 21:01:33 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:qXqs,&|:,y:hJl:|683i':;2`3x{3 Lm6>s#kxwsi#u&9,ps2*  Z.?gll?CwL^A!u]Q_#-2R1wti?:`}47~]
Mon May 21 21:01:35 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:[Ebnvs{Oo|GcC  '17`ba=N_s*  1z~7ev%suQDQ_78MGc~c9\cj{fs`:~$r2U/fkJ|-zr={f;+1gm/]
Mon May 21 21:01:37 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:p>g\q@Nga>7yc`(=Q_'Kly#h,Z-%~{d]
Mon May 21 21:01:39 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:~?<9@i2-p&0r32h9e_ppOd$N!s:{{qeEhkfx9G]NC|fD> Sz4@3<&WKYP]Mq0M~hq'A(WOe(!7GI?NI`C|Cy<4N]
Mon May 21 21:01:41 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:cC_E:#1JfifcCxl4  rfomL"4f]"/!$GI:jXcvH(  9Oo'?q@P_1<Yql'T>>?Jlj6@<M+OucmuN}[XQ>($?2.]2L/cB]
Mon May 21 21:01:44 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:(y6uwe%]GAE_spJ67Brz`MIQ6aKx5y$]O{3W2S&=k/~  9jz$ss.LQ?v(;yk>#  =VFjA~@OwSt6RS"jzp\~ XaWX]
Mon May 21 21:01:46 CEST 2018
```

Figure 4.8 – *New dashboard.*

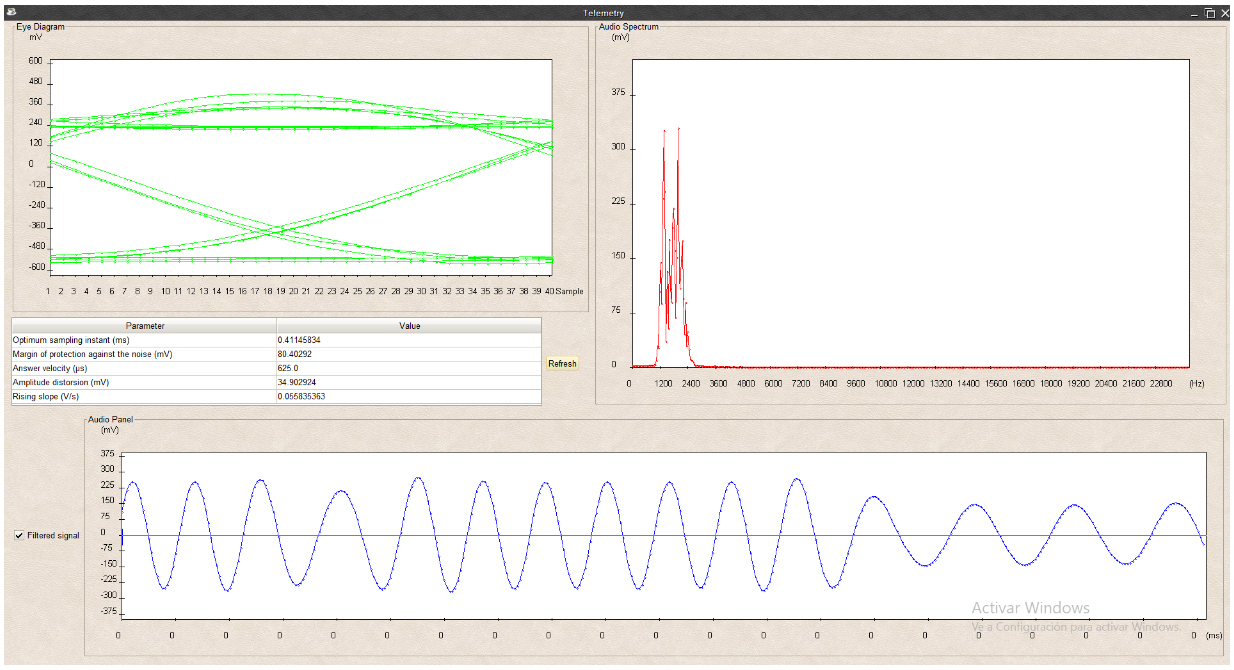
Related to buttons, main changes are the elimination of *Reload directory* and *Stop* buttons, and creation of *telemetry* board button. Image is reloaded automatically now, and location in file system is shown in the text field under it. Now the window is resizable, and it shows a list of received messages, with the time of reception, whereas all version only showed last received message. In new version, all components are resizable.

A video receiving packets at 1200 bauds shows the new dashboard:

It is also included a video receiving at 9600 bauds, to show the increased speed:

4.3.2 Telemetry Board

Telemetry is the remote acquisition of physical magnitudes for being sent after it. In a satellite communication system, this is rather useful, because as mentioned before, radio channel affects the signal a lot, so it is important to know how these changes are.



Parameter	Value
Optimum sampling instant (ms)	0.41145834
Margin of protection against the noise (mV)	80.40292
Answer velocity (µs)	625.0
Amplitude distorsion (mV)	34.902924
Rising slope (V/s)	0.055835363

Figure 4.9 – *Telemetry window.*

As it can be seen in figure 4.9, four elements have been implemented. The have been implemented in three classes: AudioGraphPanel, SpectrumAlanyzer and EyePanel. Following sections are dedicated to explain how they have been implemented in these three classes.

4.3.3 Time Domain Signal

Time domain signal representation has been implemented in the class AudioGraphPanel. It is a Runnable class which extracts samples from the demodulator (it can be seen in the code exposed in 4.2.1.4), and it adapts them for being displayed. Signal can be displayed filtered or not. The method *run*illustrates the routine of this method:

```java
public void run() {
    done = false;
    running = true;

    while(running) {

    try {
        Thread.sleep(1000/60); // approx 1/60 sec refresh
      } catch (InterruptedException e) {
        System.err.println("ERROR: Audiograph thread interrupted");
      }
//Choose between filtered audio or not
            if(soundcard != null && soundcard.getFilteredSignal()!= null){
              if(filteredAudio) {

                    for(int i=0;i < audioData.length;i++){
                            audioData[i]=soundcard.getFilteredSignal()[i];

                    }
              }else {

                    for(int i=0;i < audioData.length;i++){
                            audioData[i]=soundcard.getSignal()[i];
                    }

                    }
            }
//Repaint the signal
            this.repaint();
  }
}
```

To know the voltage value of the received signal, a sinusoid was generated with audacity and measured with the polimeter as described in the picture:

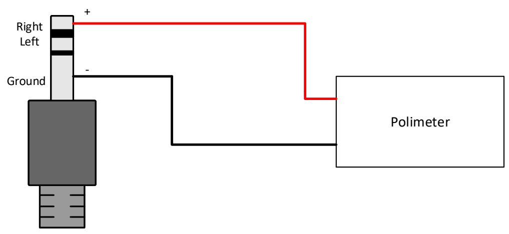


Figure 4.10 – *Tone voltage measure to give format to the vertical axis of the audio panel.*

The method *paintComponent* is used to update the dashboard with new samples:

```
public void paintComponent(Graphics g) {
  super.paintComponent( g ); // call superclass's paintComponent
  Graphics2D g2 = ( Graphics2D ) g; // cast g to Graphics2D

//30 pix of border
  int border = 30;
  int graphHeight = getHeight() - border;
  int graphWidth = getWidth() - border;

//White layer
    int polygonX[]={border*2,border*2,graphWidth,graphWidth};
    int polygonY[]={border*2,graphHeight-border,graphHeight-border,border*2};

    Polygon layer=new Polygon(polygonX,polygonY,4);
    g2.setColor(Color.WHITE);
    g2.fillPolygon(layer);


    g2.setColor(Color.BLACK);

// Draw a black square which surrounds the white layer
  g2.drawLine(border*2, graphHeight-border, graphWidth, graphHeight-border);
  g2.drawLine(border*2, border*2, graphWidth, border*2);
  g2.drawLine(border*2, getHeight()-2*border, border*2, border*2);
    g2.drawLine(graphWidth, graphHeight-border, graphWidth, border*2);

  int lastx = border*2+1;
  int lasty = graphHeight/2+border;
  int x = border*2+1;

