UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

TREBALL DE FI DE GRAU

TÍTOL DEL TFG: S-Band communications design and implementation for 3Cat-6

TITULACIÓ: Double bachelor's degree in Aerospace Systems Engineering and Telecommunications Systems Engineering

AUTOR: Mar Munuera Vilalta

DIRECTORS: Hyuk Park
Adrián Pérez

DATA: July 4, 2022

Títol: Disseny i implementació de comunicacions en banda S per a 3Cat-6
Autor: Mar Munuera Vilalta

Directors: Hyuk Park
Adrián Pérez

Data: 4 de juliol de 2022

Resum

El Laboratori de Nanosatèl·lits i càrregues útils (UPC NanoSat Lab) és una iniciativa interdepartamental de l'Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona. La seva activitat principal és el desenvolupament i disseny de missions de nanosatèl·lits, centrant-se en l'exploració de conceptes innovadors tecnològics i el desenvolupament i la integració de subsistemes i càrregues útils per a l'observació de la Terra.
Actualment, el laboratori està desenvolupant el 'Remote Sensing and Interference Detector with Radiometry and Vegetation Analysis', també conegut com a RITA, que és una de les càrregues útils de teledetecció que va ser seleccionada pel segon GRSS Student Grand Challenge al 2019 per volar a bord de l'AlainSat-1. Aquesta càrrega útil s'està desenvolupant sota la supervisió del IEEE GRSS.
Aquesta tesi pretén contribuir al desenvolupament de la missió RITA amb el disseny i la implementació de les comunicacions en banda S de la càrrega útil. Comença amb un estudi de les comunicacions per satèl·lit en banda S amb focus en la seva utilitat per a la descàrrega de resultats científics. Així mateix, es realitza un estudi de l'escenari de comunicacions mitjançant simulacions de l'òrbita del satèl·lit i la realització d'un balanç d'enllaç, que proporciona informació crucial per al disseny del sistema i és decisiu per establir els requisits necessaris. Els estàndards fets servir per l'Agència Espacial Europea per a les seves missions son avaluats per la seva viabilitat en aquesta missió, amb les condicions de contorn que hi apliquen. Així, una variació dels esquemes de comunicació tradicionals és desenvolupada per a la missió de RITA, i el procés de disseny és explicat en detall.
El nucli principal de la tesi consisteix en la implementació i disseny del sistema. Aquest es divideix en tres seccions principals, la capa d'aplicació, la codificació del canal i la capa física. D'una banda, la capa d'aplicació inclou els protocols, el disseny de trama i els sistemes necessaris per transformar els fitxers en paquets i a l'inrevès. D'altra banda, la codificació de canals inclou tots els sistemes de codificació i descodificació per garantir que el sistema sigui capaç de corregir errors que es produeixin en el canal físic de transmissió i recepció. Finalment, la capa física inclou la transmissió de símbols i la recepció de senyals, juntament amb les tècniques de processament de senyal necessàries per garantir que les trames transmeses inicials es puguin recuperar correctament, encara que efectes com el "deep fading" o Doppler hagin afectat la senyal rebuda.
La tesi també presenta diversos tests que comproven el correcte funcionament del sistema utilitzant dos dispositius ADALM-PLUTO SDR, representatius del hardware de vol, per funcionar com a transmissor i receptor.
La tesi conclou amb una verificació del sistema comentant les dificultats i problemes que s'han plantejat durant el desenvolupament i el treball que s'ha de dur a terme abans del llançament.

Title : S-Band communications design and implementation for 3Cat-6
Author: Mar Munuera Vilalta

Advisors: Hyuk Park
Adrián Pérez

Date: July 4, 2022

Overview

The Nanosatellite and Payload Laboratory (UPC NanoSat Lab) is a cross-departmental initiative belonging to the Barcelona School of Telecommunications Engineering. Its main activity is the development and design of nano-satellite missions, with its focus on the exploration of innovative small spacecraft system concepts and developing and integrating subsystems and payloads for Earth Observation.
The laboratory is currently developing the 'Remote Sensing and Interference Detector with Radiometry and Vegetation Analysis', also known as RITA Payload, which is one of the Remote Sensing payloads that was selected by the second GRSS Student Grand Challenge in 2019 to fly on board of the AlainSat-1. This Payload is being developed under the supervision of the IEEE GRSS.
This thesis aims to contribute to the development of the RITA mission with the design and implementation of the payload's S-Band communications. It begins with a study of satellite communications and particularly S-Band, focusing on its usefulness in the downlink of scientific results. Moreover, it presents a study of the communications scenario containing orbital simulations as well as a link budget, providing crucial information for the system's design and decisive in establishing the necessary requirements. The standards used by the European Space Agency for its missions are evaluated for their viability in this mission, with the boundary conditions that apply to them. Thus, a variation of traditional communication schemes is developed for the RITA mission, and the design process is explained in detail.
The main core of the thesis consists of the implementation and design of the system. This is been split into three main sections, the application layer, the channel coding and the physical layer. On the one hand, the application layer includes the necessary protocols, frame design and systems to transform files into packets to be sent, and on the other way around, to recover files from a number of packets. On the other hand, channel coding includes all the coding and decoding systems to ensure that the system is able to recover the initial data if errors occur in the physical channel of the transmission and reception. Finally, the physical layer includes the transmission of symbols and the reception of signals, together with the necessary signal processing techniques to ensure the initially transmitted frames can be recovered correctly even if deep fading's or Doppler shifts have affected the received system.
The thesis also presents several tests that verify the correct functioning of the system using two ADALM-PLUTO SDR devices, representative of the hardware that will be used in the satellite, to work as transmitter and receiver.
The thesis concludes with a verification of the system, commenting on the difficulties and problems that have arisen during its development, and the work that should be carried out before the launch.

”Forget all the reasons it won’t work and believe the one reason that it will.”
- Unknown

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Professor Hyuk Park for his guidance and help during this project, as well as for providing me with the opportunity to participate in a CubeSat mission.

I am also very grateful to the NanoSat Lab team and especially the RITA team for welcoming me with open arms and providing me with this amazing topic. Moreover, I would also like to extend my gratitude to Professor Adriano Camps for creating a magnificent work environment and allowing students like myself to gain invaluable experience and knowledge of the space sector.

Words cannot express my gratitude to Adrian for his endless patience with my never-ending questions and his constant guidance along the project's development. He has encouraged me to challenge myself and I am most fortunate to have had him as an advisor.

In addition, a special thank you to Professor Gregori Vazquez for his invaluable help in understanding the Costas Loop. His kindness and dedication to teaching are inspiring.

Lastly, I would be remiss in not mentioning my family and friends, whose unconditional support has helped me be constantly motivated during this endeavour.

ACRONYMS

ACK Acknowledgement. 13–15, 63, 65–67, 69, 71

ADC Analog to Digital Converter. 29, 82, 94, 97, 98

ARM Advanced RISC Machine. 20

ARQ Automatic Repeat Request. 15, 65, 66, 73, 119

ASK Amplitude-Shift Keying. 84

BCH Bose Chadhuri Hocquengh. 76

BER Bit Error Rate. 49, 85, 86

BPSK Binary Phase Shift Keying. 18, 49, 57, 77, 79, 85, 91, 110, 116

CCSDS Consultative Committee for Space Data Systems. 19, 60, 75, 76, 79, 89, 119

CNAF Cuadro Nacional de Atribución de Frecuencias. 27

DA Data-Aided. 90

DAC Digital to Analog Converter. 29, 82, 94, 97, 98

DD Decision Directed. 90

DoD Department of Defense. 10

ELINT Electronic Signals Intelligence. 17

ESA European Space Agency. 12, 17, 18

FLL Frequency Locked Loop. 91

FPGA Field Programmable Array. 16, 25, 97, 121

FSK Frequency-Shift Keying. 85, 91

FSS Federated Satellite System. 18

GEO Geostationary Earth Orbit. 33

GFET Graphene Field Effect Transistor. 17

GNSS Global Navigation Satellite System. 18

GOCE Gravity Field and Steady-State Ocean Circulation Explorer. 12

GPS Global Positioning System. 32

GRSS Geoscience and Remote Sensing Society. 21

HEO High Earth orbit. 31, 32

ICGC Cartographic and Geologic Institute of Catalonia. 17

ICU Image Classification Unit. 21

IEEC Institut d'Estudis Espacials de Catalunya. 17, 29

ISI Inter-Symbol interference. 86

ITU International Telecommunication Union. 27

JSON JavaScript Object Notation. 13–15, 71, 103

LEO Low Earth Orbit. 16, 31, 33, 39, 44, 46, 50, 53, 69

LHCP Left Hand Circular Polarization. 45

LNA Low Noise Amplifier. 53, 56

LO Local Oscillator. 98

LoRa Long Range Wide Area. 22–25

LOS Line of Sight. 38, 57, 119

MEO Medium Earth Orbit. 31, 32

MIRAS Microwave Imaging Radiometer with Aperture Synthesis. 12

MWR Microwave Radiometry. 22, 24, 25

NACK Non Acknowledgement. 15

NASA National Aeronautics and Space Administration. 7, 8, 16, 27

NCO Numerically Controlled Oscillator. 93, 94, 113

NDA Non-Data-Aided. 90

NDVI Normalized Difference Vegetation Index. 22

NORAD North American Aerospace Defense Command. 35

NRZ Non Return to Zero. 86

NSSTC National Space Science and Technology. 21, 28, 29, 31, 39, 40

OISL Optical Inter-Satellite Link. 18

PLL Phase Locked Loop. 90, 91

PSK Phase-Shift Keying. 85

RF Radio Frequency. 22, 29, 53, 82, 97, 105

RFI Radio Frequency Interference. 22–25

RHCP Right Hand Circular Polarization. 45

RS Reed-Solomon. 76, 77

SCORE Signal Communication by Orbiting Relay Equipment. 8

SDR Software Defined Radio. 29, 55, 56, 84, 89, 94, 97, 98, 105, 107–110

SIGINT Signals Intelligence. 17

SMOS Soil Moisture and Ocean Salinity. 12

SNR Signal-to-Noise Ratio. 25–27, 55–59, 85, 112, 115, 119, 120

SPACECOM United States Space Command. 46

SRRC Square-Root Raised Cosine Signals. 18, 87

SSO Sun Synchronous Orbit. 33–35, 39

STK System Tool Kit. 37, 57

TLE Two Line ELement. 35, 36, 46

TOML Tom's Obvious Minimal Language. 14, 71, 72, 103

TVAC Thermal and Vacuum Chamber. 18

UAE United Arab Emirates. 29

UHF Ultra High Frequency. 29

USNS United States Naval Ship. 10

USSF United States Space Force. 46

VCO Voltage Controlled Oscillator. 91

VHDL VHSIC Hardware Description Language. 121

VHF Very High Frequency. 29

INTRODUCTION

Humans have always had a deep desire to know and understand our planet Earth. However, with the increasing environmental problems that our society is facing such as global warming and deforestation amongst others, this desire has become a necessity.

Satellites have provided a solution to this need, becoming a way for scientists to gather essential data of the Earth that would not be available or much more harder to gather with other technologies. Earth observation missions now have a crucial role: *"to aid to monitor and asses the status and changes present in the environment"*[1].

In the last decades, great technological advances have been made. With the new space era, the world is not limited to using sizable traditional satellites with long development times, but can now reach space through small and cost-effective satellites carrying the latest technologies available.

These nanosatellites have revolutionized the sector, allowing research institutes easier access to space, significantly increasing the number of remote sensing missions around the world.

Furthermore, with the strides made in the sector, remote sensing technologies are now able to gather extensive amounts of data. Some of the technologies that are currently used are [2]: radiometers, imaging radiometers, spectrometers, spectroradiometers, radars, scatterometers, and laser altimeters, amongst others.

Stemming from the large amount of remote sensing techniques available and currently deployed, a necessity to recover the data from the space-borne platform arises. This creates a need to have better space communication systems, that are reliable, efficient and able to transmit information at faster rates, keeping up with the needs of the new and future Earth observation missions.

Objectives

This final degree thesis has as its main purpose to design and implement an S-Band communications system in a 3-Unit CubeSat designed for Earth observation. More specifically, this development is aimed at providing the RITA payload with a functional, performant, and reusable communications subsystem that is able to provide it with enough downlink capacity so that the scientific mission can obtain its targeted objectives. This endeavour can be further divided into several topics which must be properly understood and defined for a correct implementation. These are as follows:

- Obtain sufficient coding skills in the language of choice for the RITA Payload (C++) to contribute to its codebase.
- Understand the state of the art of spacecraft communication systems, their limitations and standards.
  - Obtain knowledge about the expected orbit of the spacecraft (Low Earth Orbit), which drives several other requirements for the mission such as link budget and duration of the passes over a ground station.

  - Understand how the scenario affects the communications link, and carry out a detailed link budget taking into account all the parameters that have an effect on the communications.

- Integrate into the multidisciplinary team developing the payload, learning about the different experiments that are running and their needs, and also about the platform's capabilities and limitations.

- Learn about Software-Defined Radio as the main technology behind the RITA Payload's Radio-Frequency chain, which will be used not only for S-Band communications but for other Remote Sensing experiments.

Methodology

This thesis has been carried out over a period of 10 months in the UPC NanoSat Lab facilities. An iterative and incremental method has been used during its development. This is a combination of two techniques (the iterative method and the incremental build) that is suggested for large software developments. It's main idea consists in the development of the system through repeated iterations and smaller portions at a time. It is iterative due to the fact that each iteration improves the work of the previous. Moreover, it is incremental because each iteration delivers completed work.

Figure 1 illustrates this method. During the development of a project, iterations are carried out, and depending on their results, their following iteration is altered. This way, each iteration improves from the previous and the project grows as they are carried out.

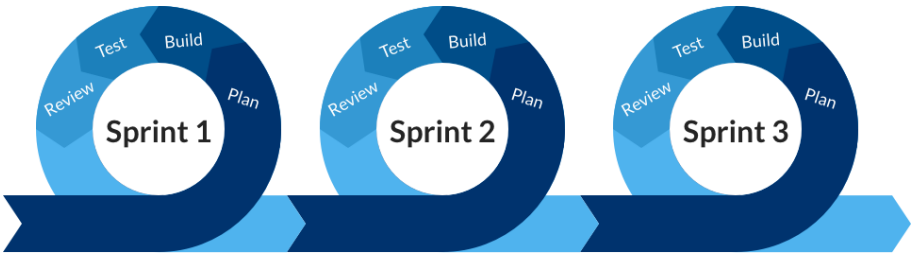


Figure 1: Iterative and incremental method [3]

This is extremely useful in the development because it allows to start with small developmental objectives, learn from their development and apply those lessons to the following iteration, where the requirements can be redefined and the initial design modified if necessary. This on-going process helps to gain perspective on a lot of design requirements that cannot be estimated at the beginning of the project but that arise in its development.

Contents

The thesis has been structured according to the main steps followed during its development and divided into 6 chapters.

The first two chapters depict the theoretical research carried out to gain the necessary knowledge to understand the need for S-Band communications in the RITA payload. Chapter 1 carries out a detailed literature study on space communications, for a better comprehension of the evolution that these have had throughout the years. Furthermore, the CubeSat standard is introduced, providing a deeper insight into RITA payload. Furthermore, Chapter 2 specifically focuses on the RITA payload, providing a profound understanding of the necessity of the S-Band communication system.

Having now a clear idea of the necessity of the system and its basic requirements, Chapter 3 focuses on the S-Band Communication scenario in order to acquire detailed knowledge regarding the radio-frequency scenario and more specific restrictions and requirements that the system design will have to abide by. In it, the used technology and the facilities available for the mission are detailed. Moreover, the orbit that the satellite will follow is studied and several simulations to better represent the scenario are carried out.

Once the communications scenario is studied and understood profoundly, the next step to model the link is to carry out a link budget. This is contained in Chapter 4, that studies all the parameters that have an effect on the communications link and provides essential values to characterize the link and design appropriately the communications system.

The core of the project is contained in Chapter 5. In it, the design is presented with all the theoretical basis behind it carefully detailed. Due to the considerable size of the software, its implementation has been divided into three main blocks: the application layer and network, the channel coding, and the physical layer. These will be illustrated along the chapter, specifically on Chapter 6, which presents the tests that have been carried out during the development, and verify its correct functioning.

The final chapter consists of the conclusions drawn from its development as well as a reflection on future advances.

CHAPTER 1. SPACE COMMUNICATIONS

This chapter aims to introduce satellite communications, depicting its evolution along the years and its current state.

1.1 Introduction to Satellites

NASA defines a satellite as *"a moon, planet or machine that orbits a planet or star"* [4]. There are two different types of satellites: natural, such as the Earth's moon, and artificial, such as the International Space Station.

1.1.1 The theoretical beginnings of satellites

The first time that the idea of satellites being launched from Earth and put into orbit was recorded in the short story "The Brick Moon". Written by Eduard Everett Hale and published in the *Atlantic Monthly* in 1869-1870, this story described the construction and launch of a satellite made of bricks. This would have the purpose of aiding sailors in navigation, bringing up for the first time the idea of communicating through a satellite.

The next marking point in the development of satellites was in 1903, when Konstantin Tsiolkovsky, a Russian scientist, published the Tsiolkovsky formula in an aviation magazine. This equation, depicted in Equation 1.1, which is still the basis of spacecraft engineering nowadays, established the relationship between rocket speed, the speed of gas at the exit, the mass of the propellant and the mass of the rocket. In this equation is, v_0 is the initial velocity of the rocket, M_0 it's initial mass and M_t the mass after a given time t. [5]

$$v(t) = v_0 + v_e * \ln \frac{M_0}{M_1} \tag{1.1}$$

In 1928, the Slovenian Herman Potočnik, also known by its pseudonym Hermann Noordung published his book *"Das Problem der Befahrung des Weltraums - der Raketen-Motor"*, which can be translated as "The Problem of Space Travel - The Rocket Motor" [6]. In it, he described his conception of a space station in detail and suggested that the space station could communicate with the Earth via a heliograph. .

A practical concept of satellite communications was also proposed in a paper entitled "Extra-Terrestrial Relays: Can Rocket Stations Give World-wide Radio Coverage?" published in *Wireless World* in 1945 by Arthur C. Clarke. In it, he proposed that a satellite at 35,786 km would move at the same speed as the Earth's rotation, therefore remaining in a fixed position relative to Earth. He presented computations showing that three satellites spaced in an equidistant manner would be able to provide radio coverage worldwide.

1.2 Satellite Communications

The space industry includes all the economic activities that are related to the manufacturing of components for space, the delivery of these and all related services. This industry

started its development after World War II, when rockets and satellites made its way into military arsenals, and where later on adapted and thought for civil applications [7].

1.2.1 Evolution of satellite communications

This sector has always presented very close ties to governments, having many launch platforms directly being operated by them, such as the space shuttle.

The 4th October 1957, the first artificial satellite was launched successfully by the Soviet Union. Sputnik 1 with a 58 cm diameters and a mass of approximately 83 kg carried four antennas of 2.4 and 2.9 meters of longitude. Moreover, it also included two radio transmitters operating at 20.007 and 40.002 MHz. This satellite only transmitted signals for 22 days but it sparked the beginning of the space race between the United States and the Soviet Union.

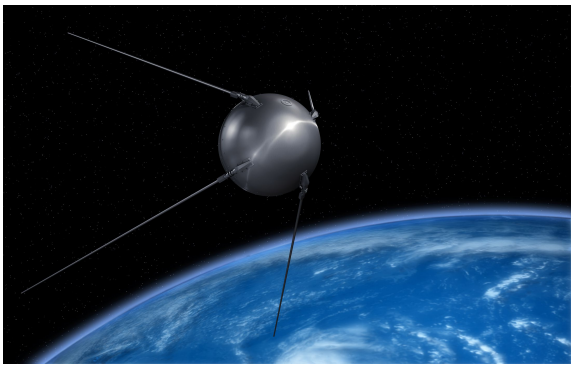


Figure 1.1: Sputnik 1 [8]

A year later, in 1958, the US government's Project Signal Communication by Orbiting Relay Equipment (SCORE) was launched on Florida. This was the first satellite to relay voice signals by broadcasting the message "peace on earth and goodwill toward men everywhere" from US president Dwight D. Eisenhower.

During the 1950s and 60s, engineers Jon Pierce of AT & Ts Bell Laboratories and Harold Rosen of Hughes Aircraft Company developed key technologies for commercial communication satellites. Pierce was the author of the article "Orbital Radio Relays", published in the April 1955 issue of *Jet Propulsion*. In it, he described the principles of satellite communications and presented his computations of the power requirements for signal transmission to various Earth orbits. Moreover, his most remarkable contribution to the field was his design of the "traveling wave tube amplifier", a device that enabled satellites to receive, amplify and transmit radio signals. On the other hand, Rosen focused on the stabilization of satellites and developed a spin-stabilization technology for orbiting satellites.

The establishment of the U.S National Aeronautics and Space Administration (NASA) in 1958 set a marking point in the space industry. NASA's first communication satellite, Echo 1, was developed in collaboration with Bell Lab and finally launched on August 12, 1960. This was a 30.5 meters diameter aluminium-coated balloon passive satellite that had two main functions. From the one hand, it acted as a communications relay, by reflecting radio and radar signals. Particularly, several intercontinental communications where carried out

at the frequencies of 960 MHz and 2390 MHz, proving that microwave communications with satellites in space was possible. On the other hand, it was also employed to track the variations in air density on top of the atmosphere during a determinate period of time and very accurate tracking.

Echo 2 was launched on January 25, 1964 in order to proceed with the passive communications experiments, as well as density measurements and radiation pressure. Moreover, this satellite was also used to carry out geodesic measurements. With an improved design from its predecessor, Echo 2 had a 41.1 meters diameter sphere, rounder and with a more spherical surface than Echo 1. Furthermore, it carried temperature and pressure sensors as well as two transmitters powered by solar cells that could transmit at 136.16 MHz and 136.02 MHz. Overall the Echo satellites were credited with the improvement of satellite tracking and development and testing of technology that was indispensable for the development of active satellites.

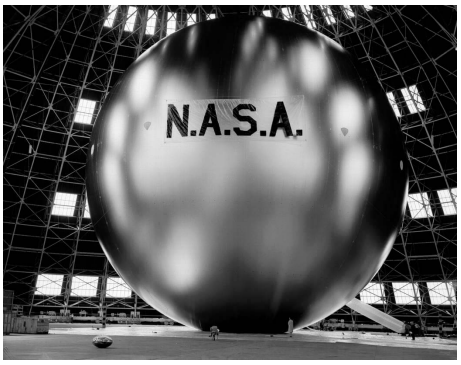


Figure 1.2: Echo 1 during a test inflation in a hangar prior to launch [9]

On July 10th of 1962, the first active two-way communications satellite was launched. This, known as Telstar 1, was developed at Bell Labs by John Robinson Pierce's team. This was a far more sophisticated satellite than the Echo pair, since it was designed to be able to amplify signals received from the ground and relay them back to other ground stations. Positioned at a Low-Earth-Orbit, this 77 kg satellite was the first to transmit live television images between Europe and North America. Furthermore, it also transmitted the first satellite phone call, which consisted of a brief interaction between Fredrick Kappel (AT&T chairman), located at Maine, and U.S President Lydon Johnson, at Washington D.C. Telstar stopped its communication in February 1963. This occurrence is said to be a result of radiation that appeared due to American nuclear-weapons tests in the atmosphere. After Telstar 1, Telstar 2, a model with minor modifications, was launched in 1963 into a higher orbit (10.720 km at the apogee). Moreover, Telstar successors where launched into circular orbits in order to maintain fixed positions in respect to Earth, and with 3 satellites, provide continuous coverage.

Figure 1.3: Telstar-1 satellite assembly [10]

On the meantime, Harold Rosen, from Hughes Aircraft, continued his advancements in the industry with the Syncom satellites, that take their name after ”synchronous communication satellites”. Rosen’s first three satellites had a cylindrical shape, measuring approximately 71 cm in diameter and 39 cm in height, with a total mass of approximately 68 kg each including fuel. His first attempt to place a satellite in a geosynchronous orbit for the first time was unsuccessful with Syncom 1, which was launched on February 14 of 1963 and with whom contact was lost shortly after. Nevertheless, it was with Syncom 2 that this task was accomplished in July 26th 1963. Even though the orbit’s period was 24 hours and the satellite remained at an almost constant longitude, the orbit had a 33º inclination and therefore was not truly geostationary. The satellite was placed over the Atlantic Ocean and Brazil at a longitude of 55º, which allowed to carry out a number of tests. Voice, teletype, facsimile and data transmission tests where carried out between Lakehurst ground station (in New Jersey) and the USNS Kingsport located in Africa’s coast. Moreover, television tests where also carried out, where the transmissions were relayed from Lakehurst to telstar ground station in Maine.

Syncom 3 was launched on August 19th 1963 and was the first satellite placed in a geostationary orbit. This satellite was located over the equator at 180º longitude in the Pacific Ocean. Syncom 3 was used to conduct a number of tests as well as the television transmission of the 1964 Olympic games in Tokyo. The operations control over the satellite was given to the Department of Defense in January 1965, where it was used in DoD’s Vietnam communications. [11]

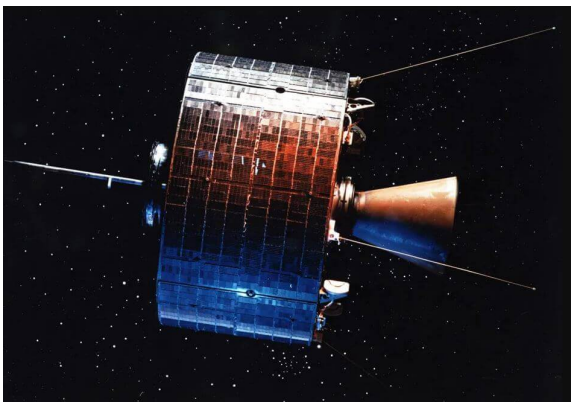


Figure 1.4: Syncom satellite [12]

In 1962, the United States passed the Communications Satellite Act, which put the U.S. in the lead on the development of the satellite communications industry. This act lead to the formation of Comsat, also known as the Communications Satellite Corporation, a private company to represent the U.S. in Intelsat.

Intelsat, an international satellite communications consortium, was formed on Agust 20th, 1964, with 11 signatures to the Interim agreement by Austria, Canada, Japan, the Netherlands, Norway, Spain, Switzerland, the United Kingdom, the United States, the Vatican, and West Germany. Its first satellite, Intelsat 1, also known as Early Bird, was launched on April 6th 1965. Designed by Huges Aircraft Company, it was the first operational commercial satellite and provided telecommunications and broadcasting services between North America and Europe.

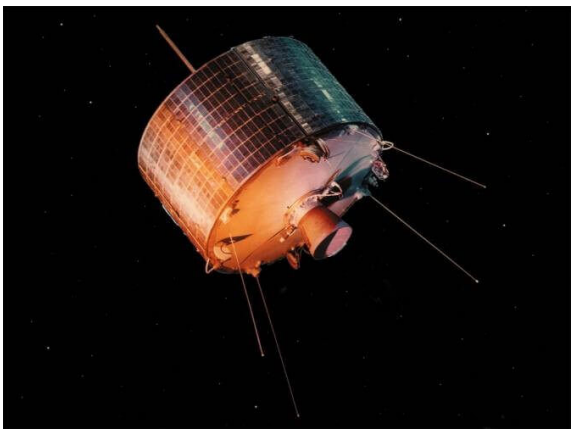


Figure 1.5: Early Bird [13]

Intelsat 1 was followed by Intelsat 2B and 2D, which launched in 1967, covered the Pacific Ocean Region. Two years after, Intelsat 3 F-3 was launched to give coverage to the Indian Ocean Region. The satellites combined in their respective geostationary orbits almost provided a global coverage, as had been envisioned years earlier by Artur C. Clarke. These satellites where also responsible for the transmission of the live broadcast of the first human on the Moon on July 20th 1969, with more than 600 million television spectators.

The Soviet union also had a big impact in the development of satellite communications. They developed the Molniya satellite series and launched them in highly elliptical orbits in order for them to be able to cover the northern regions of the country. The first functional satellite of the Molniya series 1, Molniya 1-1, was launched successfully on the 23rd of April 1965, and by 1967 there where 6 satellites providing the desired coverage. These where used to broadcast nationwide the Red Square annual Parade on October 1st 1967. Molniya 1, which is depicted in Figure 1.6 weighted 16000kg and measured 4.4 meters in height with a 1.4 meters in diameter. These satellites where designed for television, telegraph and telephone usage in the Soviet union. Moreover, these where equipped with cameras in order to carry out meteorological predictions and analysis. The main problem that these satellites had was their useful lifetime, which was limited to 18 months due to the deterioration of the solar panels. This lead to the constant need of renewing the satellites, launching a total of 94 satellites until 2004.

Molniya series 1 was mainly used by the military, and series 2 was then designed for civil use. This new program started in 1974 and it's first satellite was launched in 1974. This

series consists of a total of 16 satellites used to develop ”Orbit”, a national soviet television net.

Finally, Molniya series 3, also known as Molniya-2M started it's development in 1972 and it's first launch was in 1974, having a total of 53 launches until 2003. [14]

Figure 1.6: Molniya-1S satellite [15]

Many satellites have been developed from then on, each with further technological advancements than the previous. ESA in particular has dedicated considerable efforts towards Earth Observation since their first meteorological mission ”Meteosat” in 1977, followed by ERS-1 and 2, Envisat, GOCE and CryoSat-2 amongst others [16].

In particular, SMOS is an ESA mission that stands out, being described as an historical hit or even a technological revolution [17]. This satellite, which was launched the 2nd November 2009, was designed to observe the soil's humidity and the salinity of oceans. This data is not only fundamental for hydrological studies, but will also help comprehend the ocean circulation patterns [18]. In order to do so, this satellite carries a Microwave Imaging Radiometer with Aperture Synthesis (MIRAS) [19]. This L-Band radiometer generates high amounts of data that has to be recovered. For this purpose, the satellite has two antennas implemented: S-Band for TT&C support and X-Band for the scientific data download [20]. Nowadays this mission is still operative and providing invaluable data to scientists.

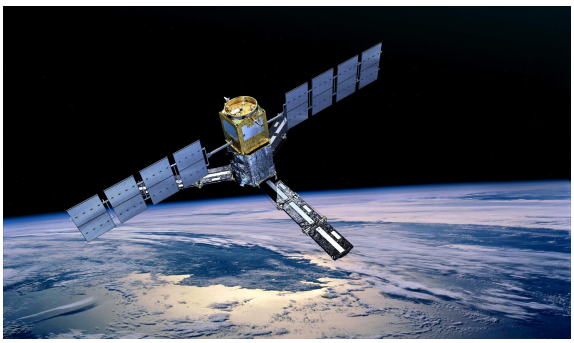


Figure 1.7: SMOS Satellite [21]

1.2.2 Current sate of space communications

Over the last decade, the space industry has had a shift in behaviour, entering the New Space Era, also referred to as the ”Space Renaissance”, which has a new playing field. New technologies and advancements have invited public companies to access space and are now gaining more and more importance, sharing the previously government-dominated sector.

According to the Space Frontier Foundation, New Space can be defined as *”people, businesses, and organizations working to open the space frontier to human settlement through economic development*”. Moreover, the New Space approach was defined by ”Hobby Space” in the ”Best Presentation of Space” in 2007 with the following main characteristics that a New Space project should follow [22]:

- Focus on cost reductions
- Assurances that the low costs will pay off
- Insurances of incremental development
- Entrance into commercial markets with high-consumer rates
- Primary emphasis in operation’s optimization
- Innovation a main focus

1.3 Cube Satellites

1.3.1 The CubeSat standard

In 1999, Professor Jordi Puig-Suari, of California Polytechnic State University, and Bob Twiggs, of Standford University conceived the CubeSat standard, which consisted on a very small spacecraft concept. The aim of this design was to enable students to participate in small space science and exploration missions, taking part in all the phases of spacecraft development (design, testing and operation) and more importantly, to do this in a reasonable amount of time. Even though this original idea was not set out to become a standard, over time it became a very popular satellite model and is now a standarized satellite concept [23].

CubeSats are small satellites multiple of one unit (1U), which is 10cm x 10cm x 11.35cm and weigh up to 1.33 kg. The current specifications define the envelopes shown in Figure 1.8 (1U, 2U, 3U, 6U). More standards are in progress such as 12U and 16U. Moreover, some companies have produced some configurations with up to 27U.

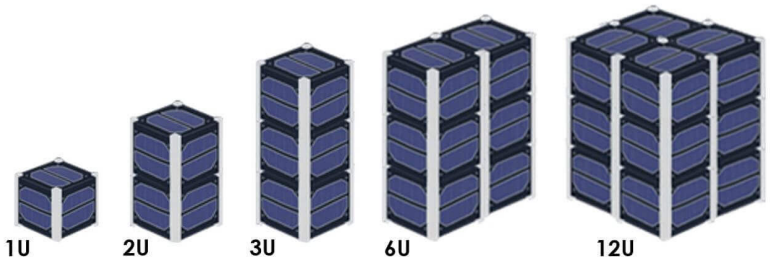


Figure 1.8: CubeSat form factors [24]

The first CubeSats were launched on the Russian Eurocokt on the 30th of June of 2003. From then on, the concept gained more and more popularity in universities and finally space agencies, which showed the reliability that CubeSat-missions had with proper engineering practices. This lead to a posterior takeoff of this technology for Earth Observation applications in 2013. Figure 1.9 shows the increase of launches that CubeSats have experienced over the years, which is a tendency that is expected to continue.

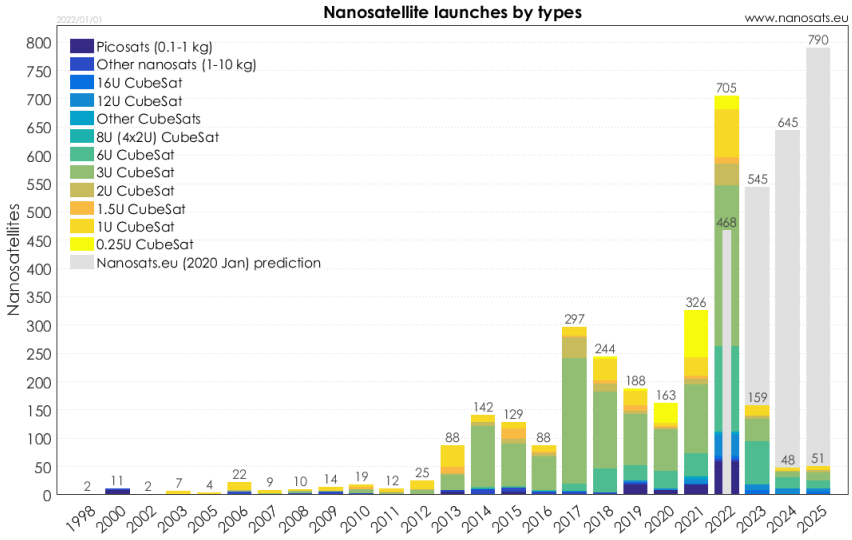


Figure 1.9: Nanosatellite launches by types [25]

Figure 1.10 depicts the organizations that have carried out the previous launches. As it can be seen, until 2013 most of the CubeSats were launched by universities. That year Planet Labs Inc and Spire Global Inc entered the sector and started launching CubeSats. In 2019, both companies had launched the biggest commercial constellations with 355 and 103 CubeSats respectively. It can be seen that since then, companies have continued to be the main responsible for new launches. Nevertheless, universities have also continued to use them, and their numbers also seem to be increasing.

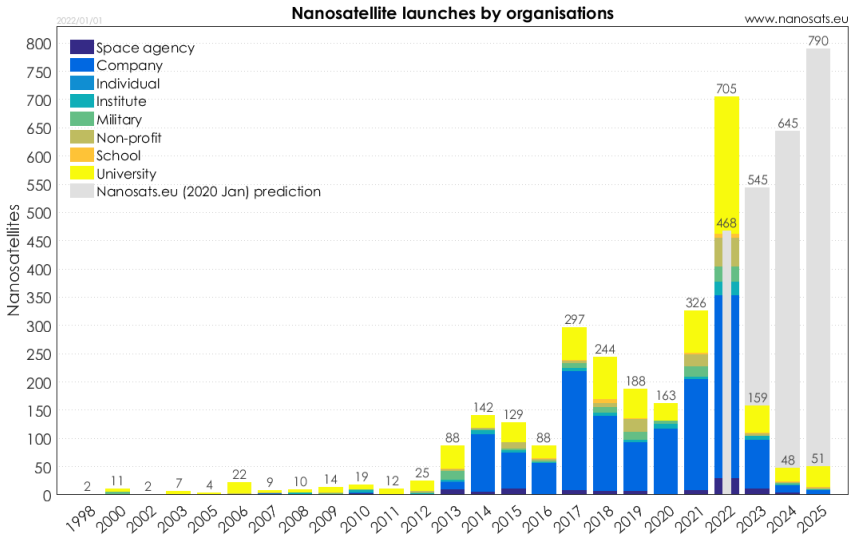


Figure 1.10: Nanosatellite launches by organization [25]

1.3.2 Advantages of CubeSats vs Conventional satellites

Satellites are used for a wide range of applications. Depending on the main requirements that a mission has, CubeSats can be more appropriate than conventional satellites [26].

One of the main and most important advantages of the CubeSat standard is that it offers more accessibility to space to companies of all types and sizes, as well as to research institutes and universities. The space sector was restricted for a long time to the few countries and companies that the financial resources for it. Nevertheless, with the introduction of CubeSats and nanosatellites, many institutions have been able to enter the sector. This has been due to the standardization, that has allowed companies to fabricate components that can be used by many missions with a ”plug and play” standard, substantially reducing costs and development time.

Another main advantage of the use of CubeSats, as has been mentioned above, is their reduced cost. Weight and size are two very important factors of satellites since these characteristics are closely related to the price. For instance, the launch price is dependent on the satellite weight in kilograms, the heavier the satellite, the more expensive the launch will be. Conventional satellites have costs of between 100 and 300 million euros. On the contrary, CubeSats tend to cost less than 500,000 euros, depending of course on factors such as its size, technologies implemented and the complexity of the service to be carried out. Moreover, if the CubeSats in question are part of a constellation, prices are decreased considerably.

The length of the development of missions is also a key advantage of the use of CubeSats. While conventional satellites require very long development times of 5 to 15 years, CubeSats can be developed and launched in months. Nevertheless, the fast development is not only an advantage by itself, but it also permits to have cutting edge technology in space.

Another key aspect to consider in space missions is risk distribution. From the one hand, should the launch be considered, the risk of losing a big and extremely expensive satellite

with a long development is much more considerable than the risk of losing a CubeSat during the launch, since another can be easily developed in a fast manner. Moreover, another aspect to consider with risk are satellite constellations. If a common satellite in a conventional constellation were to fail, the consequences would be catastrophic and it would take millions and years to resolve. Conversely, CubeSat constellations commonly have backup satellites that can fill in the gap should one of the CubeSats fail, and a replica of the failing satellite can be created and launched in a small period of time.

CubeSats also allow adaptability and flexibility that other satellites cannot offer, since they allow companies to implement new functionalities and services in a very reduced amount of time expanding their business. Moreover, many CubeSats using FPGA platforms can be reconfigured in space.

The fact that companies can afford to have CubeSat constellations of their own reduces the need to contract third-party satellite networks. This does not only allow for greater independence but also a greater data security, and adds a competitive advantage with respect to competitors.

CubeSats, nevertheless, do present some limitations. For instance, their reduced mass budget does not permit them to have a protection against radiation, which shortens their lifespan. Another constraint is that their small size does not allow them to have big instruments such as antennas, which limits their precision. However, these difficulties and many others are being overcome and this standard is being more and more adopted in the fields of astronomical and telecommunications missions. [27]

1.3.3 CubeSat applications

Due to its many advantageous characteristics, CubeSats have a wide range of applications [24].

The most common and popular application up to date is Earth observation. CubeSats are commonly launched in LEO orbits, which offer a very wide range of visibility of the Earth, and being the closest orbit to the ground, also offers the best possible resolution for observation. It provides a very effective manner of collecting agricultural, forestry, geological and environmental data of the planet. For instance, NASA developed ”RainCube”, which is a weather satellite whose objective was to measure storms [28]. Another similar CubeSat mission is ”IceCube”, also developed by NASA, it aimed to test instruments ability to make space-based measurements of the small frozen crystals that make up ice clouds [29]. Countless other CubeSats missions have been and are currently being designed for this purpose, such as the one that this thesis is based on.

CubeSats have also been extremely useful in communications as well as in the development of the Internet of Things, and have laid its foundations. For the same reasons that have been stated previously, CubeSats have a bright future in this area since, through the use of constellations, they allow to connect remote parts of the world that do not have access to terrestrial communications, just like conventional satellites do.

Geolocalization is another of the many CubeSat applications. Due a CubeSat constellation’s ability to monitor large extensions of ground permits them to locate assets such as vehicles or aircraft that cannot be found as easily by other current technologies.

CubeSats have also opened doors to space observation programs, interplanetary mis-

sions, systems testing and biomedical research amongst others. For instance, "Mars Cube One (MarCO)", launched in 2018, was the first interplanetary mission to use CubeSats to leave Earth. Two CubeSats, named EVE and WALL-E, where used as communication relays during the Mars landing in November 2018. They beamed back data at all the landing stages in near real time. This successful mission was carried out with experimental technology that reduced the budget to a fraction of typical space missions ($18.5 million) [30].

Satellites have had Signals Intelligence (SIGINT) as a mission for many years, particularly for Electronic Intelligence (ELINT), which studies the analysis and location of radio frequency signals. CubeSats constellations can monitor radio signals, that in the event of a disaster can be used to provide useful information.

1.3.4 NanoSat Lab

The Nano-Satellite and Payload Laboratory is a cross-departmental initiative that belongs to the Barcelona School of Telecommunications Engineering. It's main activity is the development of nano-satellite missions. The lab studies innovative small spacecraft systems and develops and integrates payloads destined to Earth Observation.

The laboratory is a multi-disciplinary environment where students from various degrees and backgrounds, ranging from bachelor, to masters and PhD, collaborate and participate in several missions.