//Horizontal axis
  g2.drawString("(ms)",graphWidth,graphHeight);

  if(soundcard!=null){

    for(int i=0;i<=40;i++)
    {
        g2.drawLine(i*((graphWidth-border*2)/20)+2*border,graphHeight-border-2,i*((graphWidth
         -border*2)/20)+2*border,graphHeight-border+2);
        g2.drawString(String.valueOf(i*1000/(soundcard.getRate())),i*((graphWidth-border*2)
         /20)+2*border-8,graphHeight);
    }
    }

//Vertica axis
    for(int i=-5;i<=5;i++)
  {
        g2.drawLine(border*2+4,(int)(graphHeight/2 +border*0.5 + i*0.1*(graphHeight-3.5*
         border)), border*2-4,(int)(graphHeight/2 +border*0.5 + i*0.1*(graphHeight-3.5*border
         )) );
      g2.drawString(String.valueOf(-i*75),
                      border-2,(int)(graphHeight/2 + i*0.1*(graphHeight-3.5*border)  +border
                       *0.5));

  }

```

```
    g2.drawString("(mV)", border-2, border);

//Draw the signal
  g2.setColor(Color.BLUE);


  if (audioData != null && audioData.length > 0) {

    for (int i=0; i < audioData.length ; i++) {
//88 is a scale factor
                        x = border*2 + i*(graphWidth-border*2)/audioData.length;
                        double y = graphHeight/2 + 88*audioData[i] + 0.5*border;

                        g2.drawArc(x, (int)y,2, 2, 0, 360);
                        g2.drawLine(lastx, lasty, x, (int)y);
                        lastx = x;
                        lasty = (int)y;

                    }
  }
//Zero axis
  g2.setColor(Color.GRAY);
  g2.drawLine(border*2,(int)( graphHeight/2+0.5*border), graphWidth,(int)( graphHeight/2+0.5*
    border));
}
```

4.3.4 Frequency Domain Signal

This window is implemented in the class SpectrumAnalyzer, it plots the spectrum of the received signal, filtered or not. To implement the FFT, the class Complex has been implemented:

```
public class Complex{
  private float real;
  private float imag;
//Costructors
  public Complex(float r,float i){
    real=r; //Real part
    imag=i; //Imaginary Part
  }

  public Complex(){
    real=0;
    imag=0;
  }
//Absolute value
  public float abs(){return (float)Math.sqrt(real*real+imag*imag);}
//Convert a real number in a Complex object
  public void Real2Complex(float s){real=s;imag=0;}
//Addition
  public Complex plus(Complex a){
    return new Complex(a.real+this.real,a.imag+this.imag);
  }
//Substraction
  public Complex minus(Complex a){
    return new Complex(this.real-a.real,this.imag -a.imag);
  }
//Multiplication
  public Complex times(Complex a){return new Complex(a.real*this.real-a.imag*this.imag,a.real
    *this.imag+this.real*a.imag);
  }
}
```

The Discrete Fourier Transform has been implemented in the method *fft*, where the input array must have a length of a power of two:

```
public Complex[] fft(Complex[] x) {
    int n = x.length;

    if (n == 1) return new Complex[] { x[0] };


    if (n % 2 != 0) {
        throw new IllegalArgumentException("n is not a power of 2");
    }

//  fft of even terms
    Complex[] even = new Complex[n/2];
    for (int k = 0; k < n/2; k++) {
    even[k]=(x[2*k]);
    }
    Complex[] q = fft(even);

// fft of odd terms
    Complex[] odd  = even;  // reuse the array
    for (int k = 0; k < n/2; k++) {
        odd[k]= x[2*k + 1];
    }

    Complex[] r = fft(odd);

//Combine
    Complex[] y = new Complex[n];
    for (int k = 0; k < n/2; k++) {
        double kth = -2 * k * Math.PI / n;
        Complex wk = new Complex((float) Math.cos(kth),(float) Math.sin(kth));
        y[k]       = q[k].plus(wk.times(r[k]));
        y[k + n/2] = q[k].minus(wk.times(r[k]));
    }
    return y;
}
```

This class is also executed as a thread, the routine can be seen in the method *run*:

```
public void run(){

  while(true)
  {
    try {
      Thread.sleep(1000/60); // approx 1/60 sec refresh
    } catch (InterruptedException e) {
      System.err.println("ERROR: Spectrum thread interrupted");
    }
//If there is a modulator object, take samples of received audio (filtered or not)
    if(mod !=null && mod.getSignal()!=null ){

      for(int i=0;i<N && !audioFiltered;i++)
        data[i]=new Complex((float)(mod.getSignal()[i]*(0.53836-0.46164*Math.cos(2*Math.PI*i
         /(N-1)))),(float)0.0);

      for(int i=0;i<N && audioFiltered;i++)
        data[i]=new Complex((float)(mod.getFilteredSignal()[i]*(0.53836-0.46164*Math.cos(2*
         Math.PI*i/(N-1)))),(float)0.0);

//FFT algorithm
    FFT=fft(data);
//Repaint the panel
    this.repaint();
      }
  }
}
```

Method *paintComponent* is quite similar to the described in 4.3.3, except that only half of *FFT* array is painted, due to the other half correspond to the a copy of the first part. This class must be used only for AX25 signals, because vertical axis is calibrated for half. If pure tones are introduced, amplitude is not accurate.

4.3.5 Eye Diagram

To implement this panel, the class EyePanel is used. It is also a runnable class, and overall, it takes the signal correspondent with 20 bits, it slices the bits and plots them overlapped. The method run shows the routine of this thread:

```
public void run() {

//Create the vectors used
    initWindowData();

  while(true) {
    try {
      Thread.sleep(1000/50); // approx 1/50 sec refresh
    } catch (InterruptedException e) {
      System.err.println("ERROR: Eye Diagram thread interrupted");
    }
//Only access if demod object exists
    if(demod!= null && demod.getSamples()!=null ){
//Add samples
          sent=new float[demod.getSamples().length];
          for(int i=0;i<demod.getSamples().length;i++)
```

```
            sent[i]=demod.getSamples()[i];
//Slice the bits
            bucketData(sent);
      }

      this.repaint();
   }
}
```

The method *paintComponent* is similar to the one explained in 4.3.3, but the signal representation, because bits are overlapped:

```
for (int i=0; i <SAMPLE_WINDOW_LENGTH ; i++) {
  for (int j=0; j < bucketSize; j++) {
    x = border*2 + j*(graphWidth-border*2)/(bucketSize-1);
    double y = graphHeight/2  - Vscale*dataValues[i][j] + 0.5*border;
    if(y > graphHeight-border)y=graphHeight-border;
    if(y < border*2)y=border*2;
```

4.3.6 Eye Diagram Parameters

Five parameters, shown in figure 4.11, are shown: Optimum sampling instant, margin of protection against the noise, answe velocity, amplitude distortion, and rising slope.

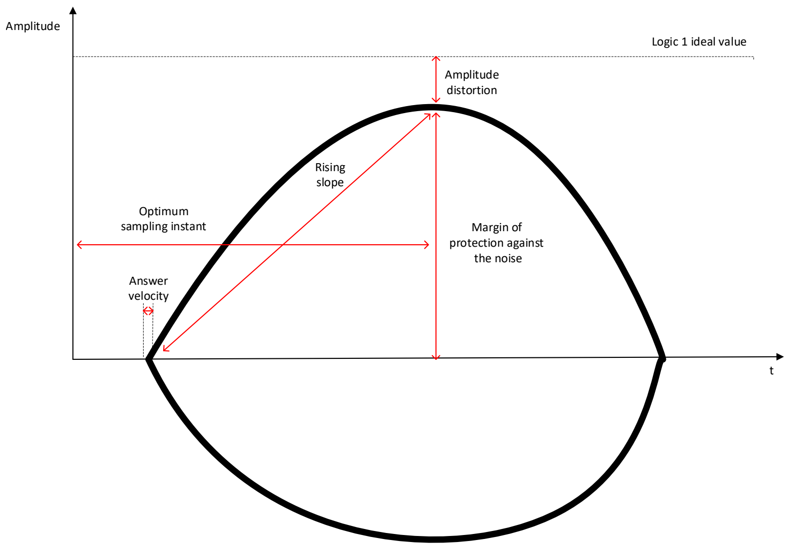


Figure 4.11 – *Diagram with eye parameters.*

They are updated each time *Refresh* button is pressed . These parameters are calculated in class *EyePanel*, concretely during the method *bucketData*:

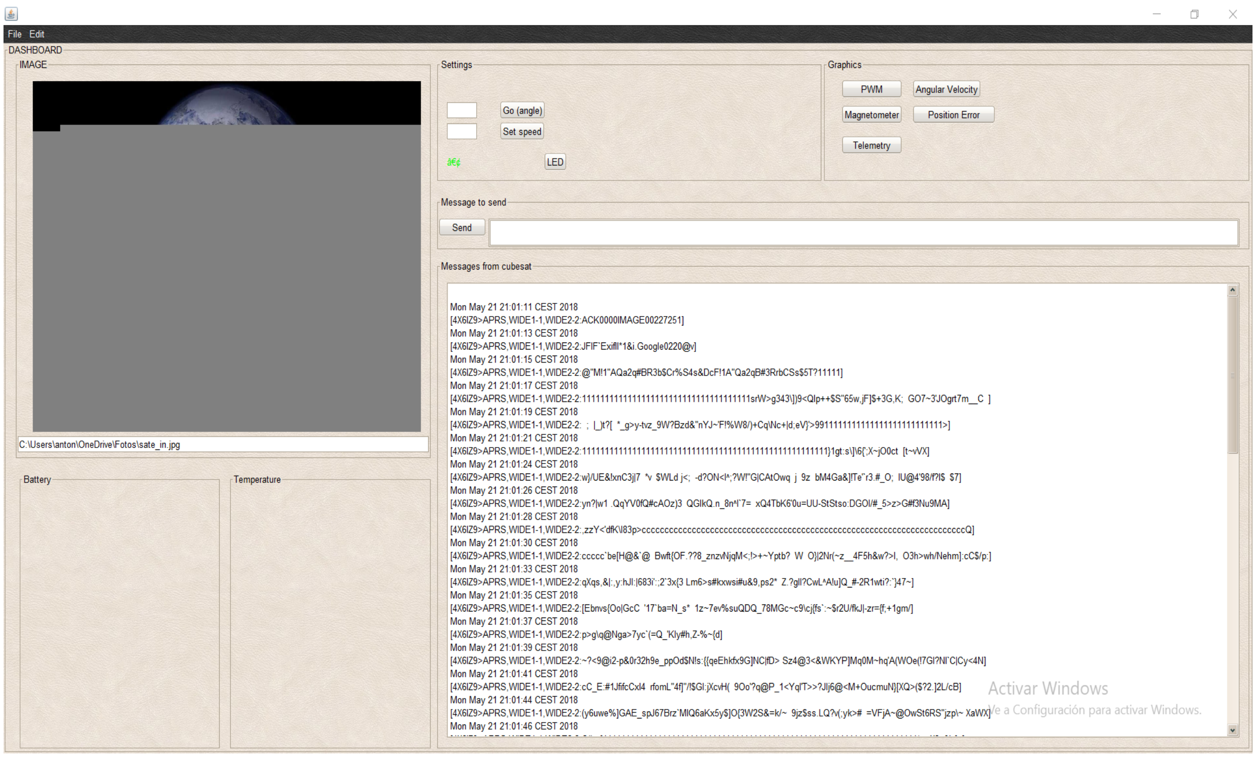
```
protected void bucketData(float[] abData) {
  int k = 0;
//Parameters to keep information
  Jright=0;
  Jleft=0;
  averageMax=0;
  averageMin=0;
  averageSample=0;
  //Bits are sliced

  for (int i=0; i < SAMPLE_WINDOW_LENGTH; i++) {
    minValue[i] = 32768;
    maxValue[i]=0;
      for (int j=0; j < bucketSize; j++ ) {
        dataValues[i][j] = sent[k];
```

```

          if (sent[k] > maxValue[i]) {maxValue[i] = sent[k];maxSample=j;}
          if (sent[k] < minValue[i]) {minValue[i] = sent[k]; minSample=j;}
          if (Math.abs(sent[k])<0.05 && Jright < j && Jleft  < j) Jright=j;
          if (Math.abs(sent[k])<0.005 && Jleft  > j && Jright > j) Jleft =j;
          k++;

        }

      averageMax = averageMax + maxValue[i];
      averageMin = averageMin + minValue[i];
      averageSample=averageSample + maxSample + minSample;

}
  averageMax= averageMax/(SAMPLE_WINDOW_LENGTH);
  averageMin= averageMin/(SAMPLE_WINDOW_LENGTH);
  averageSample=averageSample/(2*SAMPLE_WINDOW_LENGTH);

}
```

And get methods are used to calculate the parameters:

```
//Eye Parameters
public float getOIS() {return averageSample/(demod.getRate()/1000);} //Optimum Instant of
  Sampling
public float getAD() {return Math.abs( 45.5f -averageMax*Vscale);} //Amplitud distorsion
public float getNM() {return averageMax*Vscale;}//Noise Margin
public float getJitter() {return (Jright-Jleft)/(demod.getRate()/1e6f) ;}//Jitter
public float getRS() {return Math.abs((averageMax*Vscale)/(Jright*demod.getRate()/1000));}//
  Rising Slope
```

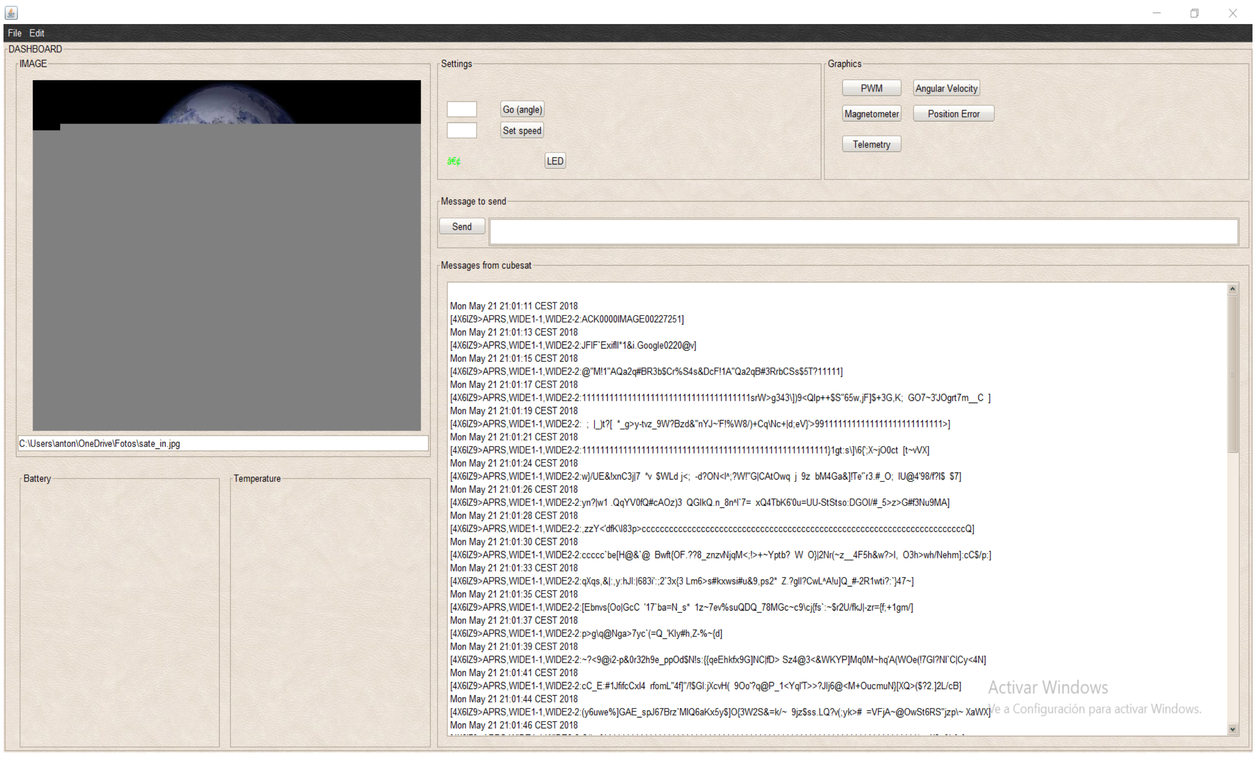
Video 4.1 Reception of an image at 1200 bauds



C:\Users\anton\OneDrive\Fotos\sate_in.jpg