1.3.4.1 Missions and projects

Some of the projects that have been carried out in the laboratory are the following:

- **3Cat-1** [31]

  CubeCat-1 was the first satellite developed in the laboratory, and was aimed at testing and investigating the CubeSat standard capabilities. This 1-Unit CubeSat weighted 1.2 kg and was launched successfully the 29th of November in 2018, sponsored by "*Institut d'Estudis Espacials de Catalunya*" (IEEC). The payloads implemented in the nano-satellite were: an eternal self-powered beacin demonstrator, a series of CellSat Solar Cells, a MEMS-based monoatomic oxygen detector, a Graphene Field Effect Transistor (GFET), a low-resolution CMOS camera and a Geiger counter and a wireless power transfer experiment.

- **3Cat-3** [32]

  CubeCat-3 is a 6-Unit CubeSat that was conceived for the Cartographic and Geologic Institute of Catalonia (ICGC) to analyze the feasibility of using small satellites with a multispectral imaging payload. The project was stopped at the design stage due to lack of funding.

- **3Cat-4** [33]

  CubeCat-4 is a 1-Unit CubeSat that was elected in the ESA Fly Your Satellite program and has the support of the European Space Agency (ESA) and their sponsor-

ship for its launch. This satellite, that focuses on to Earth Observation purposes, is expected to be launched with Ariane 6.

- **FSSCat** [34]

  The FSSCAT mission, also known as 3Cat-5/A and 3Cat-5/B, is the winner of the 2017 Copernicus Master ESA Small Satellite Challenge. It is composed of two 6-Unit CubeSats containing a GNSS-Reflectometer, an L-band radiometer and a multispectral optical payload to measure soil moisture, ice extent and thickness. Moreover, it also has a technology demonstrator of an Optical Inter-Satellite Link (OISL) and a proof-of-concept of a Federated Satellite System (FSS). This mission was launched on September 3rd 2020.

1.3.4.2 Facilities

The laboratory has the necessary equipment to carry out the assembly, integration and testing of CubeSats and their subsystems.

The laboratory has a ISO 8 clean room containing all the necessary tools for integration and testing. This is shown in Figure 1.11.

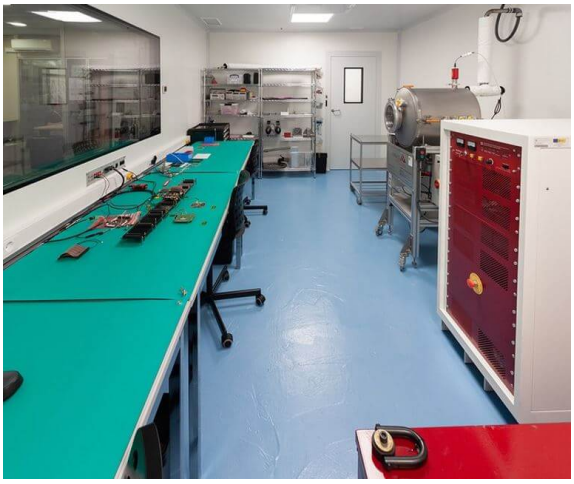


Figure 1.11: Clean room [35]

The most notable machinery for testing is the following:

- **Thermal and Vacuum Chamber (TVAC).** This chamber emulates outer space conditions and is key in performing environmental campaigns with CubeSats.
- **Vibration Table.** The Electrodynamic Vibration Table is used to emulate launch conditions.

(a) TVAC [36]

(b) Vibration table [37]

Figure 1.12: Testing instrumentation

CHAPTER 2. RITA PAYLOAD

This chapter aims to introduce the RITA Payload scientific mission and its need for an S-Band communications link.

2.1 Introduction to AlainSat-1

In 2019, the GRSS Student Grand Challenge selected the RITA Payload as one of the Remote Sensing payloads to fly on board of the AlainSat-1, a 3-unit satellite developed by the National Space Science and Technology Center (NSSTC) in the United Arab Emirates University.

The 3 teams designing each Earth observation payload have worked under the supervision of IEEE GRSS, aiming to get their payloads ready for launch, which is currently expected to be in Q2 2023.

2.1.1 Indeonesia's Telkom University: Locana team

Locana Bhumi is one of the payloads that will fly onboard the AlainSat-1. It's main objective is to monitor cloud coverage along several regions of the planet, such as Australia, the United Arab Emirates, Oman and Indonesia amongst others, to later study the effect of cloud coverage on climate. In order to carry out this mission, the payload carries a small serial camera equipped with a 1/4-inch 5-megapixel OV5642 image sensor.

2.1.2 Japan's Kyutech Institute of Technology: ICU team

The ICU, also known as Image Classification Unit, is the 3rd payload flying onboard the AlainSat-1. This payload aims to classify the images taken in the Locana payload into predefined categories and detect the clouds in them.

2.1.3 NanoSat Lab: RITA team

The RITA payload, also known as "Remote sensing and Interference detector with radiomeTry and vegetation Analysis", has been designed as a compact evolution of past NanoSat Lab missions, including an L-Band Microwave Radiometer, a Hyperspectral camera, and an RFI detector and classifier as its main instruments. As stated before, this payload is the focus of the thesis, and will be explained in detail in the following sections.

2.2 RITA's mission analysis

The RITA payload aims to gather data regarding the state of vegetation, desertification and floods amongst others. It has four main mission objectives that are as follows:

- Acquisition of soil moisture measurements: This data plays a key role in several fields such as agriculture, drought and flood forecasts amongst others.
- Acquisition of vegetation measurements: The NDVI provides an indicator of the state of terrains. The study of this data over time can help scientists to detect climate trends and changes in biodiversity.
- RF Interference detection: RF interference is nowadays a concerning topic in Earth observation and navigation since it corrupts signals and can render them impossible to recover.
- LoRa Experiment: This will test the retrieval of measures from sensors located in rural areas and carry out on-demand executions.

The payload is equipped with the following instruments to fulfill the scientific objectives once the satellite is in orbit:

- **Software-Defined Radio for microwave radiometry**

  The use of GNSS Reflectometry and L-band radiometry has shown great potential in improving the measurements of soil moisture and salinity. RITA payload carries 3 L-band MWR antennas in order to obtain soil moisture measurements. RITA Payload's radiometer is based on a Total Power Radiometer (TPR) with frequent calibration. It's two main requirements are that the instrument sensitivity must be lower than 1 K and it's accuracy lower than 2K. Moreover, for its correct functioning and operation the device must know the antenna temperature and its stability to calibrate appropriately.

- **Hyper-spectral camera**

  Hyper-spectral cameras collect and process information across the electromagnetic spectrum. This camera will be used to acquire images of interest regions, which will allow to obtain information of the ground vegetation through vegetation indexes, to latter extract data regarding the biomass and algae blooms. The specific camera used has a 16 mm Kowa lens LM16HC and a photonfocus MV1-D2048x1088-HS05-G2 sensor. The camera has a total of 25 bands, ranging from 650 to 975 nm.

- **Radio-Frequency Interference detection**

  RFI measurements can be beneficial for future missions using L-band MWR to obtain data. The payload will carry out normality tests and the obtained signals will be post-processed and stored if a strong RFI is obtained.

- **LoRa transceiver**

  RITA has included an SDR-based LoRa transceiver. This module has as its main objective the verification of this novel technology, as well as the retrieval of soil moisture measures from remote sensors. Furthermore, on-demand executions will also be performed by the IoT sensors.[38].

This instruments generate high amounts of data that have to be transmitted to the ground. For instance, it is expected that each image taken by the hyper-spectral camera will have 2 MBytes and that each orbit around 16 images will be captured. This would mean that

approximately 32 MBytes of data are generated every orbit, and a higher quantity of data is expected considering the other instruments. This amounts to an extremely large sum of data that has to be transmitted back to the ground station for its latter analysis and use. From this arises the need to include a communications link, able to deal with these large quantities of data. An S-Band communications link was chosen as a capable and adequate link for the data retrieval. The reasoning for choosing this particular band instead of others is explained in Chapter 3.

Figure 2.1 depicts the location of the RITA Payload within the AlainSat-1. Moreover it indicates the location of the LoRa patch Antenna as well of the radiometer Antennas used for the mission and the location of the communications' S-Band antenna.

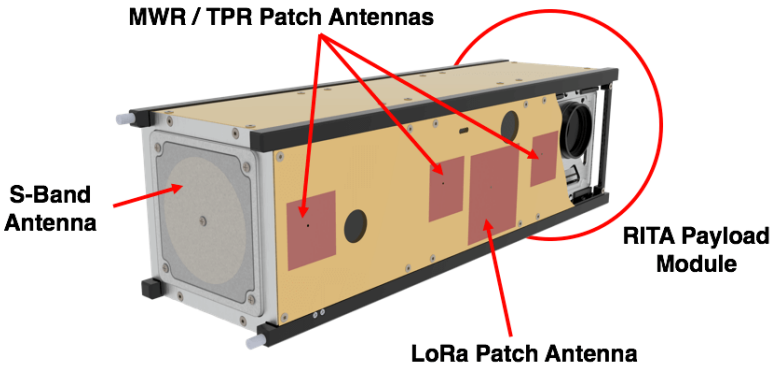


Figure 2.1: AlainSat-1 model

2.2.1 Target areas

In order to better carry out RITA Payload's mission several areas have been chosen as targets to study. These have been chosen for their particular terrain characteristics as good candidates to carry out the following:

- Soil Moisture Measurements
- Vegetation analysis
- Algae bloom detection
- Oil spills detection
- RFI detection
- Remote Sensor data retrieval using LoRa

Figure 2.2, shows the areas to be studied by RITA payload, that extend all over the globe. A detailed table indicating which areas will be used for each experiment can be found in Appendix A.

Figure 2.2: RITA Payload's target areas [39]

2.2.1.1 Operational modes

The RITA payload has been designed with 5 different operational modes in order to carry out all of its functions. These are:

- **CAM** : This mode does the following functions:
  - Calibration
  - Vegetation analysis
  - Algae detection
  - Oil spills detection
- **MWR + RFI** : This mode consists in the following:
  - Microwave radiometry
  - RFI detection
  - Maps classification
- **CAM + MWR + RFI** : This operational mode's main purpose is to carry out pixel downscailing for soil moisture measurements.
- **LoRa** : This operational mode does a multilevel data acquisition over Spain
- **S-Band** : This operational mode consists activates the S-Band communications and allows the payload to receive instructions and transmit the collected data.

Each mode requires the activation and use of specific devices. These are depicted in table 2.1.

Table 2.1: RITA Payload Operational modes

	Devices					
Mode	**Front End**	**RFI**	**HS Camera**	**LoRa**	**S-Band**	**FPGA**
CAM			X			X
MWR + RFI	X	X	X			X
CAM + MWR + RFI	X	X				X
LoRa				X		X
S-Band					X	X

CHAPTER 3. S-BAND COMMUNICATIONS SCENARIO

This chapter introduces S-Band communications and studies the radio-frequency scenario, fist by analyzing the satellite antenna as well as the ground stations and then by studying the satellite's orbit. Several simulations are carried out to to better understand the conditions and restrictions of the communication link.

3.1 Introduction to S-Band

The frequency spectrum is regulated by the International Telecommunications Union (ITU). From all the available bands, there are a few allocated for satellite applications. This is depicted in Figure 3.1.

Choosing an adequate frequency band is very important for the correct functioning of the communication system. The higher the frequency is, and therefore the lower the wavelength, the higher bandwidth assigned to the channel is. While this seems to be the optimal case, higher frequencies also come at a cost, being more susceptible to signal degradation due to rain fade as well as other losses.

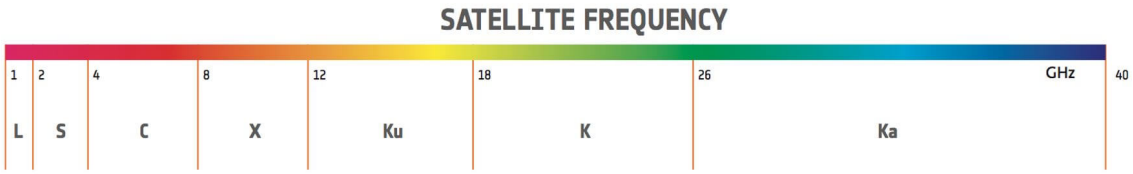
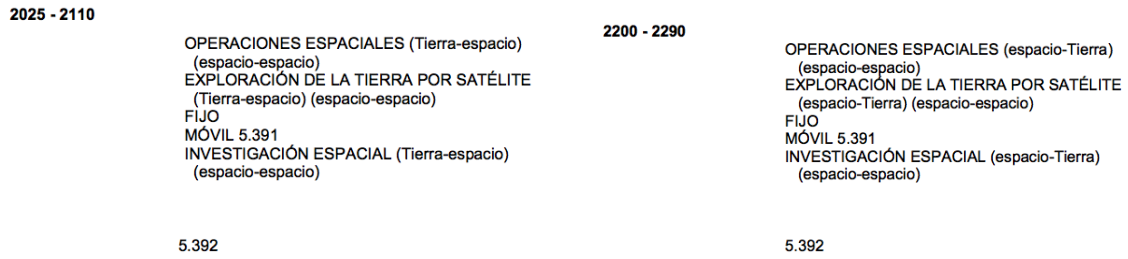


Figure 3.1: Frequency spectrum for satellite applications [40]

CubeSat communications are commonly done with UHF, S or X Bands. The first is not enough to provide the necessary data rate for the transmission of the collected data of the RITA mission. S-Band, having a higher bandwidth is a more appropriate band, full filling the necessities of the payload. X-Band presents a higher bandwidth than S-Band, however, it is more difficult to implement and since S-Band fulfills the needs of the payload it has been deemed as appropriate.

RITA mission has then chosen to carry out the communications with S-Band. CNAF allocates this band in the frequency spectrum from 2025 to 2110 MHz for the uplink and from 2200 to 2290 MHz for the downlink, as is noted in Figure 3.2. This band is nowadays used for weather radars, surface ship radars as well as communications satellites. For instance, NASA uses S-band for communication with the ISS and the Space Shuttle. Furthermore, Inmarsat and Solaris mobile where given a 30 MHz portion of the spectrum in 2009 [40].



(a) Uplink frequency range (b) Downlink frequency range

Figure 3.2: S-Band CNAF allocation [41]

3.2 Scenario description

Having defined the frequency band used for the communications link, a frequency filling must be done in order to obtain S-band frequency channels for the communication that will not interfere with other communications already in place.

In this case, the frequency filling was requested by NSSTC and the resulting summary is shown in Table 3.1. This table depicts the available frequencies that have been allocated for S-Band communications.

Table 3.1: Frequency filling summary report

B1a Beam designation	BR7a Group id.	BR47 Frequency band (MHz)
ASU1	65	2028.2 - 2029.4
ASU2	66	2068 - 2069.2
ASU3	67	2088.2 - 2089.4
ASD1	68	2266.8 - 2269.2
ASD2	69	2277.8 - 2280.2
ASD3	70	2258.8 - 2261.2

As it can be seen, there are three possible channels both for the downlink and uplink that can be used. The middle channels have been chosen to carry out the design and their respective frequencies are presented in Table 3.2. These are required for choosing the link components and later carrying out the link budget.

Table 3.2: S-Band frequency parameters

	Uplink	Downlink
Channel	66	69
Frequency spectrum (MHz)	2068 - 2069.2	2277.8 - 2280.2
Central frequency (MHz)	2068.6	2279
Bandwidth (MHz)	1.2	2.4

3.2.1 Satellite Antenna

In order to choose an appropriate antenna for the satellite, the scenario parameters must be carefully considered. Since the CubeSat has to be able to transmit as well as receive information, the best choice is to get a duplex patch antenna. This antenna must contain the frequencies depicted in Table 3.2.

The chosen antenna is the NanoCom ANT2150 DUP, which is produced by GOMSpace. This antenna has the frequency bands shown in table 3.3, which makes it an appropriate choice. Moreover, it has adequate gain values and measures that are appropriate for the satellite. Further information is contained in its datasheet, available in appendix I.

Table 3.3: Antenna parameters

	ANT2150 DUP	**Unit**
TX band	2200 - 2290	MHz
RX band	2025 - 2120	MHz
Tx Gain	8.5	dB
Rx Gain	7.75	dB

3.2.2 Ground Stations

There are two ground stations available for S-Band use, one provided by NSSTC located in its center in the UAE and the other, provided by NanoSat Lab, and located the Montsec Astronomical Observatory, in Spain.

Both of these stations are taken into consideration for the communication's design, since they will both be used. Nevertheless, since NanoSat Lab has full access to Montsec's Ground station, this is the one that will be used in order to compute the link budget.

3.2.2.1 Montsec Ground Station

The UPC NanoSat Lab Ground Station is the main commanding station for its satellites. It has been developed and built by NanoSat Lab students, and is currently being managed together with IEEC. It is equipped for UHF, VHF and S-Band communications. The station has Yagi antennas for VHF reception and UHF transmission and reception. Moreover, it has a 3-meter diameter dish S-Band antenna, shown in Figure 3.3. This antenna has an azimuth and elevation controller. The RF chain is currently composed by a Helical feed, a Bidirectional Amplifier, and an Analog Devices Pluto SDR as the ADC/DAC, with an automated software for scheduling and the retrieval of data through an optical fiber connection to the Barcelona Operation Center. A REST API is used to interface with the center in order to request passes and to download retrieved data.

Figure 3.3: Montsec Ground station S-Band Antenna

The location of the facility as well as the characteristics of the antenna are further explained in Table 3.4.

Table 3.4: Montsec Ground Station S-Band RF Chain Parameters [42]

Parameter	**Value**	**Unit**
Latitude	42.0514	deg
Longitude	0.7294	deg
Altitude	1.570	km
Height above the ground	6	km
Elevation angle	0	deg
Transmission frequencies	2025-2110	MHz
Reception frequencies	2200-2290	MHz
Diameter	3	m
Gain	35	dBi
Antenna-gain-to-noise-temperature	9	dB
Polarization	RHCP	-

Figure 3.4: Montsec Ground station [43]

3.2.2.2 *NSSTC Ground Station*

The NSSTC Ground Station is located within its campus. Its parameters are depicted in Table 3.5.

Table 3.5: NSSTC Ground Station Parameters

Parameter	Value	Unit
Latitude	24.1961	deg
Longitude	55.6817	deg
Altitude	0.2127	km
Height above the ground	0	km
Elevation angle	10	deg

3.3 Introduction to Orbits

Having gathered information regarding the frequency bands, as well as the satellite antenna and the ground stations, the next step towards fully understanding the scenario is to study how the satellite's trajectory affects signal transmission, and how can this effect be modelled and compensated. Therefore, an understanding of orbits is necessary.

An orbit is a regular elliptical path that a satellite follows in space around another satellite. The altitude of the orbit, being the distance between the Earth's surface and the satellite, determines the velocity of the satellite. The higher the orbit, the lower the gravitational pull and the lower the velocity of the satellite. Following the same reasoning, the lower the orbit, the higher the velocity. Orbits can be categorized in different manners. According to their height from Earth, they can be categorized into three types which are: High Earth orbit (HEO) , Medium Earth Orbit (MEO) and Low Earth Orbit (LEO).

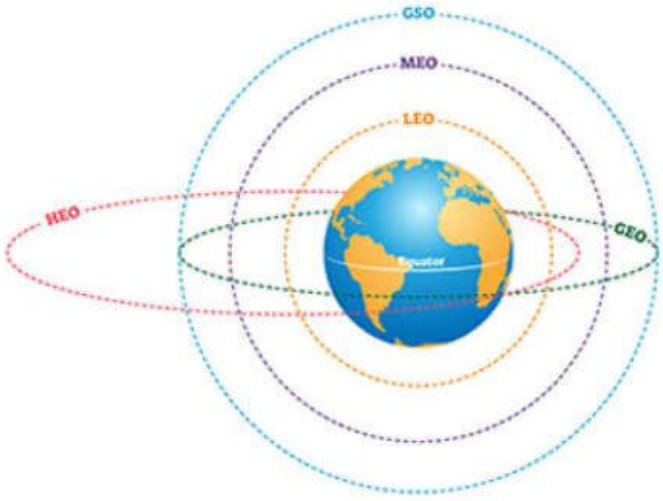


Figure 3.5: Orbit types [44]

3.3.1 High Earth Orbit

HEO includes all orbits that are in an altitude equal or higher to 42,164 km from the Earth's center or 36,000 km from its surface. Satellites orbiting at precisely this height match the Earth's rotation speed. This means that these satellites maintain the same view of Earth. This orbit is known as geosynchronous.

Geosynchronous satellites orbit over the equator, and therefore, have a null inclination. This way, they keep still relative to the ground. This characteristic is extremely valuable for some purposes such as weather monitoring and communications. Moreover some geostationary satellites have been used to monitor solar activity.

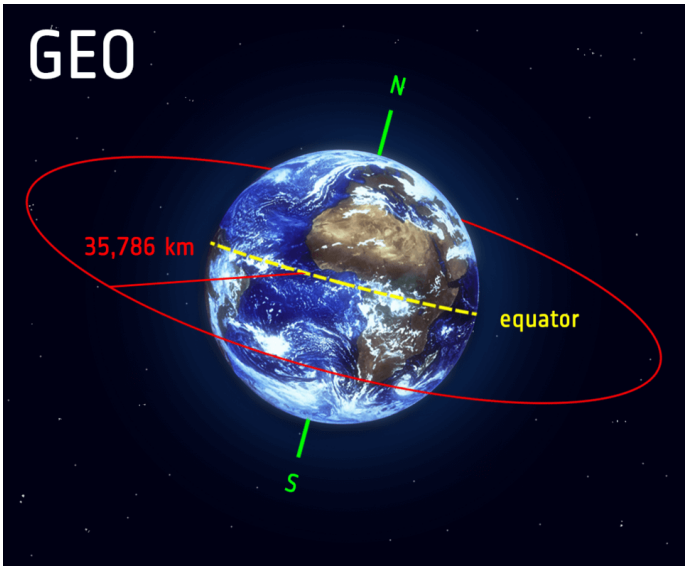


Figure 3.6: Geostationary orbit [45]

It should be noted that beyond HEO, there are other orbital important spots, known as the Lagrange points. At these, the Earth's gravity pull is equal to the Sun's gravity pull, therefore any objects located at these points orbit with the Earth around the Sun.

3.3.2 Medium Earth Orbit

MEO orbits are located approximately in altitudes from 10,000 to 20,000 km. There are two notable orbits, these are: the semi-synchronous orbit and the Molniya orbit.

The semi-synchronous orbit rotates at about 20,200 km above the Earth's surface with a nearly circular ellipse. Satellites following this orbit present a 12 hour period. Therefore, a satellite crosses the exact same point on Earth twice in a day. This orbit is mainly used by the Global Positioning System (GPS) satellites.

The Molniya orbit, opposed to the previous, is characterized by an extremely eccentric orbit. Also with a 12 hour period, satellites following this orbit will spend two thirds of their period over one hemisphere, and will quickly pass through the others. These orbits are most useful for communication purposes in north and south regions. More information can be found in [46].

3.3.3 Low Earth Orbit

Low Earth Orbits have relatively small altitudes, commonly smaller than a 1000 km but can be as low as 160 km [45]. These are the fastest orbits and have very small periods.

LEO orbits, contrary to GEO orbits, do not follow a specific path. Depending on the needs of the satellite, the plane of the orbit can be tilted appropriately. This characteristic allows for LEO satellites to follow a great variety of routes. The chosen inclination depends what areas are to be monitored. The lower the inclination, the closer the satellite will stay to the equator, and the higher, the closer to the poles it will pass through.

Moreover, it's close proximity to the Earth offers great advantages. In fact, this orbit is commonly used for satellite imaging, since being the closest to the Earth's surface, they allow for better resolution. Moreover, it also has a smaller propagation delay than the other orbits.

Nevertheless, LEO orbits also present some disadvantages with respect to the others. Firstly, being the easiest orbit to access, a great amount of satellites have already been deployed in this orbit. Secondly, the high velocity of the satellites in orbit also presents some disadvantages. Because of the great velocities, these satellites present very high Doppler effects that must be compensated for their correct functioning. Moreover, the high speed also results in the satellite having small periods of visibility, complicating communications with ground stations on Earth. For this reason, communication satellites in LEO orbits work within large constellations to have constant coverage. Perfect examples of these are Starlink and Oneweb constellation among others.

3.3.3.1 Sun-synchronous orbit

Sun-synchronous orbits (SSO), also known as heliosynchronous orbits, are characterized by the fact that, as their name indicates, they are synchronous to the sun. This means that the satellites following these orbits are synchronized to always maintain the same "fixed" position relative to the Sun.

This orbits are achieved by oscillating the orbital plane precess approximately one degree east each day with respect to the celestial sphere in order to keep the pace with the Earth's movement around the sun. The angular precession per orbit on Earth is described with Equation 3.1:

$$\Delta\Omega = -3\pi\frac{J_2 R_E^2}{p^2}\cos i \tag{3.1}$$

where J_2 is the coefficient for the second zonal term that equals 1.0826310^{-3}, R_E is the Earth's radius, p represents the semi-latus of the orbit and i is the orbit's inclination with respect to the equator. An orbit is sun-synchronous when the precession rate ρ is equal to the mean motion of the Earth around the Sun.

$$\rho = \frac{d\Omega}{dt} = \frac{\Omega}{T} = \frac{\Omega}{2\pi}\sqrt{\frac{\mu}{a^3}} \tag{3.2}$$

This orbits are commonly used at 600 to 800 km of altitude with inclinations similar to 98º.

The most typical uses of these orbits are imaging and weather satellites amongst others, since the illumination that the satellite experiences remains constant. This is crucial for remote sensing satellites, such as AlainSat-1, that requires sunlight to perform acquisitions with the Hyperspectral camera. Moreover, these orbits allow the satellite to cover a large area of the Earth, having from 16 to 13 orbits per day. Figure 3.7 depicts the path difference between consecutive sun-synchronous orbits.

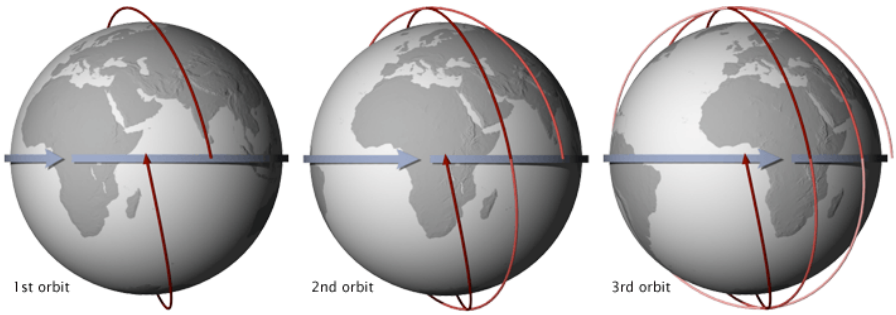


Figure 3.7: Sun-synchronous orbit [47]

3.4 AlainSat-1 Orbit

Having an understanding of orbits and of which is the best suited for the mission, the AlainSat-1 orbit must be studied. It should be noted that CubeSats are launched in the remaining space of a main payload, that is the main objective of the launch. This means that the orbit of the launch depends on the main payload's necessities and preferences over those of the CubeSats. Once the main payload of the launch is deployed, the remaining payloads are injected into a similar orbit. Therefore, it should be noted that the CubeSats' orbital parameters might slightly differ from the previous orbit parameters.

This satellite will follow a SSO orbit. The main orbital parameters that the AlainSat-1 satellite is expected to follow are given in Table 3.6.

Table 3.6: Tentative AlainSat-1 Satellite Orbit parameters

Orbit type	Syn-Synchronous (SSO)
Altitude	550 km
Inclination	97.6 deg
Orbital Period	95.55 min

Several simulations of the satellite's orbit have been made to not only study its effect on the Earth Observation instruments, but also to properly determine the communications' scenario. These are useful to demonstrate the path that the satellite will follow. Moreover, they also allow to compute the line of sight time that the S-Band Communication system will have to transmit the data, which is a key parameter of the communications link.

As it can be seen in Table 3.6, these parameters are an approximation to the orbit that will be followed. Many necessary values such as the eccentricity of the ellipse that are required for a correct simulation are missing. Therefore, in order to obtain a good estimate

of the orbit, the TLE sets from a previous satellite with a similar orbit as the one AlainSat-1 will be using, can be used.

In order to find an appropriate SSO orbit, website [48], containing the main orbital parameters of satellites in a SSO orbit has been used. From it, SAUDISAT 5B has been chosen for having the closest orbital parameters to those of AlainSat-1. The following simulations will use the orbital parameters depicted in Table 3.7.

Table 3.7: SAUDISAT 5B Orbit parameters [49]

NORAD ID	43833
Int'l Code	2018-102C
Perigee	534.0 km
Apogee	557.4 km
Inclination	97.5 deg
Period	95.4 minutes
Semi major axis	6916 km
Launch date	December 7, 2018
Source	Saudi Arabia (SAUD)
Launch site	Jiuquan Satellite Launch Center, China (JSC)

Moreover, [49] also provides the Two Line Element (TLE) Set of the satellite's orbit. This was retrieved on the 24th of March 2022 at 9:30h UTC, and is as follows:

1 43833U 18102C 22083.19254038 .00001003 00000-0 66815-4 0 9994

2 43833 97.5038 159.1481 0016825 115.0107 329.3400 15.09229590181363

The TLE is a format in which orbital elements of a satellite at a given point in time (epoch) are encoded. This is provides extremely important information about the exact parameters of the orbit in order to simulate its propagation in time. Appendix B describes how it can be interpreted and decoded.

The given TLE can be decoded and used to obtain the orbital parameters. These are shown in Table 3.8.

Table 3.8: Data extracted from TLE necessary for the simulation

Parameters	**Value**	**Units**
Orbit inclination	97.5038	deg
Longitude of the Ascending Node at the ToA	159.1481	deg
Orbital eccentricity	0.0016825	
Argument of the perigee at ToA	115.0107	deg
Mean Anomaly at ToA	329.3400	deg
Mean motion	15.09229590181363	rev/day

3.4.1 Matlab Orbital Simulation

Having the orbit's parameters, the satellite's trajectory can be estimated using Matlab. It should be noted that to get an accurate simulation of the trajectory over a long period of time, the TLE of the satellite should be constantly renewed since they require constant corrections. Nevertheless, for the purposes of this thesis, only using one TLE is sufficient.

Figure 3.8 depicts the orbit that the satellite would follow for 8 periods of time, starting at the point marked by the green dot. This 2D plot is very useful to see easily how the ground is covered by the satellite.

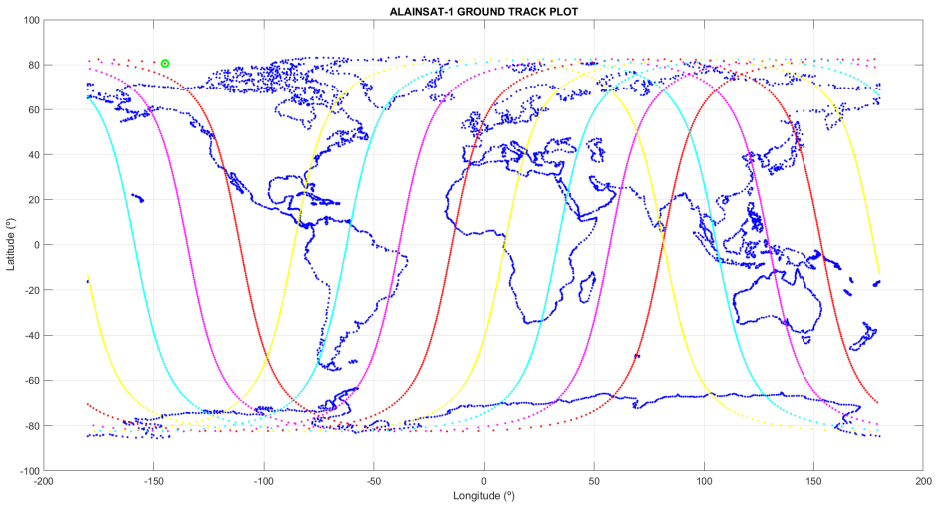
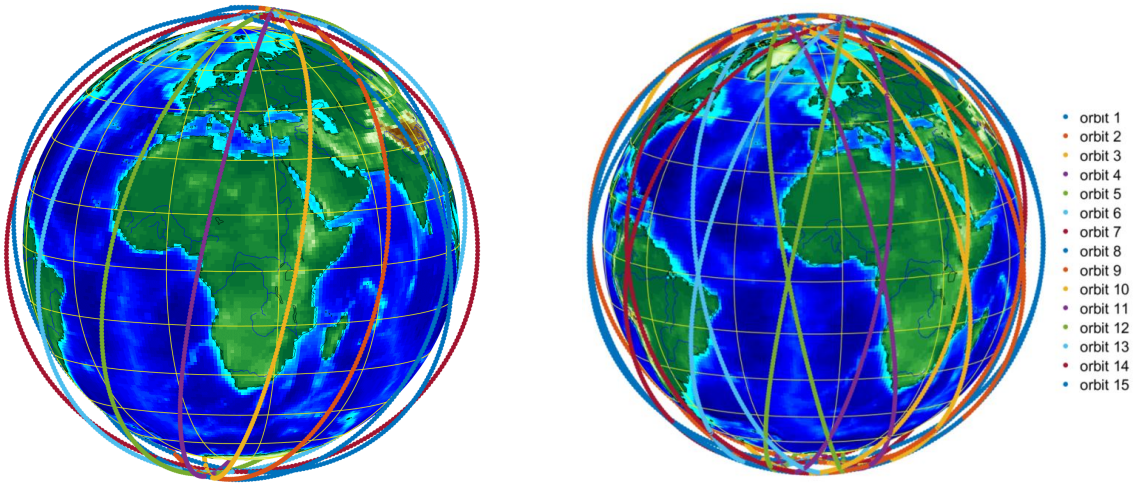


Figure 3.8: AlainSat-1 ground track plot for 8 periods

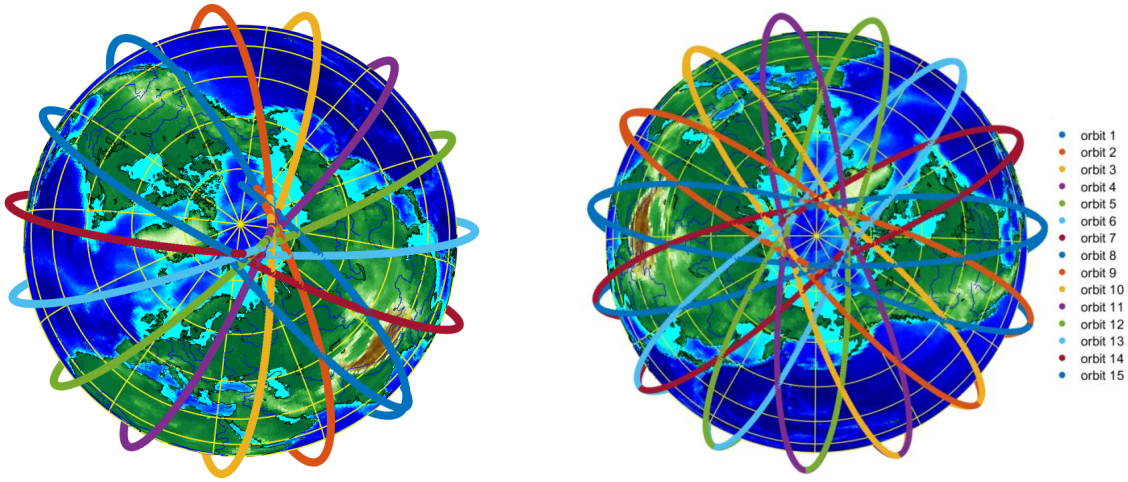
As it can be seen and has been explained earlier, this orbit will allow the satellite to cover most of the Earth and all of RITA's target areas.

Another useful way of plotting the orbits is in 3D. Figure 3.9, contrary to the previous plot, is particularly useful to note the orbit's inclination (97.5º). Moreover, this can also be noted in Figure 3.10, where the intersection of the orbits is slightly displaced from the Earth's pole. This is particularly noticeable in 3.10(b), where it is clear than none of the orbits of the satellite go through the center of the pole, but always maintain their inclination.



(a) 8 orbits (b) 15 orbits

Figure 3.9: AlainSat-1 3D plot centred in the equator



(a) 8 orbits (b) 15 orbits

Figure 3.10: AlainSat-1 3D plot centred in the pole

3.4.2 STK Orbital Simulation

The Systems Tool Kit (STK) [50] is a digital application that is used for mission engineering in the aerospace, defense and telecommunications fields amongst others. This is a key application for space orbital simulation, since it that allows to model accurately and analyze payloads in an environment in a realistic mission context.

This tool has allowed to carry out two types of simulations that are needed for the appropriate design of the communications system:

- Orbit simulation: Provides a more visual and accurate representation of the path followed by the satellite also taking into consideration the Sun.

- Line-Of-Sight simulation: Provides an estimate of the time that the satellite will have LOS with both ground stations.

3.4.2.1 Orbit simulation

This simulation is very useful to visualize the Sun-Synchronous path of AlainSat-1. The previous Matlab simulation provided information regarding it's estimated path, while this simulation complements that with images to confirm how the satellite moves relative to the sun and maintains the same.

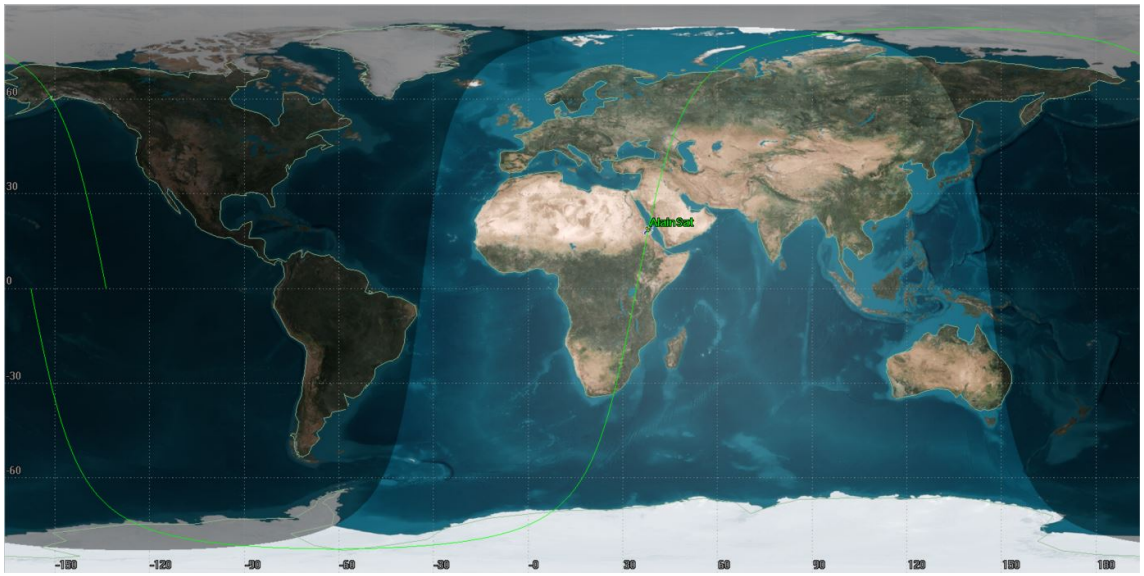


Figure 3.11: STK AlainSat-1 ground track plot simulation with sunlight

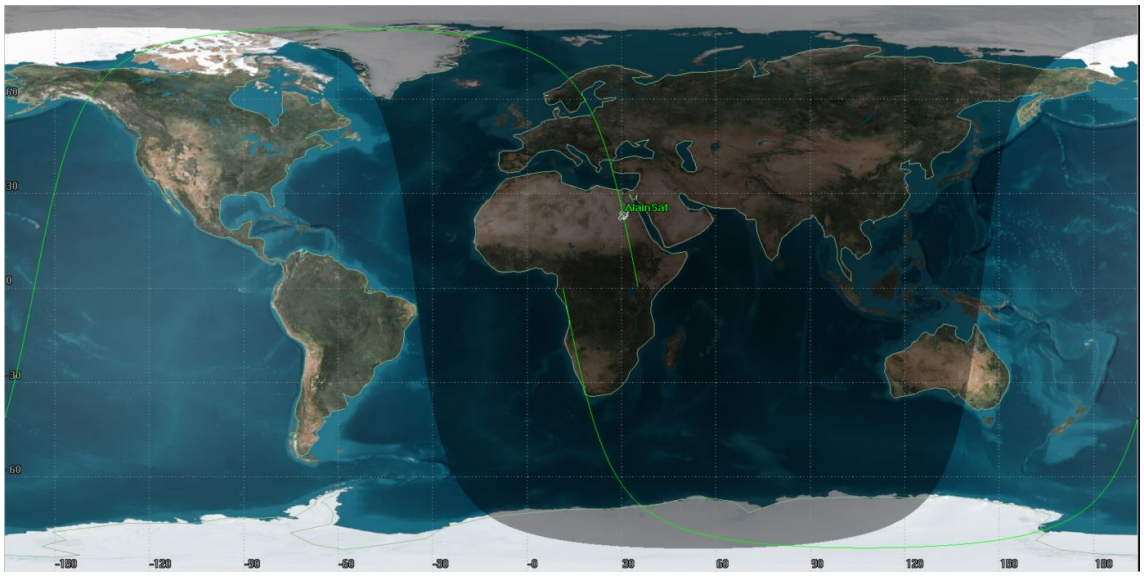


Figure 3.12: STK AlainSat-1 ground track plot simulation without sunlight

As it can be seen in the previous images, with sunlight (Figure 3.11) and at dark (Figure 3.12), the satellite maintains the same local mean time at all points of its path, as expected with a SSO.

Figure 3.13 is very useful to visualize the 3D path of the LEO sun-synchronous orbit, depicting its very high inclination. Moreover, just like the previous figure, it is also useful to be able to check how the satellite and the sun.