```
Mon May 21 21:01:11 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:ACK0000IMAGE00227251]
Mon May 21 21:01:13 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:JFIF`Exifll*1&i.Google0220@v]
Mon May 21 21:01:15 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:@"M!1"AQa2q#BR3b$Cr%S4s&DcF!1A"Qa2qB#3RrbCSs$5T?11111]
Mon May 21 21:01:17 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:111111111111111111111111111111111111srW>g343\])9<Qlp++$S"65w,jF]$+3G,K;  GO7~3'JOgrt7m__C  ]
Mon May 21 21:01:19 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:  ;  |_)t?[  *_g>y-tvz_9W?Bzd&"nYJ~'F!%W8/)+Cq\Nc+|d;eV}'>9911111111111111111111111111>]
Mon May 21 21:01:21 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:1111111111111111111111111111111111111111111111111111}1gt:s\]\6{';X~jO0ct  [t~vVX]
Mon May 21 21:01:24 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:w}/UE&!xnC3j|7  *v  $WLd j<;  -d?ON<l^;?W!"G|CAtOwq  j  9z  bM4Ga&]!Te"r3.#_O;  lU@4'98/f?l$  $7]
Mon May 21 21:01:26 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:yn?|w1 .QqYV0fQ#cAOz)3  QGlkQ.n_8n^l`7=  xQ4TbK6'0u=UU-StStso:DGOl/#_5>z>G#f3Nu9MA]
Mon May 21 21:01:28 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:,zzY<'dfK\l83p>ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccQ]
Mon May 21 21:01:30 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:ccccc`be[H@&`@  Bwft{OF.??8_znzvNjqM<;!>+~Yptb?  W  O}|2Nr(~z__4F5h&w?>l,  O3h>wh/Nehm]:cC$/p:]
Mon May 21 21:01:33 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:qXqs,&|:,y:hJl:|683i'::2`3x{3 Lm6>s#kxwsi#u&9,ps2*  Z.?gll?CwL^A!u]Q_#-2R1wti?:`}47~]
Mon May 21 21:01:35 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:[Ebnvs{Oo|GcC  '17`ba=N_s*  1z~7ev%suQDQ_78MGc~c9\cj{fs`:~$r2U/fkJ|-zr={f;+1gm/]
Mon May 21 21:01:37 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:p>g\q@Nga>7yc`(=Q_'Kly#h,Z-%~{d]
Mon May 21 21:01:39 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:~?<9@i2-p&0r32h9e_ppOd$N!s:{{qeEhkfx9G]NC|fD> Sz4@3<&WKYP]Mq0M~hq'A(WOe(!7Gl?Nl`C|Cy<4N]
Mon May 21 21:01:41 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:cC_E:#1JfifcCxl4  rfomL"4f]"/!$Gl:jXcvH(  9Oo'?q@P_1<Yq|'T>>?Jlj6@<M+OucmuN}[XQ>($?2.]2L/cB]
Mon May 21 21:01:44 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:(y6uwe%]GAE_spJ67Brz`MlQ6aKx5y$]O{3W2S&=k/~  9jz$ss.LQ?v(;yk>#  =VFjA~@OwSt6RS"jzp\~ XaWX]
Mon May 21 21:01:46 CEST 2018
```

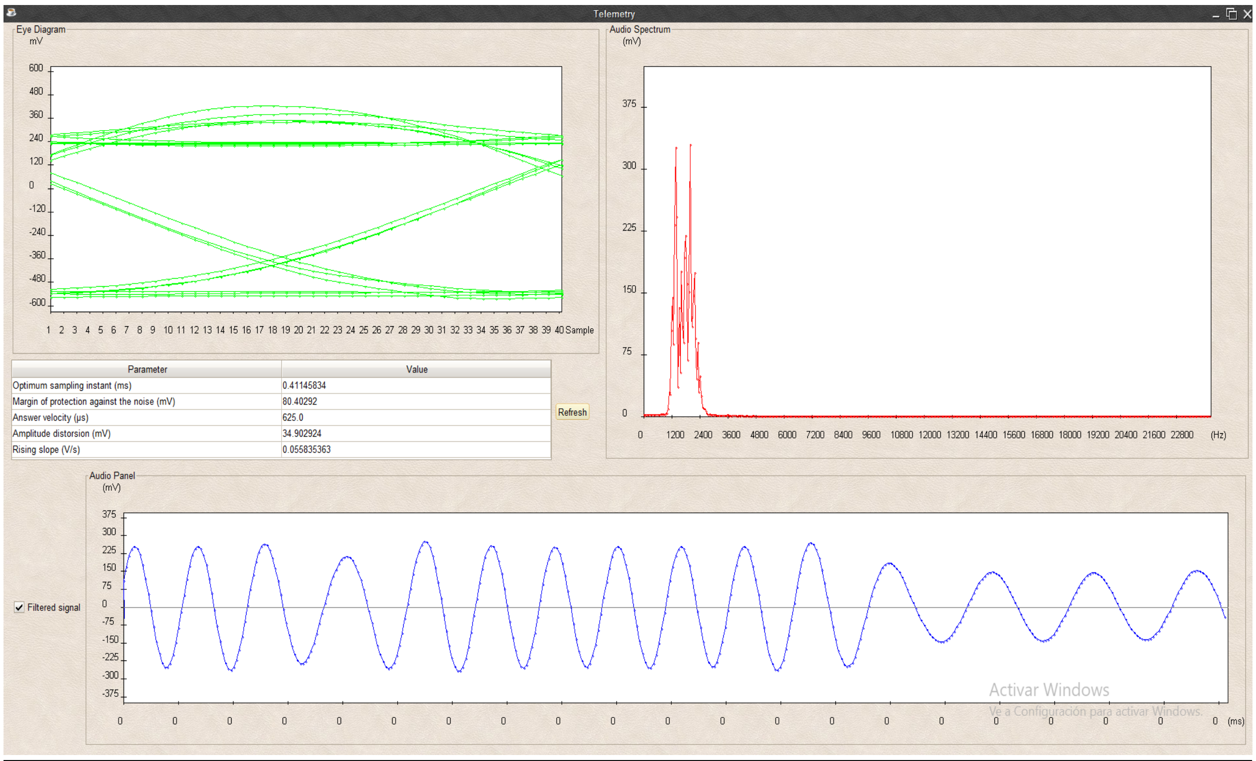
Video 4.2 Reception of an image at 9600 bauds



C:\Users\anton\OneDrive\Fotos\sate_in.jpg