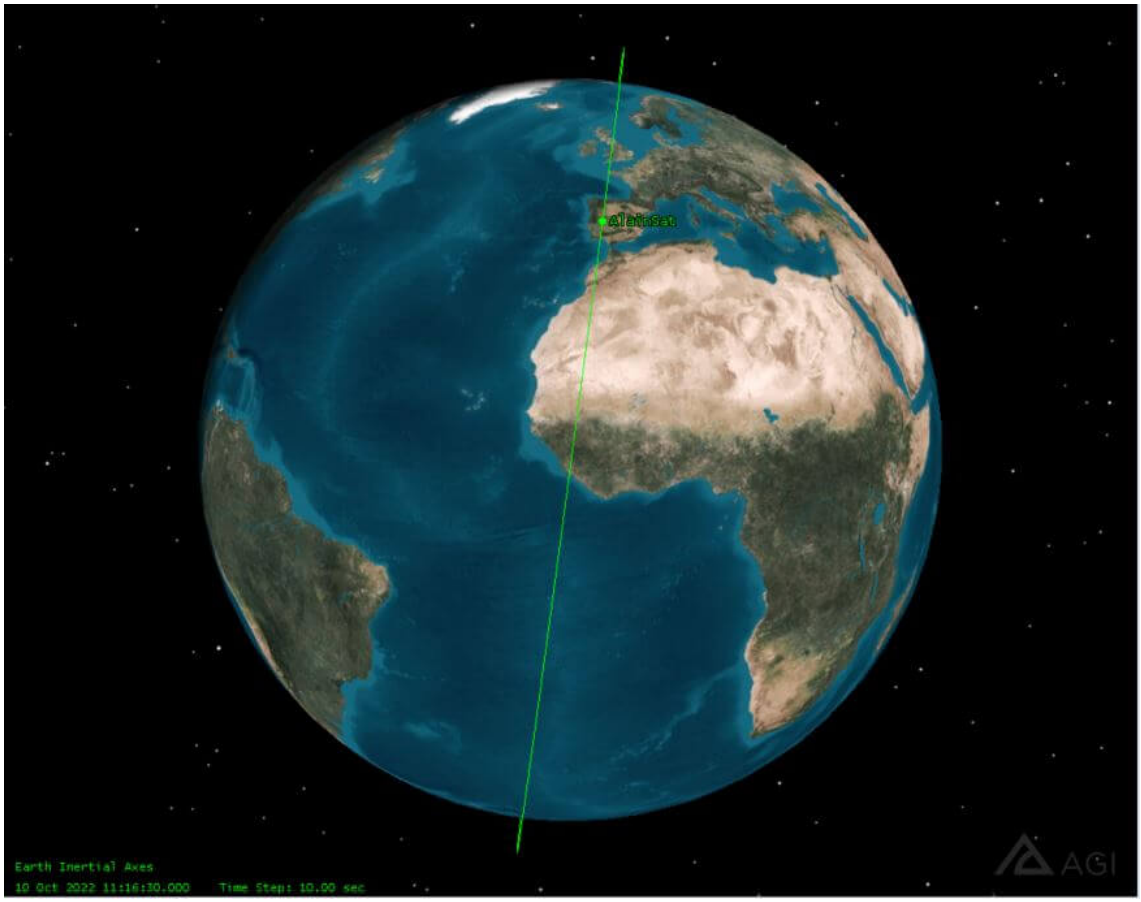


Figure 3.13: STK AlainSat-1 3D orbit simulation

3.4.2.2 S-Band LOS simulation

This simulation intends to compute the coverage time that the satellite will have with the ground stations per pass.

In order to carry out this simulation, the ground stations of Montsec and NSSTC are inserted in the program. These are represented in Figure 3.14.

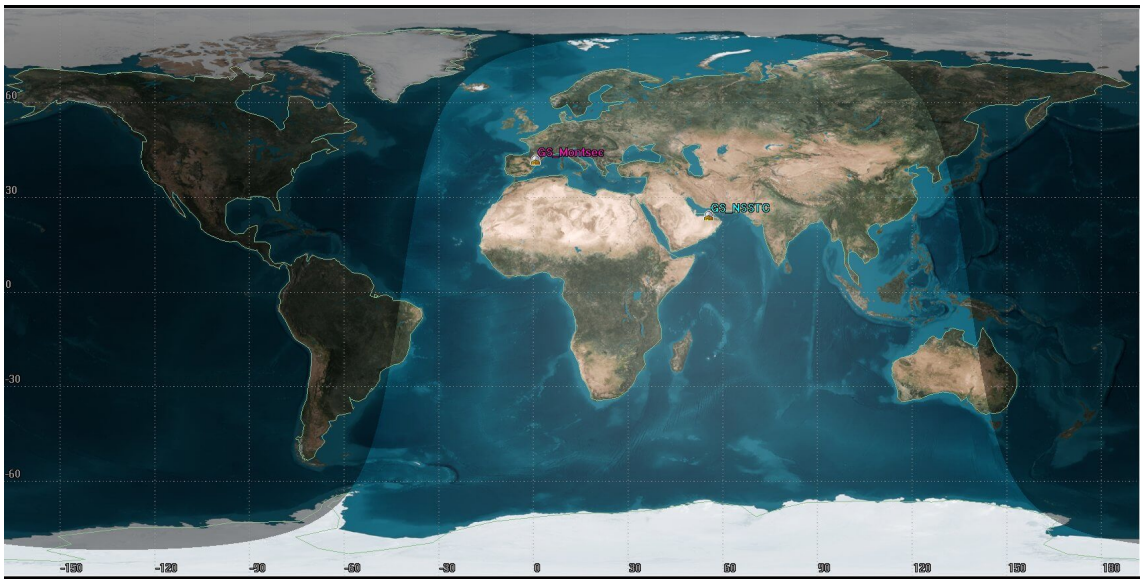


Figure 3.14: STK Ground Stations

Having inserted the ground stations and the satellite orbit's parameters, the coverage time can be obtained for each satellite. A whole week has been simulated in order to obtain more representative "Access Summary Reports". These have indicated that the mean duration of each access is 8.33 minutes for Montsec ground station and of 8.76 for NSSTC ground station. Nevertheless, in order to be conservative, the thesis will consider an average time of 7 minutes for its design.

Figures 3.15 and 3.16 show the link between the satellite and Montsec ground station in the first case and NSSTC ground station in the second.

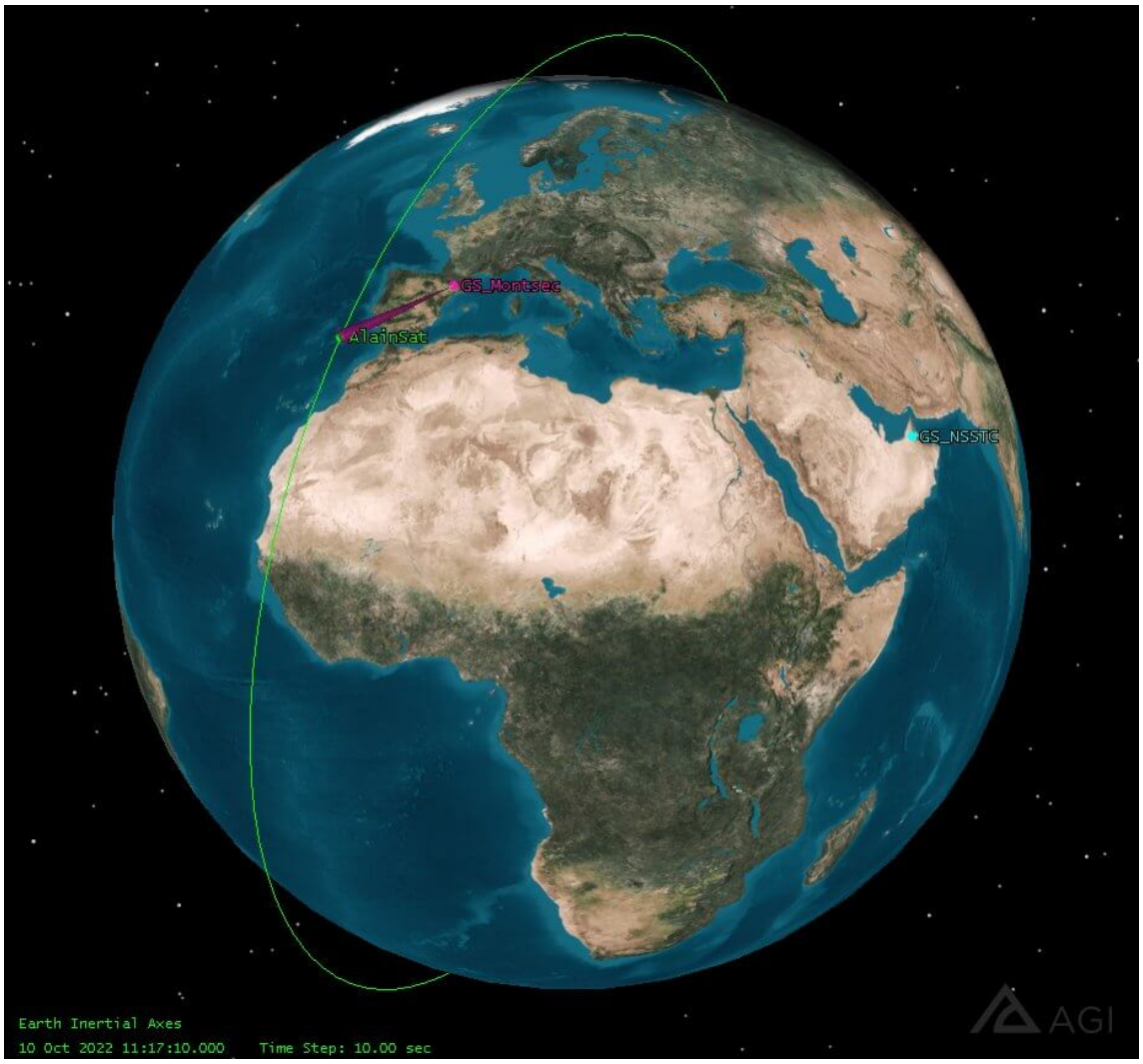


Figure 3.15: LOS between AlainSat-1 and Montsec Ground Station

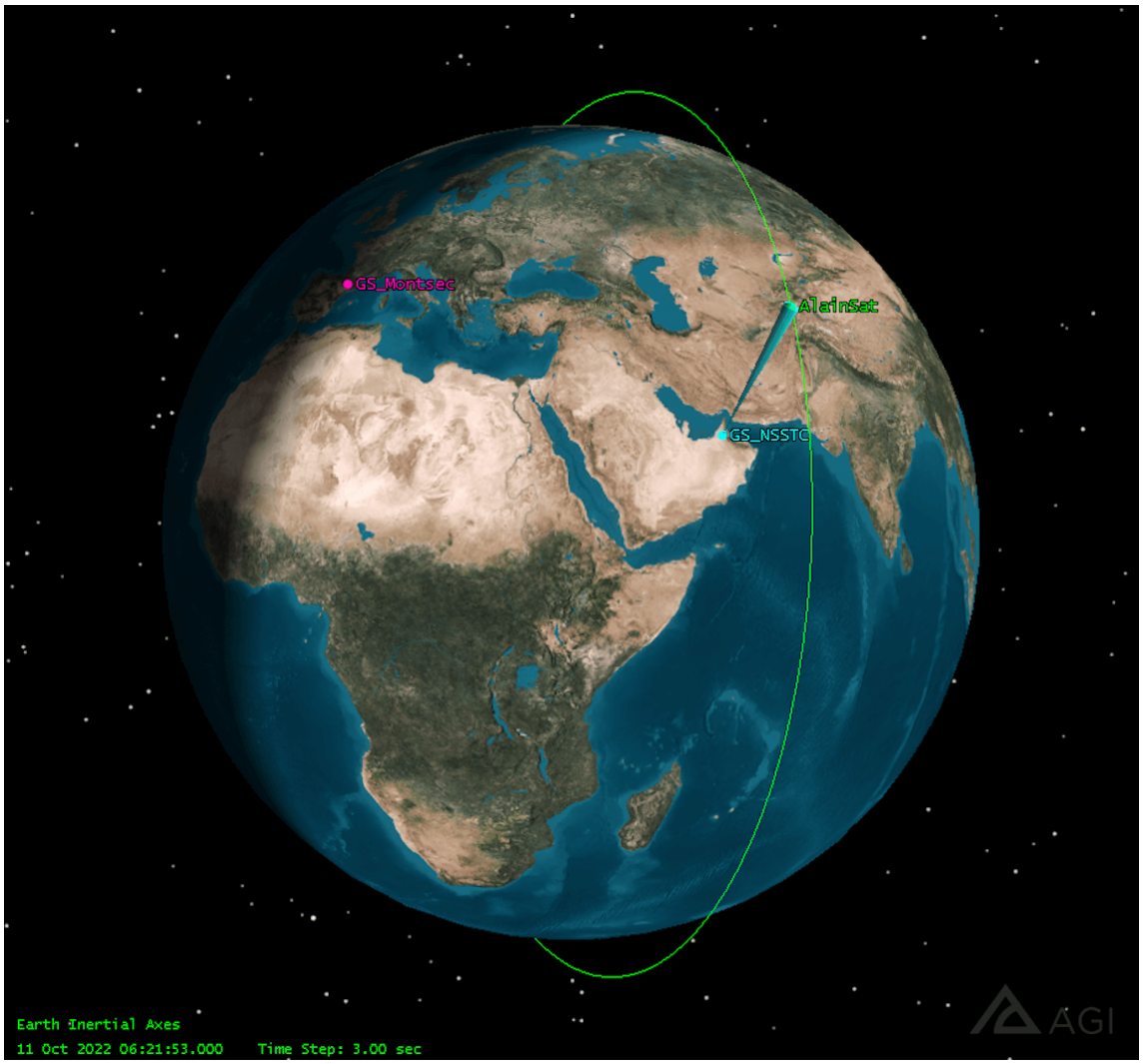


Figure 3.16: LOS between AlainSat-1 and NSSTC Ground Station

CHAPTER 4. S-BAND LINK BUDGET

The link budget is an account of the parameters that define a communication link for a reliable connection between transmitter and receiver [51]. This section will first detail the parameters that are involved in the communication link and then carry out its computation.

The communication scenario will consider both downlink and uplink communications. Nevertheless, this chapter will focus on the downlink computation since it is the most relevant scenario for the mission, and the one that will transmit the scientific data generated within the RITA payload.

4.1 Link Budget Analysis

Figure 4.1 depicts an schematic of the communication scenario to be studied.

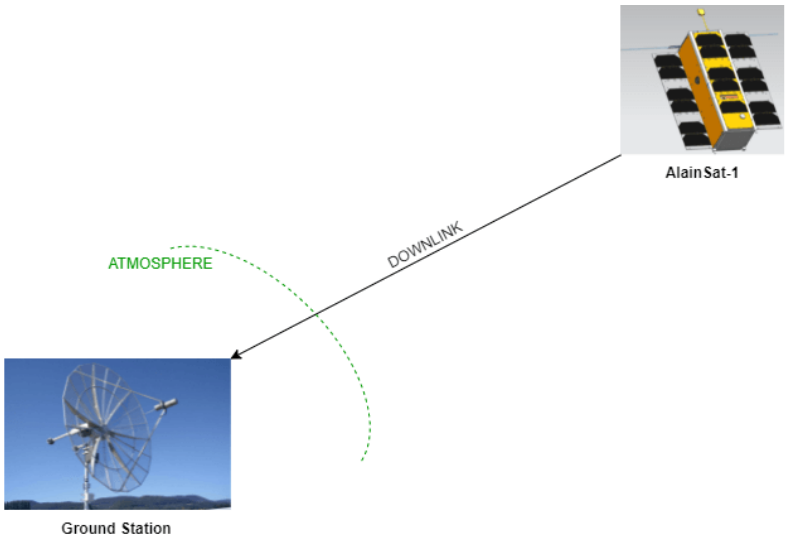


Figure 4.1: Downlink scenario schematic

The received power (P_r) by the ground station can be computed with the following equation:

$$P_r[dBm] = P_t[dBm] + G_t - L_t - FM \tag{4.1}$$

where P_t is the transmitted power by the transmitting antenna in dBm, G_t is the gain of the transmitter in dB, L_t are the total losses of the system in dB and FM is the link margin desired for the system in dB.

4.1.1 Gain

An isotropic antenna is an ideal antenna that radiates equally in all directions. Its power intensity can be described as:

$$U(\theta,\phi) = \frac{P_t}{4\pi}[W/sr] \tag{4.2}$$

The Gain of directive antennas can be computed with the following equation.

$$G(\theta,\phi) = \frac{U(\theta,\phi)}{\frac{P_t}{4\pi}} \tag{4.3}$$

The gain of a receiving parabolic dish antenna can be computed with the following equation through the reciprocity theorem.

$$G_R = \frac{4\pi}{\lambda^2}A_{eff} \tag{4.4}$$

where A_{eff} is the effective area of the antenna and can be computed as:

$$A_{eff} = A_{phy}\eta \tag{4.5}$$

Moreover, since it is a parabolic antenna, its area can be computed with Equation 4.6.

$$A_{phy} = \pi R^2 = \pi\frac{\pi D^2}{4} \tag{4.6}$$

4.1.2 Losses

The power of radio signals experiences a considerable decrease over distance, this is due to losses. In the following sections the losses experienced by the signal in a transmission satellite to ground station will be explained.

The total losses can be computed with equation 4.7.

$$L(dB) = L_p + L_a + L_{pol} + L_{aml} \tag{4.7}$$

4.1.2.1 Free Space Path Losses

Free Space Path Losses occur when a signal travels through space without having any other effect under consideration.

These losses can be computed with the following equation, where d stands for distance in meters, f stands for frequency in Hz and c for the speed of light in m/s.

$$L_p(dB) = 10\log\left(\frac{4\pi df}{c}\right)^2 \tag{4.8}$$

It should be noted that, since the link is between a ground station and a satellite in a LEO orbit, the distance is not be constant and changes with the elevation angle of the satellite. This is expressed with equation 4.9.

$$d = \sqrt{(R_E + h)^2 - R_E^2 \cos\phi^2} - R_E \sin\phi \tag{4.9}$$

where ϕ represents the elevation angle and θ, that represents the angle between the position of the CubeSat and the ground station. Also, h stands for the altitude from the surface of the Earth, and R_E represents the Earth radius. This can be seen in 4.2.

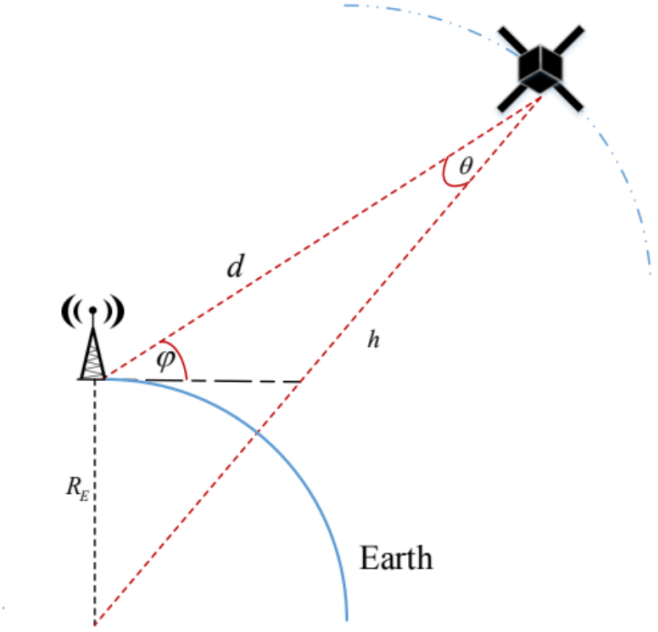


Figure 4.2: CubeSat trajectory schematic [51]

4.1.2.2 Polarization losses

The polarization of an electromagnetic wave can be defined as the orientation of its electric field vector. It is common that the polarization of the receiving antenna differs from that of the incident wave. This is known as polarization mismatch, and the electromagnetic power that is lost due to it is characterized by the polarization loss factor (PFL). The PFL will be 1 when there is no mismatch [52].

For linear polarization, losses can be expressed as a function of the angles between the polarization vectors at the transmitting and receiving antennas. However, this link will work with circular polarization.

For this particular link budget, the used polarization will be circular, in particular a Right Hand Circular Polarization (RHCP). In order for the link to work correctly, the polarization of the transmitting antenna must be equal to that of the receiving antenna. This is the case, since Montsec parabolic S-Band antenna has a RHCP polarization with an LHCP feed (this is because when the RHCP signal bounces back from the parabolic it converts into LHCP), the link should work properly and only errors in the polarization and efficiencies should affect the losses.

4.1.2.3 Pointing losses

In an ideal link, the transmitter and receiver are expected to be perfectly aligned. Nevertheless, when put into practice, the alignment of transmitter and receiver antennas will always present some error and therefore lead to some losses in the transmission.

Pointing losses for the parabolic antenna at reception can be computed as [53] :

$$L_{pointing}(\theta) = 12\left(\frac{\theta}{\theta_{3dB}}\right)^2 \tag{4.10}$$

where θ is the deviation and θ_{3dB} is the half power beamwidth of the antenna in degrees, which can be determined with equation 4.11.

$$\theta_{3dB} = 70\left(\frac{\lambda}{D}\right) \tag{4.11}$$

These equations show how the bigger the directivity of the antenna, the larger the pointing losses that it will have, and as expected, the larger the deviation, the bigger the losses.

Since the communication is between two antennas, the pointing losses of both must be accounted for.

On the one hand, pointing losses in the ground station will depend on the pointing capabilities of the antenna as well as the tracking of the satellite.

Tracking the satellite consists of being able to determine its position at every time instant. Even if the parameters of the satellite's orbit are known prior to the launch, these will change depending on the launch itself and the ejection of the CubeSat. However, the satellite's position can be tracked with radars that can compute its TLE. For instance, SPACECOM, a combatant command of the USA's Department of Defense has a military branch, USSF, that tracks objects orbiting Earth and makes all non-confidential information public. These TLE's can be accessed in CelesTrak website, available in [54]. These present a very high accuracy, with typical position error lower to one kilometer in LEO orbits [55], nevertheless errors up to 10 km have been reported. For the LEO orbit that AlainSat-1 will follow, this can be translated into approximately a 0.083 degrees deviation.

Moreover, when it comes to the tracking capabilities of the antenna, these are approximately 0.5 degrees due to the mechanisms and the software used.

Having determined these values, in order to compute the pointing losses, the beamwidth of the antenna must be computed. By using equation 4.11, and knowing its parameters, shown in Table 3.4, the beamwidth obtained is of 3.07 degrees.

Taking into consideration the total pointing errors introduced and the antenna beamwidth, equation 4.10 determines that the total pointing losses will be of 0.43 dB. However, it should be noted that this approximation does not account for many factors that might affect the pointing accuracy such as structure vibrations in the receiving antenna amongst others.

On the other hand, pointing losses in the AlainSat-1 must also be considered. The AlainSat-1 is built for Earth Observation it has a very accurate attitude control for functions such as the use of the hyper-spectral camera. Therefore, the pointing losses should be negligible in comparison with those in the ground station. Nevertheless, considering the scenario

and conditions in which the satellite has been developed, some pointing errors are to be expected.

Finally, in order to account for the worst case scenario, the link consider pointing losses of 3 dB, which is a commonly used value, and will ensure a more reliable link.

4.1.2.4 Atmospheric attenuation

Atmospheric transmittance is the capacity that the atmosphere has to transmit electromagnetic energy. This value is dependent on the wavelength of the energy, as it can be seen in Figure 4.3. Moreover, this value also depends on the dust present in the air, the water vapor as well as the air mass.

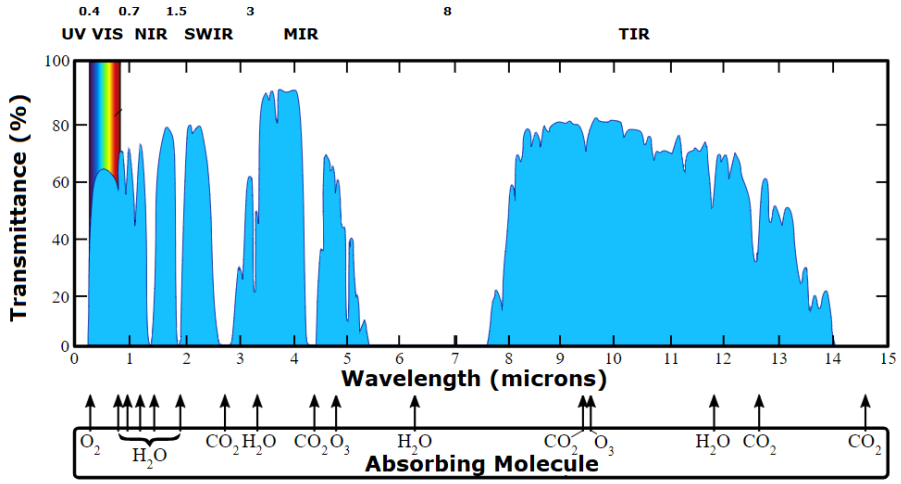


Figure 4.3: Atmospheric transmittance [56].

For the purposes of the link budget, it can be considered that the main attenuation of the signal caused by the atmospheric transmittance is due to aerosol and molecular particles that are present in the atmosphere and cause scattering and absorption of the irradiance of the signal. This factor can be obtained by using SpectralCalc [57] or programs such as MODTRAN [58].

The formula that represents this factor is dependent on the elevation and azimuth's angle of the link and is the following:

$$L_a = L_{a,azimuth}^{\left(\frac{1}{\cos\zeta}\right)} \tag{4.12}$$

As Figure 4.4 shows, the higher the frequency, the bigger the effect that rain, fog and drizzle have on communication links. The S-Band frequency band is not affected relevantly by these factors and therefore can be dismissed for the link budget computation.

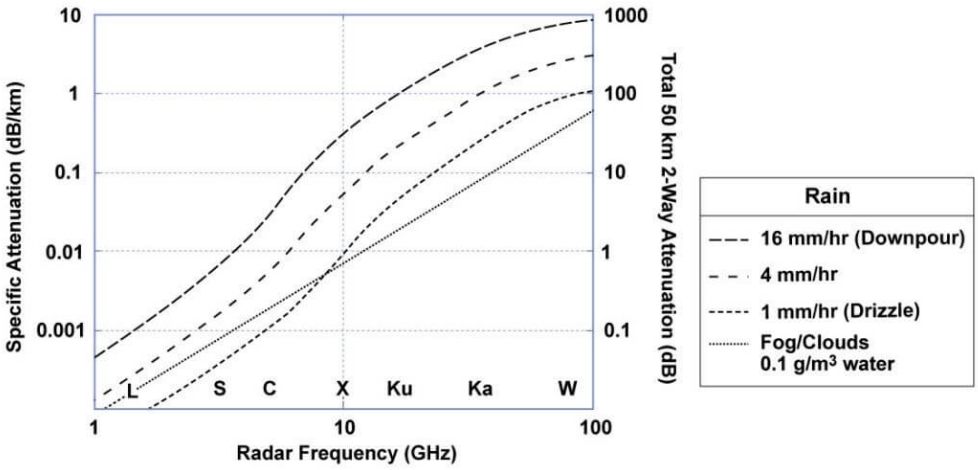


Figure 4.4: Atmospheric attenuation [59]

4.1.3 Fading Margin

The fading margin allows the link to accommodate unexpected fadings that might occur. This budget will consider a standard value of 5 dB.

4.1.4 Noise

The thermal noise, also known as white noise, is the electronic noise generated by the agitation of the electrons inside an electrical conductor. This is distributed throughout the band, and is proportional to the bandwidth and temperature of the system:

$$P_n = kT_{eq}B \tag{4.13}$$

where K is the Boltzmann constant $(1.3910^{-23}JK^{-1})$, T_{eq} is the equivalent noise temperature of the receiver system in kelvins and B is the bandwidth in Hz.

4.1.5 Signal to Noise Ratio

The Signal-to-noise ratio is a measure that compares the level of the desired signal power to that of the background noise. This value is necessary to obtain the link characteristics. It can be computed with the following equation:

$$SNR[dB] = P_r - P_n \tag{4.14}$$

where P_r is the power of the signal received in the satellite and P_n is the thermal noise power.

4.1.6 Eb/No

An indicator of the quality of the link is the energy-per-bit to noise spectral density. At the ground station, this can be computed with the following equation:

$$\frac{E_b}{N_o} = \frac{P_t G_t G_r}{LkTR_b} \tag{4.15}$$

where P_t stands for the transmitted power by the satellite in Watts, G_t stands for the gain of the transmitter and G_r for that of the receiver. Moreover, T represents the system temperature noise, k the Boltzmann constant, L the total losses of the link and R_b the data rate.

The manner of computing the required Eb/No is through the desired modulation and the BER. In this case, the modulation used will be a BPSK (explained in detail in Chapter 5, section 5.5.2) and the accepted bit error rate will be of 1.00e-6. With these parameters and using Figure 5.15, a Eb/No of 11.3 can be obtained.

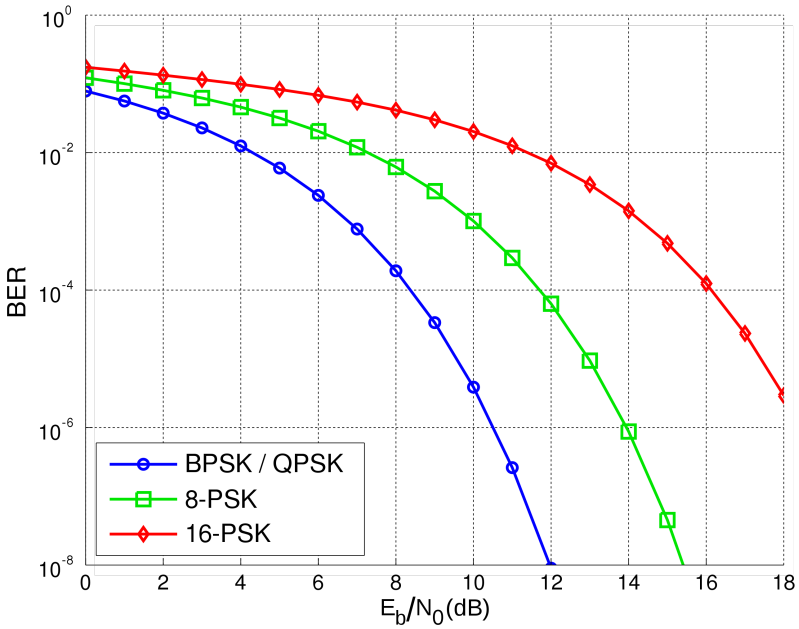


Figure 4.5: BER vs Eb/No [60]

Moreover, in order to prove that the approximation of the value from the graph has been done correctly, Equation 4.16 can be used. As it can be seen, the BER for this Eb/No value satisfies the condition.

$$p_b = 0.5erfc\left(\sqrt{\frac{E_b}{N_o}}\right) = 0.5erfc(\sqrt{11.3}) = 9.97423e-7 \tag{4.16}$$

Having obtained the Eb/No, the bit rate of the link can be computed through the following formula (4.17).

$$SNR[dB] = \frac{E_b}{N_o} + 10\log(f_b) - 10\log(B) \tag{4.17}$$

4.1.7 Doppler Effect

The Doppler Effect, also known as Doppler shift, is a phenomenon characterized by the change in the apparent frequency of a wave as a result of a relative motion between the source and the observer. When designing space communications it is important to consider the Doppler effect to account for it appropriately in the design of the system. Since the velocity of satellites following LEO orbits is considerably high, the effect will be considerable for an S-Band communications link.

The Doppler effect can be computed with equation 4.18.

$$f_{doppler} = \frac{V_r}{\lambda} = \frac{V_r f}{c} \tag{4.18}$$

Where V_r stands for the radial circular velocity of the satellite, λ stands for the wavelength of the signal, f stands for the frequency of the carrier signal and c for the speed of sound.

In order to compute the effect of the Doppler shift that the ground station will need to compensate, the communication scenario has to be studied. The Doppler shift will be zero when the satellite is at the zenith of the ground station, but will be maximized when first entering the line of sight as well as when leaving it. Figure 4.6 represents the scenario with the highest Doppler frequency shift (considering a 0º elevation angle in the ground station), which would be equivalent to having the satellite in the opposite side, for which the shift would have the same magnitude but opposite sign.

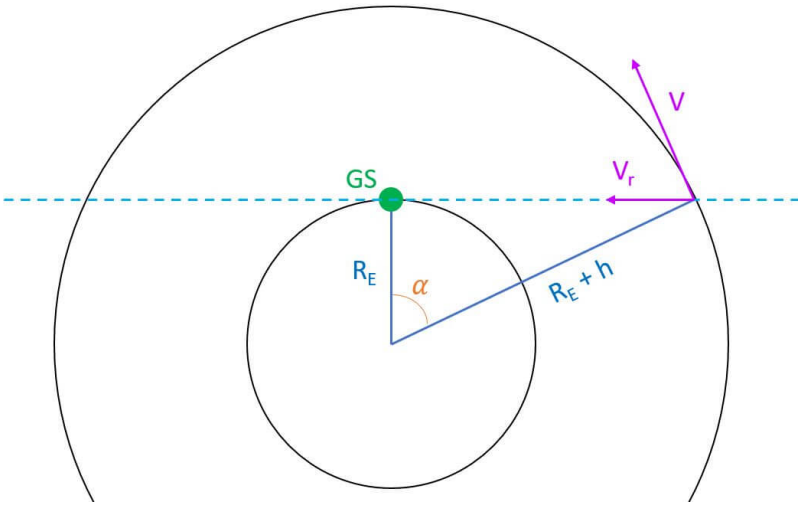


Figure 4.6: Doppler effect computation scenario

It should be noted that the following computations are carried out considering a circular LEO orbit of radius 550 km, that provides a very close approximation to the real result.

The basis of the computation is Kepler's equation (4.19), which allows obtaining the period of an orbit around the Earth from the major axis and the Earth's mass and gravity constants.

$$T = \sqrt{\frac{a^3}{GM}} \tag{4.19}$$

In order to compute the radial velocity of the satellite with respect to the ground station, the Earth's rotation must be considered. Knowing that $w = \frac{2\pi}{T}$, Equation 4.20 can be obtained.

$$w' = w_{Satellite} - w_{Earth} = \sqrt{\frac{GM}{a^3}} - \frac{2\pi}{T_{Earth}} \tag{4.20}$$

The radial velocity of the satellite can be computed with Equation 4.21.

$$V_r = V cos(\alpha) = V_r \frac{R_E}{R_E + h} \tag{4.21}$$

Considering that $v = wR$, the resulting Doppler shift computation can be expressed as 4.22 using the obtained equations 4.18 and 4.20.

$$f_{Doppler} = \left[\sqrt{\frac{GM}{(R_{Earth} + h)^3}} - \frac{2\pi}{T_{Earth}} \right] R_{Earth} \frac{f}{c} \tag{4.22}$$

Using the parameters depicted in Table 4.1, the maximum Doppler obtained for the satellite is $\pm$ 47.997 KHz.

Table 4.1: Doppler computation parameters

Parameters	Value	Units
G	$6.67428e-11$	$m^3kg^{-1}s^{-2}$
M	$5.972e24$	kg
R_{Earth}	6371	km
T_{Earth}	86344.1	s
h	550	km
f	2.279	GHz
c	$2.99792e8$	ms^{-1}

Having computed the maximum value, it should also be considered that since the satellite is constantly moving, this Doppler frequency changes. Figure 4.7 details the expected Doppler frequency depending on the position of the satellite. In addition, it should be noted that the frequency shift will be higher when the satellite is at zenith, and therefore, more difficult to track appropriately.

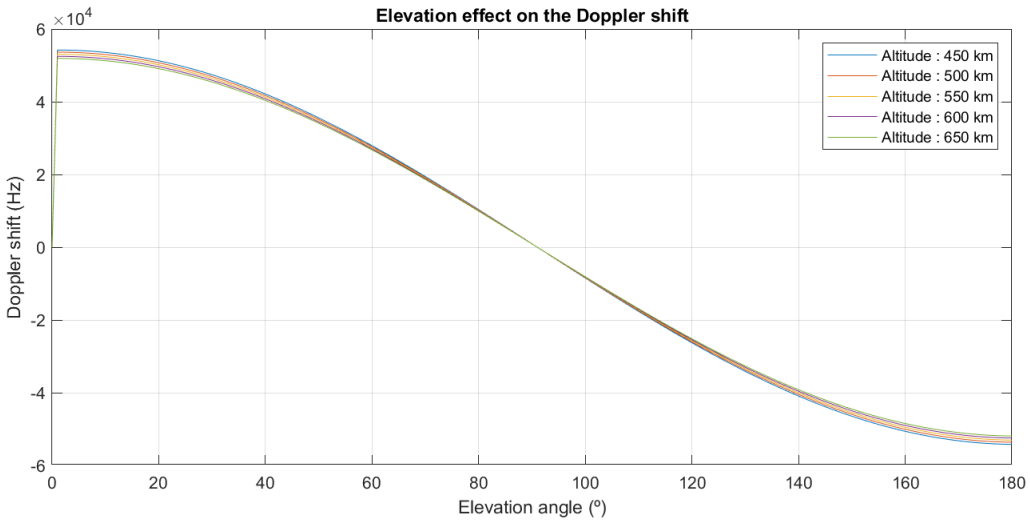


Figure 4.7: Doppler shift depending on the elevation angle

The maximum Doppler frequency shift is a very important value for some communication systems, as is this case, where the receiver of the communications, whether if its the satellite in the uplink or the ground station in the downlink, must be able to correct it. This is taken into consideration in the design of the code and detailed in Chapter 5, specifically in section 5.5.6 which details the signal tracking mechanisms.

4.2 Link Budget Computation

Having studied the parameters that have an effect on the link budget of the communication system, the downlink link budget is now developed further.

The system will have an uplink (transmission from the ground station to the satellite) and a downlink (transmission from the satellite the ground station). Nevertheless, the critical link is the downlink since the satellite has a limited power. Moreover, it has to transmit the highest amount of data.

It should be noted that the link budget has been studied for the worst case scenario between the satellite and the Montsec Ground Station, and therefore, a link distance of 2500 km has been used, which is equivalent to the horizon distance. By using the worst case scenario in the computations, the design is more robust and better prepared for working properly.

Table 4.2 depicts all of the necessary data for the link budget computation. The transmitter data has been extracted from the ANT2150 DUP datasheet, which can be accessed in appendix I.

Table 4.2: Downlink data for the link Budget

Parameters	Value	Units
Physical parameters		
Frequency	2279	MHz
Wavelength	0.1316	m
Link Distance	2500	km
Bandwidth	2.4	MHz
Transmitter (NanoCom ANT2150 DUP)		
Transmitted Power	32	dBm
Transmitter Antenna Gain	8.5	dB
Transmitter pointing losses	3	dB
Losses		
Fade Margin	5	dB
Polarization losses	0	dB
Antenna (Ground station)		
Receiver Antenna Gain	35	dB
Receiver LNA Gain	30	dB
Receiver LNA NF	0.4	dB
Antenna Temperature	10	ºC
Target Eb/N0	11.3	dB
RF Chain		
Cable length	16	dB
Cable losses	0.15	dB/m

It should be noted that the temperature for the satellite in LEO is expected to range from –65 ºC to +150 ºC [61], but a standard value of 10 ºC has been used for the computation.

4.2.1 Gain Computation

In order to compute the total gain of the link, all the gain components can be summed. In this case, the gain from the transmitter antenna, the receiver antenna and the LNA are considered, as indicated in Table 4.3.

Table 4.3: Link Budget gain computation

Parameters	Value	Units
TX Antenna Peak Gain	8.5	dB
RX Antenna Peak Gain	35	dB
LNA Gain	30	dB
Receiver Total Gain	73.5	dB

4.2.2 Losses Computation

As section 4.1.2.1 explains, the free space losses of the link are dependent on the elevation of the satellite. With equation 4.9, a plot of distance from the ground station to the satellite depending on the elevation angle can be computed in Figure 4.8. As it is to be expected, the distance is the smallest when the satellite overflies the ground station, and the biggest when it enters and leaves its line of sight, which can go up to approximately 2500 km.

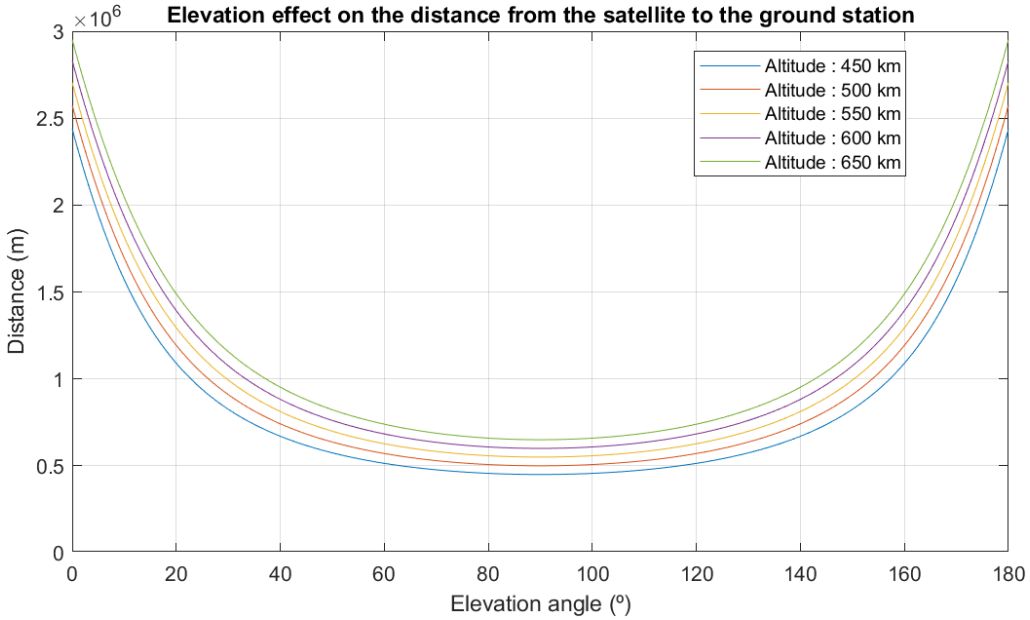


Figure 4.8: Distance vs elevation

Having computed the distance dependence, the free space losses can be computed depending on the elevation angle. This dependence is computed and depicted in Figure 4.9. As it can be seen, the smaller the distance, the smaller the losses, which can range from 153 dB to 169 dB.