```
Mon May 21 21:01:11 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:ACK0000IMAGE00227251]
Mon May 21 21:01:13 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:JFIF`Exifll*1&i.Google0220@v]
Mon May 21 21:01:15 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:@"M!1"AQa2q#BR3b$Cr%S4s&DcF!1A"Qa2qB#3RrbCSs$5T?11111]
Mon May 21 21:01:17 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:111111111111111111111111111111111111srW>g343\])9<Qlp++$S"65w,jF]$+3G,K;  GO7~3'JOgrt7m__C  ]
Mon May 21 21:01:19 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:  ;  |_)t?[  *_g>y-tvz_9W?Bzd&"nYJ~'F!%W8/)+Cq\Nc+|d;eV}'>9911111111111111111111111111>]
Mon May 21 21:01:21 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:1111111111111111111111111111111111111111111111111111}1gt:s\]\6{';X~jO0ct  [t~vVX]
Mon May 21 21:01:24 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:w}/UE&!xnC3j|7  *v  $WLd j<;  -d?ON<l^;?W!"G|CAtOwq  j  9z  bM4Ga&]!Te"r3.#_O;  lU@4'98/f?l$  $7]
Mon May 21 21:01:26 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:yn?|w1 .QqYV0fQ#cAOz)3  QGlkQ.n_8n^l`7=  xQ4TbK6'0u=UU-StStso:DGOl/#_5>z>G#f3Nu9MA]
Mon May 21 21:01:28 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:,zzY<'dfK\l83p>ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccQ]
Mon May 21 21:01:30 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:ccccc`be[H@&`@  Bwft{OF.??8_znzvNjqM<;!>+~Yptb?  W  O}|2Nr(~z__4F5h&w?>l,  O3h>wh/Nehm]:cC$/p:]
Mon May 21 21:01:33 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:qXqs,&|:,y:hJl:|683i'::2`3x{3 Lm6>s#kxwsi#u&9,ps2*  Z.?gll?CwL^A!u]Q_#-2R1wti?:`}47~]
Mon May 21 21:01:35 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:[Ebnvs{Oo|GcC  '17`ba=N_s*  1z~7ev%suQDQ_78MGc~c9\cj{fs`:~$r2U/fkJ|-zr={f;+1gm/]
Mon May 21 21:01:37 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:p>g\q@Nga>7yc`(=Q_'Kly#h,Z-%~{d]
Mon May 21 21:01:39 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:~?<9@i2-p&0r32h9e_ppOd$N!s:{{qeEhkfx9G]NC|fD> Sz4@3<&WKYP]Mq0M~hq'A(WOe(!7Gl?Nl`C|Cy<4N]
Mon May 21 21:01:41 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:cC_E:#1JfifcCxl4  rfomL"4f]"/!$Gl:jXcvH(  9Oo'?q@P_1<Yq|'T>>?Jlj6@<M+OucmuN}[XQ>($?2.]2L/cB]
Mon May 21 21:01:44 CEST 2018
[4X6IZ9>APRS,WIDE1-1,WIDE2-2:(y6uwe%]GAE_spJ67Brz`MlQ6aKx5y$]O{3W2S&=k/~  9jz$ss.LQ?v(;yk>#  =VFjA~@OwSt6RS"jzp\~ XaWX]
Mon May 21 21:01:46 CEST 2018
```

Video 4.3 Video of telemetry board working


Telemetry
Eye Diagram
mV
600
480
360
240
120
0
-120
-240
-360
-480
-600
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 Sample
Audio Spectrum
(mV)
375
300
225
150
75
0
0 1200 2400 3600 4800 6000 7200 8400 9600 10800 12000 13200 14400 15600 16800 18000 19200 20400 21600 22800 (Hz)
Refresh
Audio Panel
(mV)
375 300 225 150 75 0 -75 -150 -225 -300 -375
Filtered signal
0 (ms)

Parameter	Value
Optimum sampling instant (ms)	0.41145834
Margin of protection against the noise (mV)	80.40292
Answer velocity (µs)	625.0
Amplitude distorsion (mV)	34.902924
Rising slope (V/s)	0.055835363

CHAPTER 5 TEST AND VERIFICATIONS

During this chapter, tests have been performed over the final system, in order to evaluate the operation of it.

5.1 1200 Decoding

During the phase of documentation of the initial system, a test was performed (3.2.2), this test is repeated to know if the decoder works better after the improvements introduced. To perform the test, AX25 frames are recorded in an audio file. Audio is played and frames are decoded by this decoder and two other ones ([2] and [5]). The target of the test is to achieve better results than in the test performed at the beginning of the project.

Results obtained are shown in figures 5.1 and 5.2:

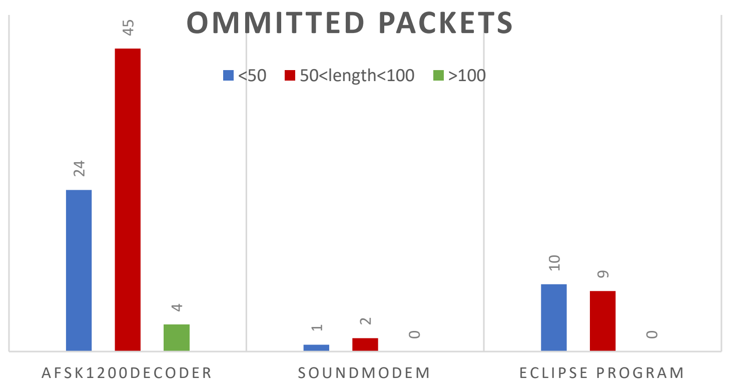


Figure 5.1 – *Omitted packets in last test.*

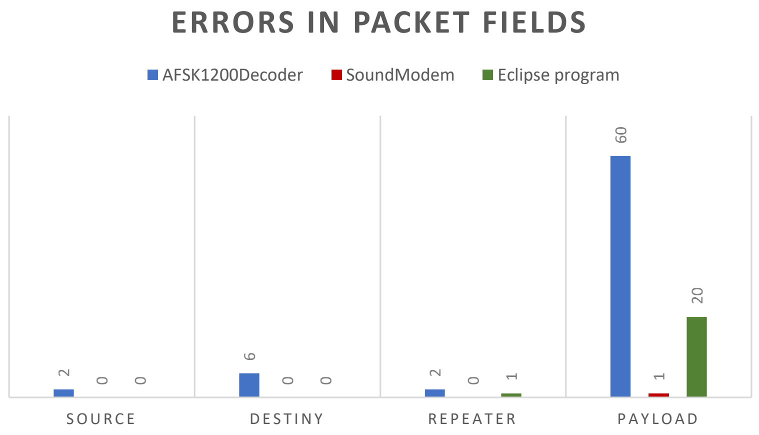


Figure 5.2 – *Corrupted fields in packets.*

5.2 9600 Decoding

The purpose of this test is to evaluate how it works the decoder at 9600 bauds. To do this, frames are generated with a hardware TNC. To do that, it was necessary to addapt the output connector of the TNC hardware (DIN-8 connector) to the input of the computer (jack connector). As shown in figure 5.3

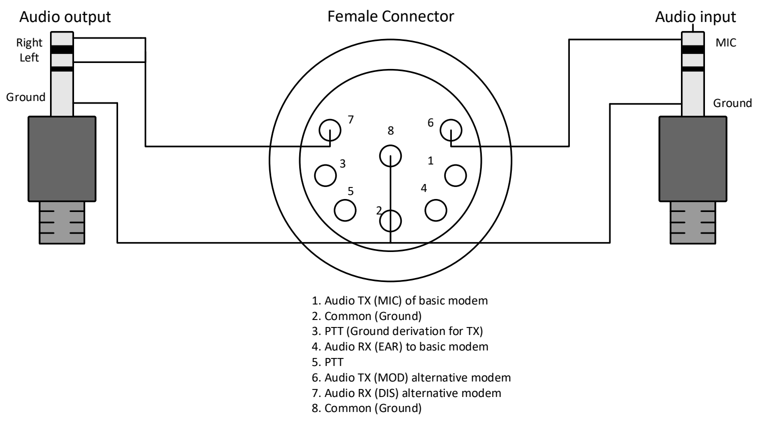

Audio output
Right
Left
Ground
Female Connector
Audio input
MIC
Ground
7
8
6
3
1
5
4
2
1. Audio TX (MIC) of basic modem
2. Common (Ground)
3. PTT (Ground derivation for TX)
4. Audio RX (EAR) to basic modem
5. PTT
6. Audio TX (MOD) alternative modem
7. Audio RX (DIS) alternative modem
8. Common (Ground)

Figure 5.3 – *TNC-Computer connection diagram.*

The connection has been made to the alternative modem, since it is the modem which generates frames at 9600 bauds per second, it can be done in the same way to the basic modem if it necessary to generate 1200 bauds frames.

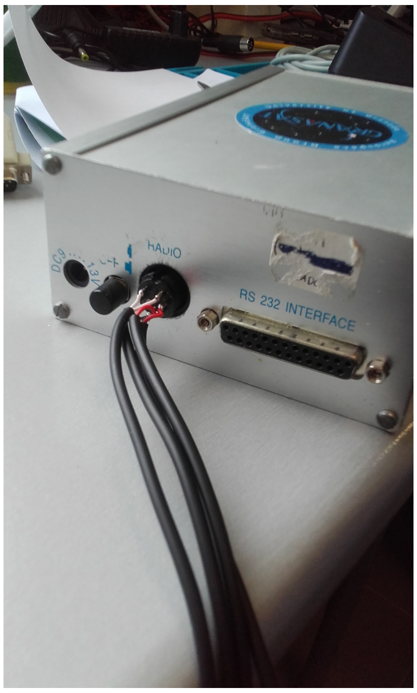


Figure 5.4 – *TNC-Computer connection.*

Once connection is made, 114 frames generated are decoded by the developed modem. Result and original frames are saved in an audio file and compared. The target of the test is to achieve a success of the 95% in the decoding.

Examples of ten frames were recorded, and the audio which contains them can be played clicking on ▶

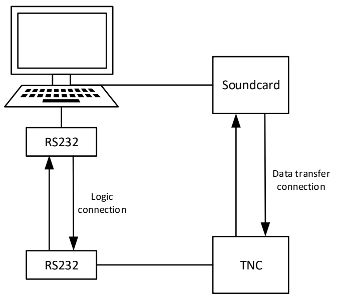


Figure 5.5 – *Test at 9600 bauds per second diagram.*

Results can be seen in figure 5.6, 111 frames were correctly decoded, and 3 frames were corrupted:

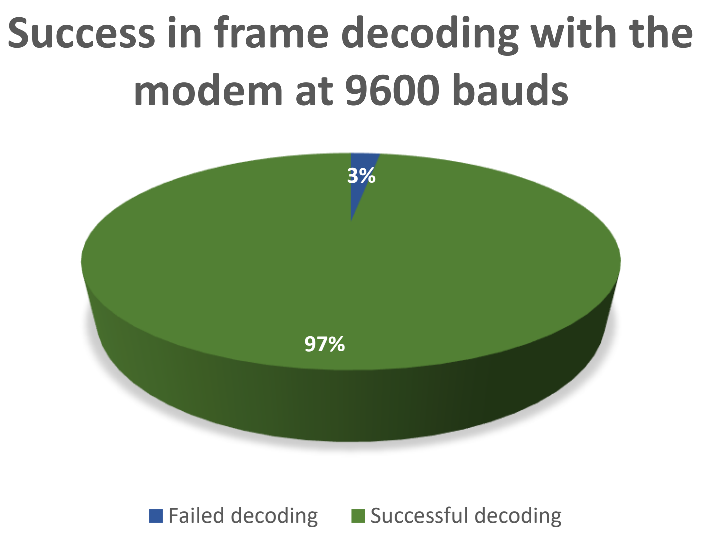


Figure 5.6 – *Success in frames decoding of the modem at 9600 bauds.*

5.3 FEC

One of the main changes were FEC modification. To test its proper working, an image will be sent with FEC at 1200 bauds. It will have success if the image with FEC arrives with no errors.

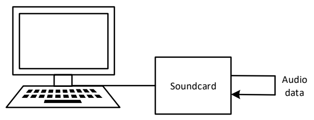
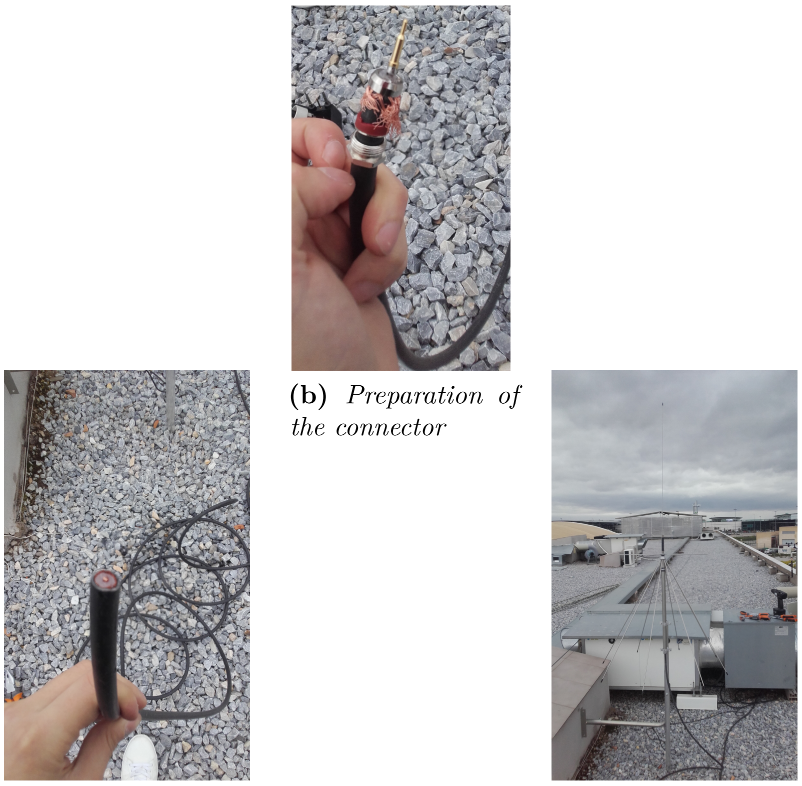


Figure 5.7 – *FEC test diagram.*

After the test, it is checked that the image arrives without errors.

5.4 Decoding signals with Doppler

This test is performed to check the success of the decoder developed decoding real signal. To do this antenna has been connected to the heterodyne receiver (3.3.2):



(b) *Preparation of the connector*

(a) *Cable without connector*

(c) *Antena connected*

Figure 5.8 – *Antenna connection*

The test consists on decoding signals using this decoder and two other ones ([2] and [5]). The target of the test is to obtain similar results.

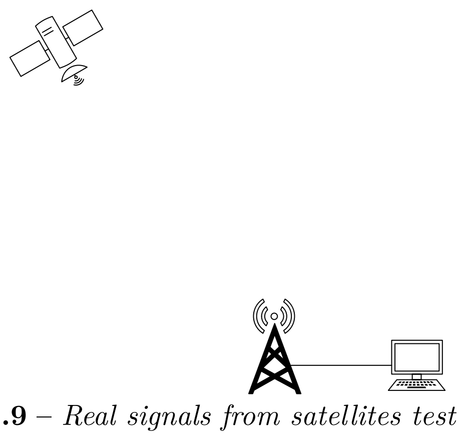


Figure 5.9 – *Real signals from satellites test diagram.*

Spite preparation of the antenna, external problem made impossible to perform this test with the antenna. Because of that, real signals were obtained from youtube, and recorded in the following file: ▶

From the 49 frames, results obtained are:

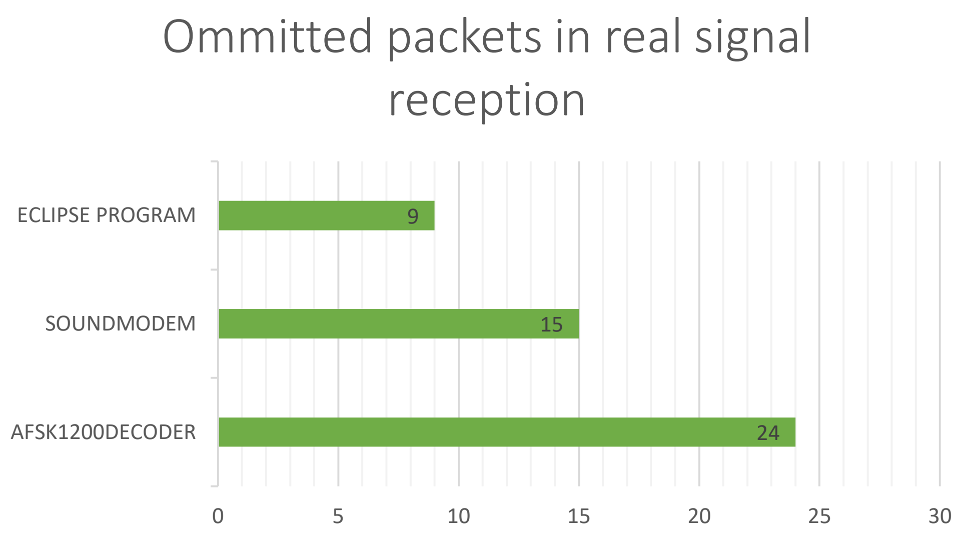


Figure 5.10 – *Result of real signal reception test.*

CHAPTER

6

CONCLUSIONS AND FUTURE LINES

During this project, it has been improved the operation of the modem. Last version of the project was focused on the second layer of the OSI model, and the present one has been centered in the physical layer, achieving the objectives:

- Received signal and demodulator operation are completely characterized by means of the telemetry dashboard.
- The dashboard has been redesigned to be more attractive.
- The modem has increased its operation in the decoding of frames.
- The modem is now able to receive and transmit frames at 9600 bauds.
- The modem is now able to receive frames from satellites.

Once achievements of this projects have been listed, some future improvements and lines of investigation are proposed:

- The implementation of a ARQ system, since satellite channel is very noisy and it introduces a lot of undesired effects which can make difficult the communication. This must introduced in images transmission, since the loss of one frame makes impossible to see the image, even more in this case where differential encoding is used.

- The implementation of sensors in the satellite, to send information about magnetic moment, angular velocity, temperature...
- The design of a dashboard for transmission mode, since, although a field for sending messages is inserted in this dashboard, it would be more suitable two separated dashboards, one for transmitter and another for receiver. Moreover, interactive interface must be created, to be able to send real commands typed from the keyboard.
- An investigation about improvements for the modem at 9600 bauds, since when images are transmitted, some frames are lost.
- The design of a modem at 5 GHz, using GMSK, to achieve a faster transmission.

To end with, I would like to express how this project has affected to my academic enrichment. It is the first time I am able to be involved in a real project, cooperating with other people. Regarding to transversal skills, I have learned how manage project and how to solve difficulties which appear during its development. Regarding to skills related to telecommunication, I am now able to manage real communication systems implemented in software, what will be very useful for my career, as well as, the rest of the practical knowledge acquired.

APPENDIX

A
AX25 AND FX25 PROTOCOL

A.1 AX25

AX25 is a protocol published by American Packet Radio Link League (ARRL). It is a protocol used between radio stations point-to-point or networked communications environment. The protocol only specifies link layer and physical layer functions.

A.1.1 Layer 2

Three types of frames are fixed:

- **Information frame**. (I)
- **Supervisory frame**. (S)
- **Unnumbered frame**. (U)

Flag	**Address**	**Control**	**PID**	**Info**	**FCS**	**Flag**
01111110	112/224 bits	8/16 bits	8 bits	N*8 bits	16 bits	01111110

Table A.1 – *I frame*

Theese messages are encoded with NRZ-I, which encode ’0s’ as no transition and ’1s’ as transitions. Tables A.1 and A.2 show the structure of different fields in an AX25 frame:

Flag	Address	Control	Info	FCS	Flag
01111110	112/224 bits	8/6 bits	N*8 bits	16 bits	01111110

Table A.2 – *U and S frame*

- **Flag**. Used to delimit frames, It is inserted one at the beginning and another at the end of each frame. This field is a fixed sequence 10000001. To avoid considering a flag when it is not, it is needed to add a '0' every five '1s', which will be removed by the demodulator after reception.
- **Address**. Used to identify the source and the destination. It also includes the possibility of adding two repeater sub-fields. Each address consists on six uppercase alphanumeric ASCII characters which form the callsign and a four bit integer called SSID, used to identify stations using same callsign.
- **Control**. Used to identify the type of frame (I, S or U mentioned above).
- **PID**. Used to identify the protocol running.
- **Info**. It contains the useful information. This field has a minimum length of 256 bytes.
- **FCS**. 8 bytes number calculated at sender and receiver, it is calculated as ISO 3309 sepcifies. It uses CRC CITT, a two byte CRC whose generator polinomy is $G(x) = x^{16} + x^{12} + x^5 + 1$.

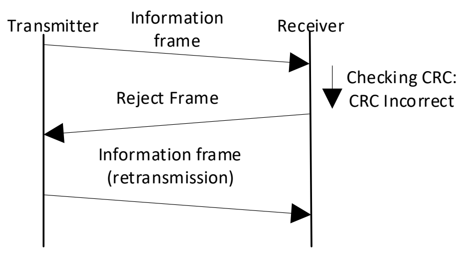

Transmitter
Information frame
Receiver
Checking CRC: CRC Incorrect
Reject Frame
Information frame (retransmission)

Figure A.1 – *Example of incorrect reception.*

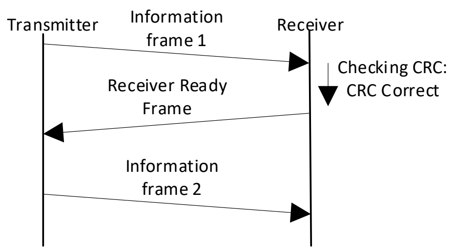


Figure A.2 – *Example of correct reception.*

AX25 can determine the integrity of the message calculating the FCS field. If the receiver detects a CRC error, it responds with a REJECT frame, asking for a retransmission. If it does not find any CRC error, it sends an RECEIVER READY message. After a transmission, the transmitter initializes a time-out counter, when it ends, it sends an RECEIVER READY message asking about the situation of the receiver. The receiver can either RECEIVER READY or RECEIVER NOT READY message, depending on the situation.

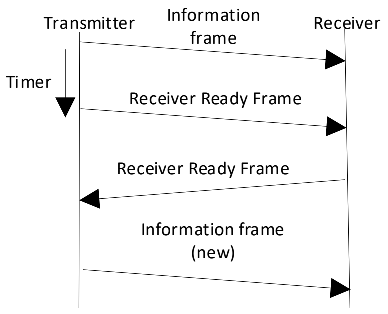


Figure A.3 – *Timer expires in transmitter and frame was received.*

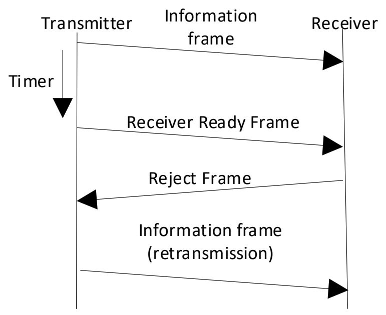


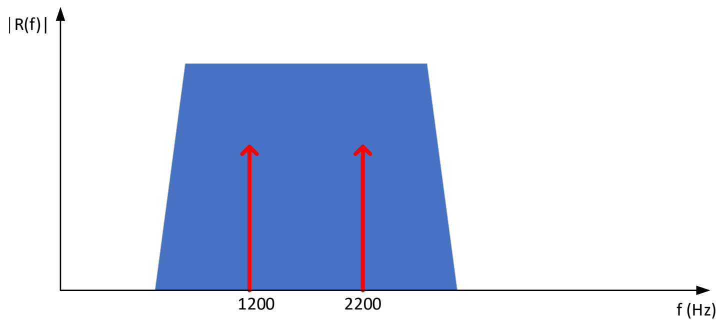
Figure A.4 – *Timer expires in transmitter and frame was not received.*

A.1.2 Phisycal Layer

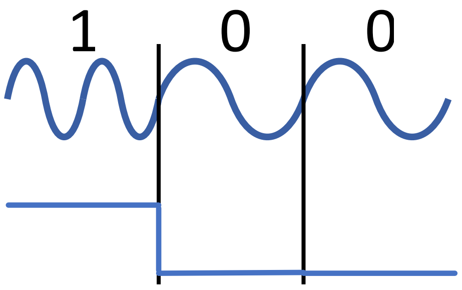
For trabsnission, the Bell 202 standard is used. It uses Audio Frequency Shift Keying (AFSK) to modulate the signal. It is able to send 1200bps This modulation sends a ’0’ as a 1200Hz tone and a ’1’ as a 2200Hz tone. Therefore, the transmitted signal has the form:

$x(t) = Acos(2\pi f_i)$

where A is the cosine amplitude and $f_i = 1200, 2200Hz$. Graphically can be seen as:



(a) *Spectrum*



(b) *Time Domain*

Figure A.5 – *AFSK signal*

As it can be seen in **??**, both carriers are orthogonal, which means that the product between one carrier with the other is 0. This property is used in the demodulator, which will be explained in 3.2. Frequency modulations are optimal for radio transmission, because they have constant envelope, which make them robust against distortions. Moreover, they are noise resistant. However, the main source of signal deterioration is Doppler effect.

A.2 FX25

FX25 is basically the same protocol as AX25 but with FEC. In this system the FX25 frame, which is different from AX25 frame, has been modified, and it is simply an AX25 frame with a FEC as can be seen in Table A.3 and Table A.4

Preamble	Correlation Tag	AX25 Packet Start	AX25 Packet End	Padding	FEC Symbols	Postamble

Table A.3 – *FX25 original frame*

Flag	Address	Control	PID	Payload + FEC	FCS	Flag

Table A.4 – *FX25 modified frame*

In this way, only two flags are used, instead of four, and FCS is performed over the whole packet. The FEC matrix is described in A.5

A linear code block(12,8) is used, which means that each byte is transformed in a codeword of twelve bits, which contains the message and redundant information for error detection and correction. The minimum Hamming dinstance (minimum number of different bits in a codeword) is 4, therefore, it able to detect three bits corrupted and correct one bit corrupted.

To calculate the parity bits, each byte of the payload is logicaly ANDed with each FEC code of the generator matrix. If the number of '1s' in the result is odd, a '1' is added to FEC, if it is even, a '0' is added. Concatenating the result of this operation for each entry of generator matrix, the four parity bits are obtained. They are added to the byte in the payload and sent.

FEC code	Parity bits
11101100	1000
11010011	0100
10111010	0010
01110101	0001

Table A.5 – *FEC matrix*

FEC code	11101100	11010011	10111010	01110101
Original bits	11010110	11010110	11010110	11010110
AND result	11000100	11010010	10010010	01010100
Parity	1	0	1	1

Table A.6 – *FEC generation example [11].*

In the receiver, each twelve bits of payload are ANDed with each entry of the generator matrix. It is repeated the operation of counting '1s' to extract the parity bits. If all the parity bits are '0', no bits are corrupted, in other case, bits were corrupted.

FEC matrix	11101100 1000	11010011 0100	10111010 0010	01110101 0001
Received bits	11010111 1011	11010111 1011	11010111 1011	11010111 1011
AND result	11000100 1000	11010011 0000	10010010 0010	01010101 0001
Parity	0	1	0	1

Table A.7 – *FEC detection example [11].*

To detect the position of the corrupted bit, ANDed 12 bits of payload and rows 1 and 3 of generator matrix must be flipped, and ANDed with the result same operation with rows 2 and 4 (not flipped): The result is the position of the corrupted bit binary coded.

Row number	Bit Sequence
1 inverted	00010011 0111
2	11010011 0100
3 inverted	01000101 1101
4	01110101 0001
AND result	00000001 0000 height

Table A.8 – *Corrupted bit location example [11].*

In Table A.8, the corrupted bit is in the position 8 (considering the most significant bit as first position). In Table A.9 it is checked if it is the corrupted bit, concluding that so it is. In the receiver, each twelve bits of payload are ANDed with each entry of the generator matrix. It is repeated the operation of counting '1s' to extract the parity bits. If all the parity bits are '0', no bits are corrupted, in other case, bits were corrupted.

FEC matrix	11101100 1000	11010011 0100	10111010 0010	01110101 0001
Received bits	11010110 1011	11010110 1011	11010110 1011	11010110 1011