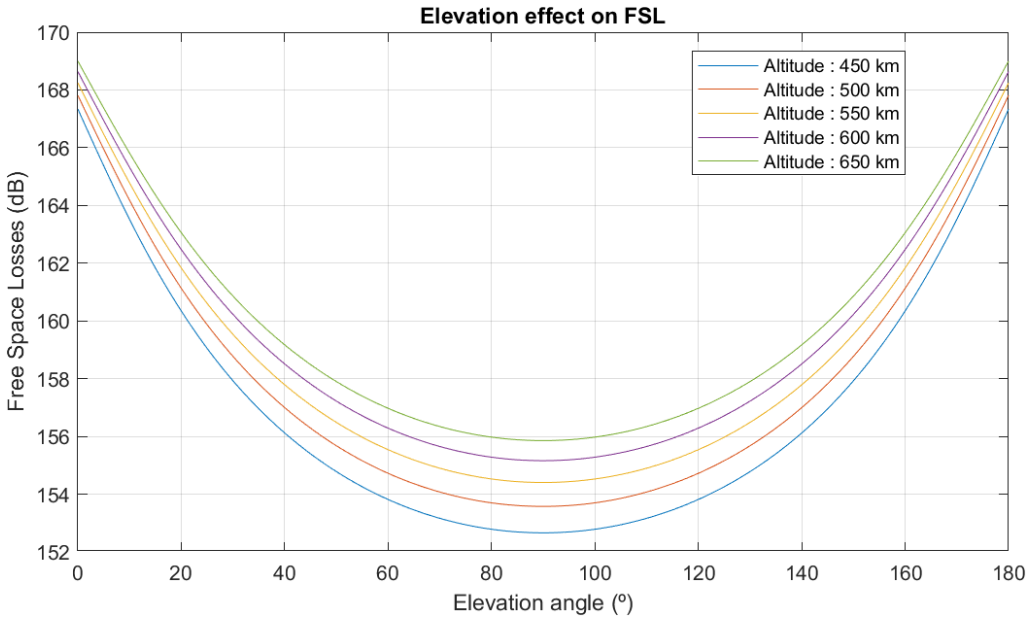


Figure 4.9: Free space losses vs elevation trajectory schematic

However, in order to have the worst case scenario, as has been noted at the beginning of the section, the total losses have been computed for a distance of 2500 km. These are depicted in Table 4.4.

Table 4.4: Link Budget losses computation

Parameters	**Value**	**Units**
TX Pointing Losses	3	dB
Transmitter Total Losses	3	dB
Free Space Path Loss	-167.56	dB
Fade Margin (dB)	5	dB
Polarization Losses	0	dB
Path Total Losses	-172.56	dB
Cable Losses	2.4	dB
Cable Total Losses	2.4	dB

4.2.3 Noise Computation

As it has been noted previously, it is critical to carry out a correct estimation of the noise that will be captured by the receiver, since the correct recovery of the signal will be dependent on it. This value will be later on used to compute the SNR of the system, which will determine its characteristics.

Table 4.5 depicts the final computation of the noise power at the entrance of the SDR, which uses equation 4.13.

Table 4.5: Link Budget noise computation

Parameters	Value	Units
Noise Figure	0.4	dB
Equivalent temperature (Te)	35.39	K
Antenna Temperature (Ta)	283.15	K
Bandwidth	2.4	MHz
LNA Gain	30	dB
Noise Power	-79.88	dBm

4.2.4 Downlink results

With the previous computations, and with Equation 4.14, the results depicted in table 4.6 can be obtained.

Table 4.6: Link Budget results

Parameters	Value	Units
Received Power	-70.06	dBm
Noise Power	-79.88	dBm
SNR	9.82	dB
SDR Rx Power	-72.46	dBm

The first value in the table is the received power, which is approximately of -70 dBm. This is the worst level of power expected, and in Figure 4.10 it can be seen that the ground station might receive up to -57 dBm.

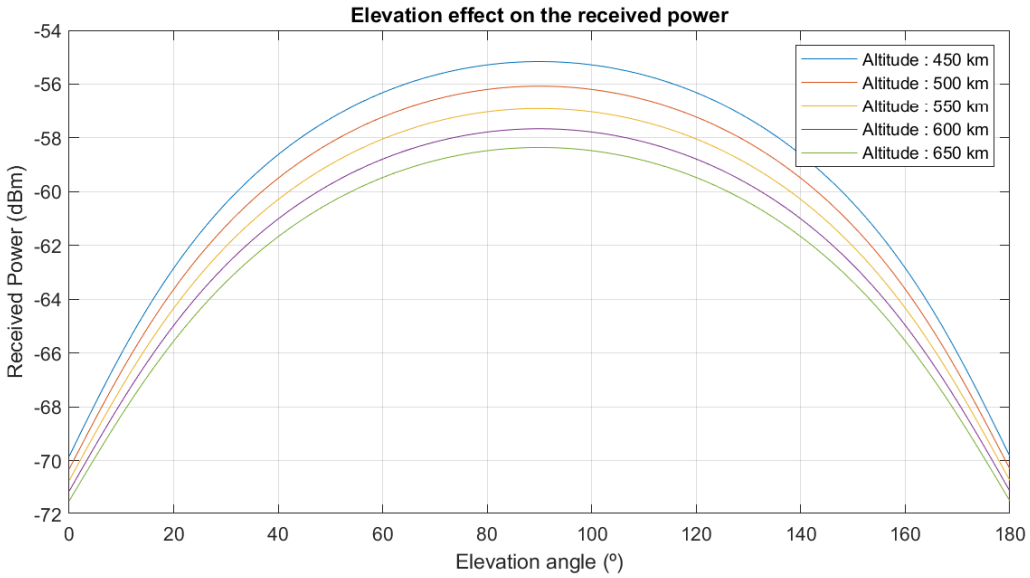


Figure 4.10: Elevation effect on the received power

Moreover, as it can be seen, the SNR of the link is of approximately 10 dB. This means that

the received signal will have a power 10 dB higher than the noise power in the receiver. Analogously to the received power, the SNR is dependent on the elevation angle, which is depicted in Figure 4.11. It can be seen that the link is expected to have values up to 23 dB SNR.

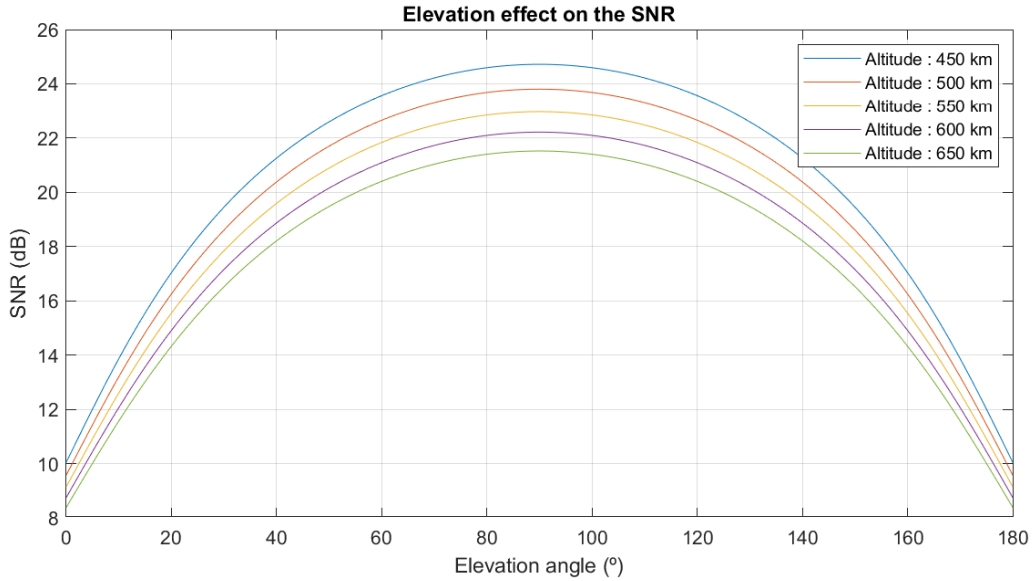


Figure 4.11: Elevation effect on the SNR

4.2.5 Bit Budget Results

In order to compute the bit rate that the link will have and the amount of data that will be sent per pass, the data contained in Table 4.7 must be used.

Table 4.7: Bit Budget parameters

Parameters	Value
Modulation	BPSK
BER	1.00e-6
Eb/No	11.3

Using the previous values and applying equation 4.17, the bit rate can be obtained.

Knowing the bit rate is paramount for the communications link design, since it determines the amount of data that the downlink will be able to transmit. A way to make this value more understandable and more useful for later system design considerations, is to compute the amount of data that can be transmitted per pass. This can be done using the STK simulation carried out in ”S-Band LOS simulation”, located in subsection 3.4.2.2. This simulation estimated over 8 minutes per passing of line of sight. Nevertheless, to consider a worst case scenario, the LOS time used will be of 7 minutes, as stated in 3.4.2.2. The results of these computations are depicted in Table 4.8.

Table 4.8: Bit Budget results

Parameters	Value	Units
Bit rate	1.705	Mbps
Download size per pass	89.514	MBytes

It is interesting to note how as the SNR changes depending on the elevation angle, the bit rate can be higher, and if a best case scenario was considered, it could go up to 4.5 MBps. This can be seen in Figure 4.12.

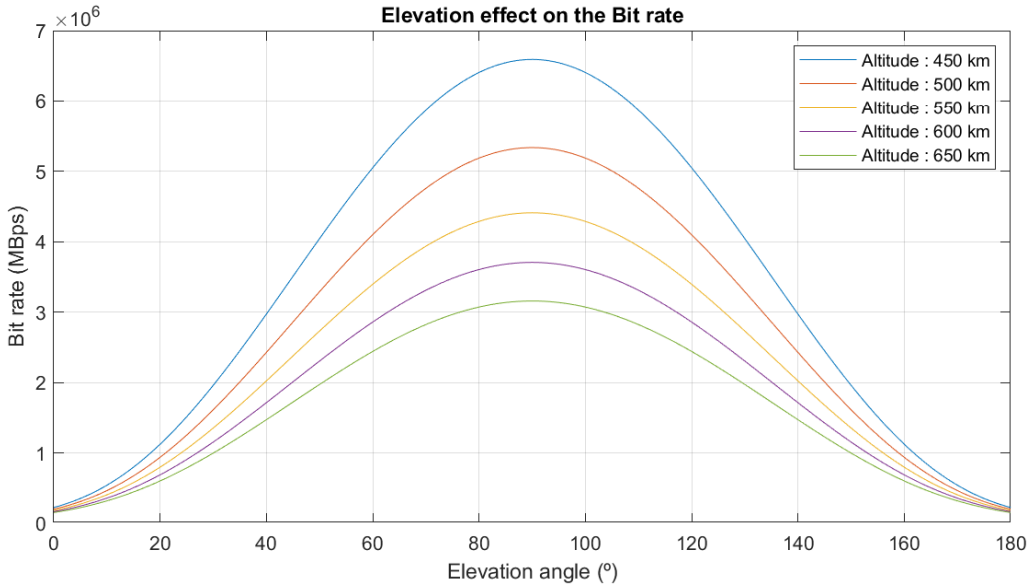


Figure 4.12: Elevation effect on the bit rate

CHAPTER 5. IMPLEMENTATION

This chapter is the heart of the project, as it details how the software has been designed and implemented.

5.1 Software Requirements

Having defined the scenario for the S-Band communications, it is time to design the communications software. This will have to deal with the interaction between the satellite and the ground station and is the basis of the project since it contains a very extensive range of functions and concepts.

In order to develop an appropriate software, several requirements must be taken into account. These are as follows:

- It must be very efficient, to be run in a limited period of time.
- The software must be appropriate for space use. This means that it should have a high level of robustness.
- Its implementation must in a language appropriate for embedded systems, that can give the programmer control over the system resources and memory.
- It must be able to communicate with the main code of the satellite, that will call it and activate its functioning.
- The software should be modular. A conscious effort should be made so that the core parts can be reused in both the satellite and the ground station
- The coding scheme must have a high level of error correction.
- The tracking implemented in the software must be able to correct the Doppler frequency shift of incoming signals.
- The codification and modulation scheme implemented must be able to function properly with a SNR of 9 dB.

5.2 Software design

5.2.1 Programming environment

The S-Band Communications software, as the rest of the RITA software, has been coded using C++ language. This language is the most appropriate for this scenario since it has many advantages. The most relevant are the following:

- **Portable:** C++ can be run easily on Linux, which is the System on Module's operative system.

- **Low-level:** This programming language is low-level, making it fully adaptable to fulfill the communication's needs.

The code editor " Visual Studio Code" has been used together with GitHub. Moreover, the code has been compiled and tested in a Linux Ubuntu operative system.

5.2.2 S-Band communication schematic

The purpose of scientific downlink systems is to convey measurements information from remotely located sources in a reliable manner. Over the years, most of the packet resources created have been custom designed and implemented on a mission-to-mission basis. The CCSDS Telemetry System was designed as a standard for space communications with the intent of both easing the transition to greater automation within space agencies as well as also ensuring harmony amongst them, resulting in a greater cross-support opportunities and services [62].

This code has been designed following the CCSDS Telemetry Standard, nevertheless, it has been adapted to the needs and requirements of the mission in order to improve its efficiency and reduce its development time.

The communication's software is in charge of three main tasks: managing the application layer and network, coding and encoding packets and transmitting and receiving them through a physical layer. In order to avoid unnecessary mistakes and to make an efficient design, the code has been designed with modularity and divided into these three parts, that are independent from each other. Each of the modules has the necessary functions needed for the uplink and the downlink. This can be seen clearly in Figure 5.1, where both uplink and downlink paths use the three blocks of the code.

This modularity is also very useful since it which allows the software to be functional in both satellite and ground station only by connecting the blocks in a different order.

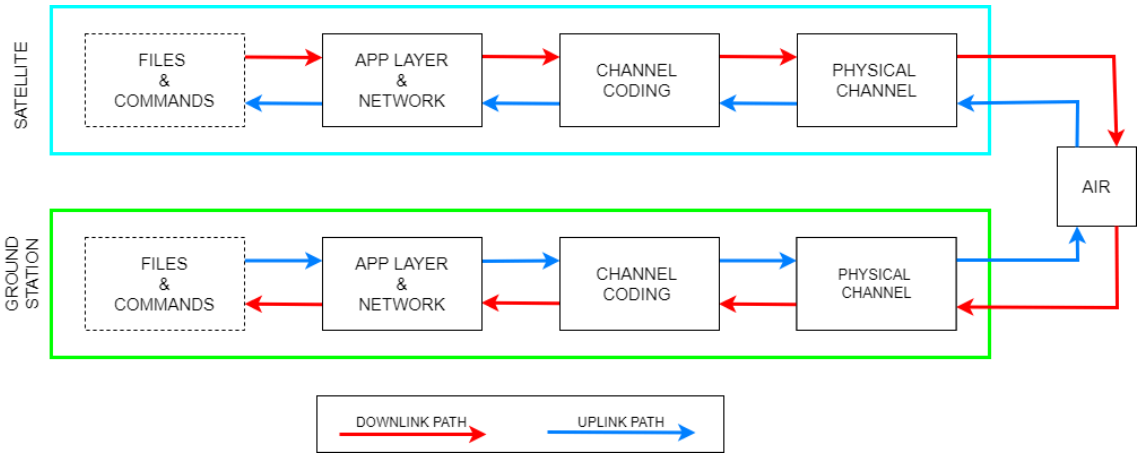


Figure 5.1: Software block schematic

5.2.3 Software structure

The software is organized in the following folders:

- **app:** This folder contains two main programs that can be run:
  - **mainSAT.cpp**: This code runs all the functions to be carried out in the satellite.
  - **mainGS.cpp**: This code runs all the functions to be carried out in the ground station.
- **build:** This folder is where the software is compiled using CMake as the build system.
- **GS Data Folder:** This folder is only for testing and simulates the data folder of the ground station.
- **RITA Data Folder:** This folder is only for testing and simulates the data folder of the satellite.
- **lib:** This folder contains all the external libraries used.
- **src:** This folder contains all the source files that have been coded and that are necessary for the main program.
  - **applayer:** This folder contains all the source files with their respective functions used to manage the app layer
  - **channel:** This folder contains all the source files with their respective functions used to manage the channel encoding and decoding.
  - **phy:** This folder contains all the source files with their respective functions used to manage the physical channel.

It should be noted that this software uses CMake in order to control the software compilation process. This is "*an open-source, cross-platform family of tools designed to build, test and package software*" [63] that is very commonly used in these kinds of projects.

5.2.4 Satellite main software

Figure 5.2 depicts how the main software has been structured for RITA. It should be noted that this schematic is but an overview of the whole code, and uses a number of specific functions that will be explained in the following sections.

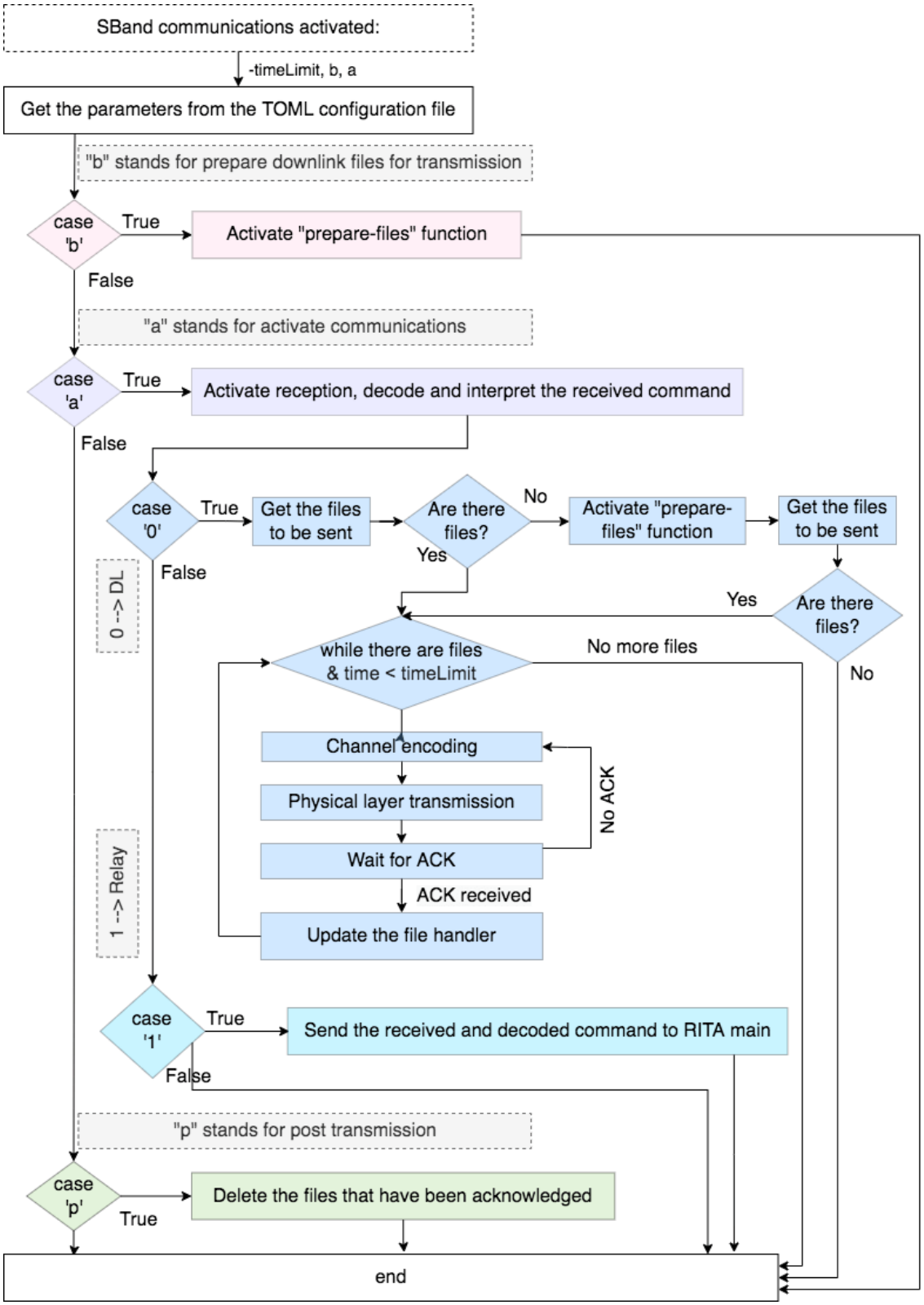


Figure 5.2: Satellite Application Layer Schematic

As it can be seen, it consists on 3 main parts:

- **"Prepare files":** This functionality prepares the files that must be sent by compressing them carrying a set of processes to create the packets that have be sent once the transmission is activated. This functionality can be called by the main code at any convenient time and aims to reduce computation time when the downlink function is activated, by carrying these processes previously.
- **"Activate communications":** This functionality first activates the reception and awaits for a ground station's command. If this indicates to start the downlink, the transmission will be carried out. On the other hand, if it receives a relay message, it will interpret it and pass it to the main code as a parameter.
- **"Post-transmission":** This functionality is also aimed at maximizing the efficiency of the communications. This should be activated by the main code of RITA after a transmission has been made, and will erase the packets that have received ACK and therefore have been transmitted correctly.