AND result	11000100 1000	11010010 0000	10010010 0010	01010100 0001
Parity	0	0	0	0

Table A.9 – *FEC correction example.*

APPENDIX

B MODEM CLASSES

To complete the description, classes used in the program are being described in more detail:

- **Principal**. It contains the main method, which reads the properties introduced to the program, and performs operations. It is also used as an information handler, in transmission mode, it reads the file to be sent (in case of long message) and forms the packets to be transmitted, for this purpose it uses the class Packet. In reception mode, it receives packet from demodulator, it extracts the payload, and it analyzes the information, performing any operation (save in a file, print data, send ACK...)
- **Packet**. Used to form packet, extract any required field from the packet, check the FCS or print the packet. It is able to add and extract FEC in FEC mode, but it is not able to calculate it.
- **FecPacket**. Same capabilities as class Packet, however, it can perform FEC.
- **Fec**. Used to calculate FEC.
- **Afsk1200Modulator**. It obtains packets for principal, it adds the flags and form the modulated signal. It performs also control flow actions: It does not transmit any message if either other modulator or itself is transmitting.
- **Afsk1200Demodulator**. It performs demodultion and detection of received signal. It implements a receiving filter with or without de-emphasis, it also is able to interpolate the signal (this operation is performed in case of 8KHz audio rate). It keeps information of the state of reception and uses class Packet to form the message.

- **Afsk1200MultiDemoduloator**. It implements two Afsk1200Modulator objects, one with de-emphasis and another without it. In order to avoid replication of received data, it uses an inner handler, which discards data already processed.
- **Afsk1200Filters**. Implementation of both possible receiving filters a correlation filter. They are implented with different impulse response length.
- **Filter**. Used to perform filtering operation.
- **Arrays**. Used to perform operations over arrays, such as contatenation or extraction.
- **SaveImageTheard**. It creates a thread which saves complete and incomplete received images in a specified folder.
- **SendImageTheard**. It creates a thread which splits the image into packets and sends them.
- **Soundcard**. Used to list audio devices in the system, and to insert or extract (normalized) audio in the mixer's soundcard, to be received or sent. For this purpose it uses classes TragetDataLine (Extraction) and SourceDataLine(Insertion).
- **Modulator9600**. Coding and modulation for frames at 9600 bauds.
- **Demodulator9600**. Demodulation and detection of the signal at 9600 bauds.
- **PacketModulator**. Abstract class for implementing modulator classes.
- **PacketDemodulator**. Abstract class for implementing demodulator classes.
- **SoundcardConsumer**. Interface used to implement PacketDemodulator.
- **SoundcardConsumer**. Interface used to implement PacketModulator
- **PacketProducer**. Abstract class for implementing Principal class.
- **Transmission**. Class which implements methods related to transmission. It takes strings which contain packet fields and sends them to the Soundcard object.
- **Dashboard**. Used to implement a window which includes information as status of reception, messages sent or received. It also allows demanding information as angular velocity of satellite, magnetometer information or position error. That information are displayed using classes extended form JPanel class.
- **AudioGraphPanel**. Class used to display received signal in time domain.
- **EyePanel**. Class used to display the eye diagram of the received signal, and to calculate parameters related with it.
- **SprectrumAnalyzer**. Class used to display received signal in frequency domain.

REFERENCES

[1] 9600 baud packet radio modem design. amsat. http://www.amsat.org/amsat/articles/g3ruh/109.html.

[2] csete/qtmm. github. https://github.com/csete/qtmm.

[3] Decompilation and reverse engineering. http://program-transformation.org. http://program-transformation.org/Transform/DecompilationAndReverseEngineering.

[4] Foxtelem software for windows, mac, and linux. amsat. https://www.amsat.org/foxtelem-software-for-windows-mac-linux/.

[5] The software packet-radio tnc. uz7.ho. http://uz7.ho.ua/packetradio.htm.

[6] Spectrum and bandplans. iaru-r1. https://www.iaru-r1.org/index.php/spectrum-and-band-plans.

[7] Videolan, un proyecto y una organización sin ánimo de lucro. videolan. https://www.videolan.org/vlc/index.es.html.

[8] wb2osz/direwol. github. https://github.com/wb2osz/direwolf.

[9] Cuadro nacional de atribución de frecuencias. minetad.gob, 2017. http://www.minetad.gob.es/telecomunicaciones/espectro/Paginas/cnaf.aspx.

[10] Agency, C. S. Cubesat. asc-csa.gc. http://www.asc-csa.gc.ca/eng/satellites/cubesat/default.asp.

[11] González, L. C. *Data Handling, Telemetry Telecommand protocol for Onboard Computer Cubesat.* University of Granada, July 2016.

[12] ICOM. *HF/VHF/UHF Transceiver IC-9100.*

[13] Peces, C. B. *NUEVAS IMPLEMENTACIONES DIGITALES DE SINCRONISMOS DE BIT Y PORTADORA EN MODEM CPM*. Universidad Politécnica de Madrid, 1999.

[14] Sanmonku. ¿la tierra no es azul?: Un satélite japonés revela el color real de nuestro planeta. actualidadrt. `https://actualidad.rt.com/ultima_hora/180135-tierra-color-verdadero-gris-azul`.