5.2.5 Ground Station main software

Analogously to the satellite's main software, Figure 5.3 depicts an overview of the main software for the ground station.

This has two main parts:

- **"Activate downlink":** This functionality generates a downlink frame and transmits it to the satellite in order to activate the downlink. Once done, it activates its receiver and recovers the packets transmitted by the satellite and sends ACK messages accordingly. Once the transmission has been finished, it processes the files and prepares them to be accessed by an end user.
- **"Activate relay":** This functionality transmits a message containing a command to the satellite.

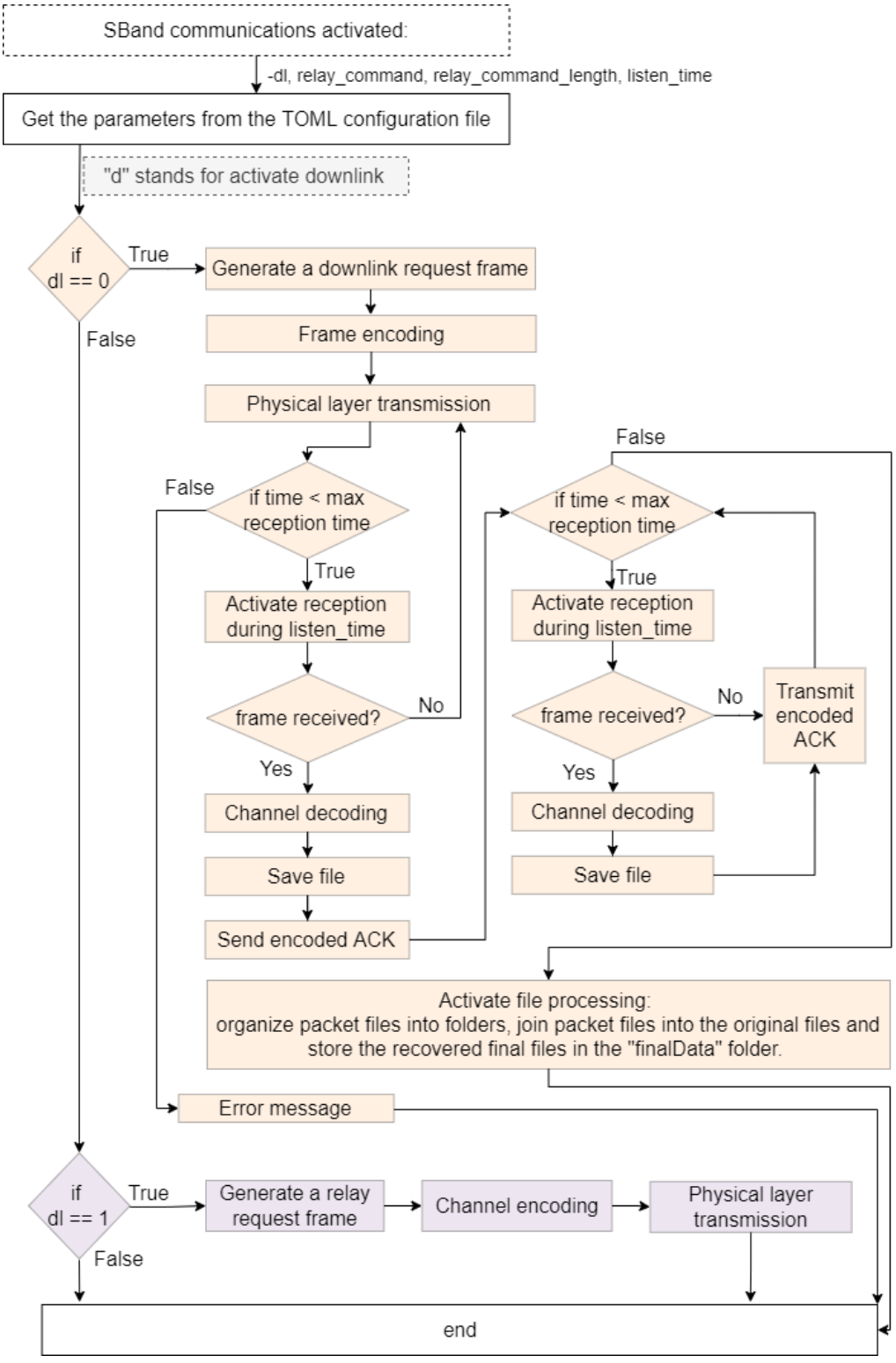


Figure 5.3: Ground Station Application Layer Schematic

5.3 Application Layer and Network

This code is in charge of managing the data. When transmitting information, it will be in charge of accessing the data, compressing it and splitting it into packets. In the receiver side, it must store the packets, join them in files in the appropriate order and decompress them as they were first before they were sent.

5.3.1 ARQ Protocol

In space communications it is of utmost importance to have a very reliable communications protocol. Therefore, the S-band communications protocol will include the Automatic Repeat Request technique to comply with this requirement. This subsection will explain with detail the protocol and its different techniques and then justify why the chosen one is best for the design of the S-band communications.

The Automatic Repeat Request (ARQ) technique was invented during World War II by H.C.A van Duuren in order to provide reliable transmission over radio. Nowadays this technique is the basis for peer-to-peer protocols and it is widely used in communications to ensure that a data stream is delivered correctly in spite of errors that might occur in the transmission.

This technique is known for its robustness and effectiveness, which are crucial for space communications. ARQ protocol has the ability to adapt to variable channels, if the channel is noisy it is capable of adapting the transmission rate for its specific capability. This makes this protocol a safe choice when dealing with channels of varying or unknown characteristics.

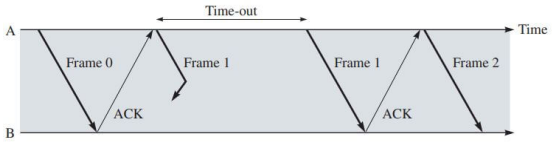
In order for this technique to work properly, the blocks of information that are sent must contain a header with control information. Moreover, in the studied scenario, the frames sent cannot pass previously sent frames, therefore, under this assumption, the main objective of applying the ARQ protocol is to ensure that packets get to the destination only one time, without any duplicates and in the same order of transmission.

There are 3 techniques used in ARQ, which are: Stop-and-wait ARQ, Go-Back-N ARQ and Selective Repeat ARQ, which are all designed as flow control mechanisms.

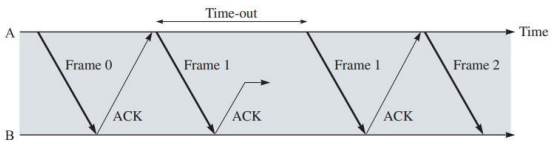
5.3.1.1 Stop-and-wait ARQ

In this protocol, the sender sends a single frame, waits for ACK (acknowledgement) from the receiver and only once the acknowledgement has been received, sends another frame. If the sender does not receive the ACK after a determinate period of time (timeout), it sends the previous frame again.

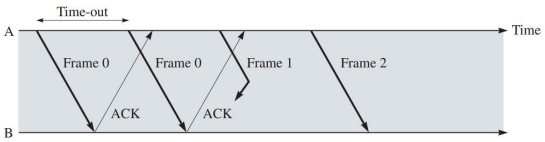
There are two possible error cases that might arise in the communication, these are represented in Figure 5.4.



(a) Stop-and-wait when a frame is lost



(b) Stop-and-wait when an ACK is lost



(c) Stop-and-wait when the time-out is incorrect for the transmission

Figure 5.4: Stop-and-wait representation with different cases [64]

Figure 5.4(a) demonstrates how this protocol works when a packet is lost. In this case, the sender (A), if after a given time has not received an ACK from the receiver (B), will automatically send again the lost packet.

Another error that might occur is represented in Figure 5.4(b), where the ACK is lost, and for the same reasons as in the previous case, since the sender (A) has not received the ACK from the receiver (B) it sends the packet again. The loss of an ACK can then result in the reception of a duplicate packet. This can be solved by adding a sequence number in the header of the packet, that would allow the receiver to later detect that the packet is a duplicate and discard it.

Furthermore, as it can be seen in Figure 5.4(c), it is of extreme importance to declare the timeout adequately for the transmission channel. If this value is too high, when a packet is lost, the sender will take a very long time to re-transmit the packet, which will be very inefficient. On the other hand, if the value is too small, the sender will mistakenly re-transmit packets before it has had time to receive acknowledgements and packets will be lost.

5.3.1.2 Go-Back-N ARQ

The Go-Back-N ARQ protocol is similar to the Stop-and-Wait, but increases its efficiency by sending a determinate number of frames before requiring an ACK from the receiver. This works as follows: firstly, the sender transmits the first frame and then proceeds to send $W_s - 1$ additional frames, trusting that the first frame was sent and received correctly. If the transmission has indeed worked correctly, the sender will receive an ACK for the first frame while it is sending more frames into the channel. If all the transmissions work properly, the sender will continue to send more frames while the ACK keep arriving. If a frame, however is lost in the transmission, the receiver ignores all subsequent frames. Therefore, the sender will stop receiving ACK for the frames, and once it reaches its maximum number of outstanding frames, it will proceed to re-transmit the lost frame and the subsequent frames.

This idea can be seen in Figure 5.5, where it can be seen that an error occurs in the transmission of frame 3, and eventually, after 4 frames (the declared window), the sender

detects that frame 3 has been lost and proceeds to re-transmit it along with its subsequent frames.

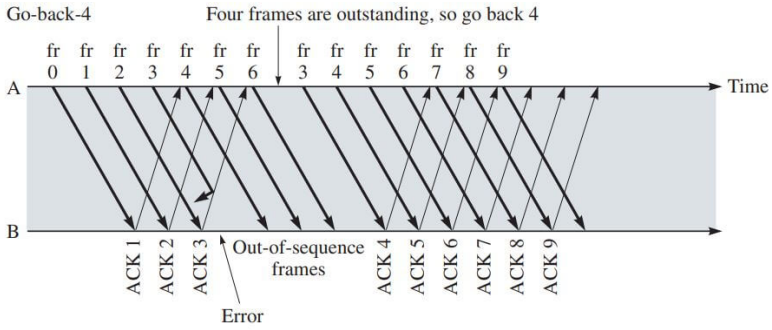


Figure 5.5: Go-Back-N protocol [64]

In order for this protocol to work correctly, the transmitter must have a timer for all the transmitted frames and have a buffer containing the frames transmitted that have not been acknowledged.

This protocol is a sliding-window protocol, which is why multiple frames can be sent at a time without the ACK from their previous frames. For this protocol to work correctly it is imperative that the window size is chosen appropriately to the system. Some key ideas are as follows:

- The size of the window should be limited by the receiver's packet processing ability.
- This should be smaller than the number of sequence numbers.
- When fulfilling the two previous conditions, the size should be the highest possible value.

5.3.1.3 Selective Repeat ARQ

The Selective Repeat protocol is a much more efficient and complex protocol than its two predecessors since it includes two new features. These are the following:

- The receiver is able to accept frames that are not in subsequent order. This is done by having reception window larger than one frame.
- When an error occurs in the transmission of a file, the receiver can request the transmission of the missing frame, so that only the individual frame is re-transmitted.

The main idea of this protocol is very similar to Go-Back-N except for the fact that buffers are used in both the transmitter and the receiver to be able to keep track of the transmitted and received files. In the transmitter side, the sliding window contains the packets that are sent or that can be sent, and on the other side, the receiver's sliding window, of the same size, covers the packets that are received or are expected to be received.

Figure 5.6 depicts how the sliding windows and buffers of both receiver and transmitter side are designed.

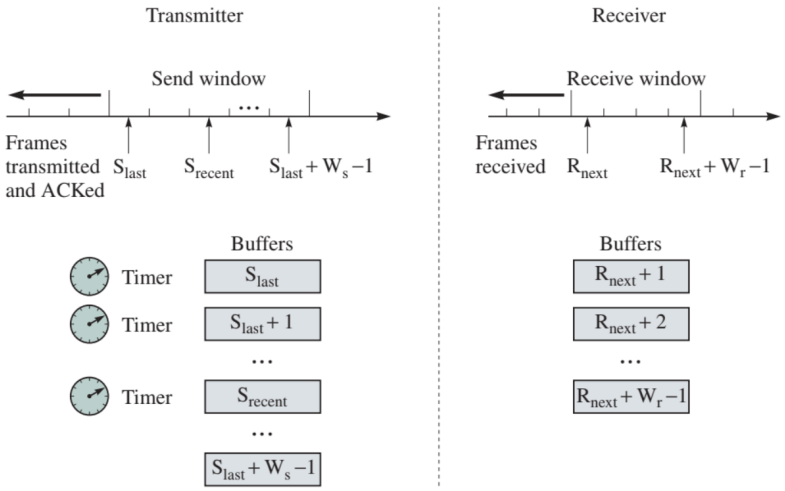


Figure 5.6: Selective Repeat protocol functioning [64]

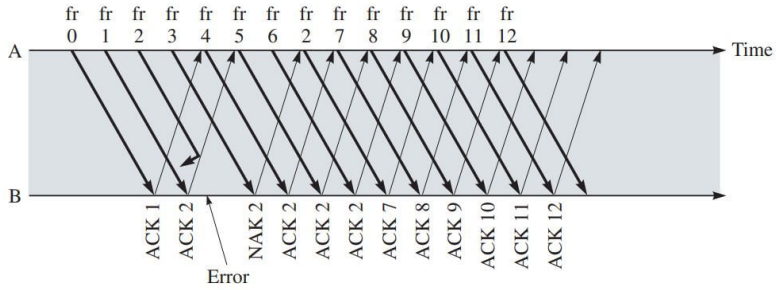


Figure 5.7: Selective Repeat protocol packet example [64]

5.3.1.4 Implementation

As it can be seen, the three techniques are very similar but have clear performance and efficiency differences. The Stop-and-Wait protocol is extremely inefficient when the propagation time of the scenario is very high. On the other hand, the Go-Back-N protocol is much more efficient than the previous since it allows the transmitter to continue sending frames while the acknowledgements are being sent. Nevertheless, this protocol can be very inefficient in channels with very high error rates, since the protocol will re-transmit the frame error and its subsequent frames. This is improved with the Selective Repeat protocol, which presents a much higher efficiency than the rest.

With the previous reasoning, the best protocol to implement is the Selective Repeat. Nevertheless, the Stop-and-Wait has been the chosen protocol for implementation. This decision has been made for the following reasons:

- This protocol is the less complex to implement and due to the time restrictions imposed by the payload delivery date, it is the best option to be able to make sure it is programmed correctly and that it can be tested adequately. Nevertheless, once it is fully working, this protocol should be reconsidered.

- This protocol is inefficient when the transmission time is very high. In this scenario, the transmission time is not high enough to have an unacceptable impact on the transmission, since the satellite will be at a LEO. It is estimated that every message will be delayed approximately 1.67 ms, which means that since a packet is sent and its respective ACK is received, approximately 3.34 ms will pass in the best case scenario that neither the packet nor the ACK messages are lost.

Moreover, it should be noted that the software has been designed without a hardcoded timeout, but using a variable that can easily be changed in the configuration file of the system. This will allow the operators to choose this value appropriately and adapt it if necessary whenever is required.

5.3.2 File management

This subsection details how the downlink file management has been carried out. There are a number of ways to manage files successfully. Several designs were made, nevertheless this specific one was chosen for its simplicity over the others.

5.3.2.1 Overview of the design

As mentioned previously, the system's task is to deal with files in a manner that they can be transmitted in frames, and reconstructed when these frames are received. The main issue then is how the frame design is done.

In order to simplify this task as much as possible, the following ideas have been implemented:

- Instead of dealing with several files one by one, all the files to be transmitted will be joined into a single compressed one.
- The files will be named with the timestamp of their creation. This will ensure that when more files are created, none will have the same filename. Moreover, it will help to keep track of the files and the user will know when the files where created originally, even if they are received on later transmission passes.
- The packet division will not be done after reading the file, but instead, the file will be split into several same-sized smaller "packet" files that will be named with the original filename and an added division number. This has many advantages:
  - By splitting the original files into smaller ones that will be sent one at a time, the RAM memory of the system is used more efficiently.
  - That each frame transmits a file makes it much easier to keep track of the packets that have been sent and the ones that haven't.

The files have to be treated in the satellite to be sent with frames, and then in the ground station, where the received frames have to be converted into the original packets. The two linked designs are depicted below.

5.3.2.2 *RITA system*

The main task of the S-Band system in RITA is to gather files and transmit them through the antenna. Therefore, it is imperative to understand how the files are organized in the satellite.

The S-Band dedicated folder in RITA is divided into the following folders, depicted on Figure 5.8. There are two folders: one for the treatment of data and another containing the software configuration files.

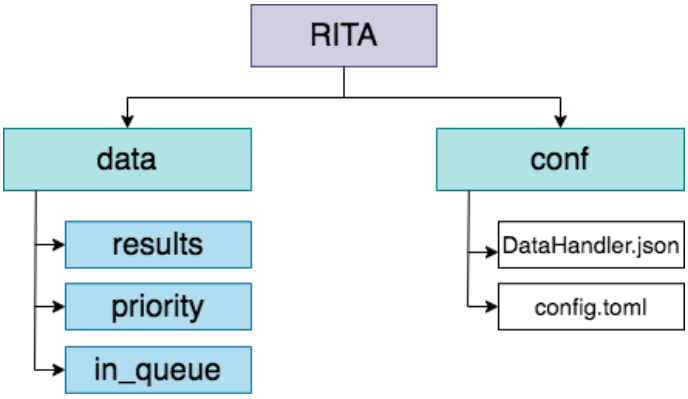


Figure 5.8: RITA S-Band Folder Schematic

The data folder contains three other folders: ”results”, ”priority” and ”in_queue”. RITA’s main code will save the files that have to be transmitted into the ”results” folder or the ”priority” folder, depending on the importance of the files.

When the ”prepare files” function of the S-Band code is activated, this will deal with the files stored in these folders carrying out the following process.

1. The system will check the ”priority” folder to see if it contains any files. If this folder is empty, nothing will be done. On the other hand, if there are files, the system will compress the folder and name it using a priority flag ”0” followed by the timestamp.

2. The system will check the ”results” folder to see if it contains any files. If this folder is empty, nothing will be done. On the other hand, if there are files, the system will compress the folder and name it using a priority flag ”1” followed by the timestamp.

3. Once the folders have been compressed, if this process has not resulted in any errors, the files inside the folders will be erased

4. The compressed files that have been created in the first two steps are now split into several smaller files with a specific size noted in the configuration file of the system. Then, the number of splits that the files have experienced is computed and added into the filename of all the files. This filename design is depicted in Figure 5.9.

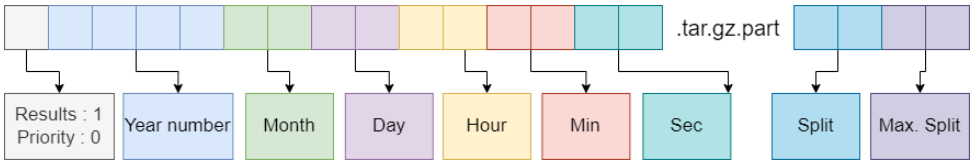


Figure 5.9: Filename structure

An example of a possible filename of a priority compressed file created the 9th of July in 2022 at 12:30:59, which was the 1st split of a total of 3 would be:

”020220709123059.tar.gz.aaac”

5. The resulting files are moved into the ”in_queue” folder. This folder is designed to contain all the files that have to be transmitted. Moreover, the added folders are written in a ”Data Handler” file in order to keep track of all the files faster. This handler is explained in detail 5.3.2.2.

6. If the previous process is done correctly, the two initial compressed files are deleted.

As it can be seen, the configuration folder contains two files:

- **sband.toml**

  In order to be able to change and adapt the functioning of the code in an easy manner, a decision has been made to create a configuration file with the main parameters of the codification of the packets.

  The configuration file has been made with TOML [65], since it is a file format that is used specifically for configuration files. This language is easy to read and write and is designed to map to a hash table [66].

  The following code contains an example of a possible configuration file that the satellite might have.

```
title = "SAT_sband"

[sband]
directory = "/RITA/sband/data"
encode = true
convolutional_coding_order =7
convolutional_coding_rate =2
interleaving = 10
split_size = 504
```

- **DataHandler.json**

  In order to keep track of the files that RITA must transmit, a JSON [67] file has been designed. This will keep track of the files that have to be sent, have been sent, and those whose ACK messages have been received. This will be used as an optimization tool to make the process more robust, and make sure that all the files are transmitted. This will also help keep track of files that might not fit in a single pass and have to be transmitted in the following one.

The following code contains an example of a possible file that the satellite might have.

```
{
    "020220222205345.tar.gz.partaaab": "ACK",
    "120220228102808.tar.gz.partabab": "ACK"
}
```

5.3.2.3 Ground station system

The ground station file system design has been done in a very similar way to the previous. Analogously to the RITA configuration, it is divided into two folders: one for the data and another for the configuration files, which can be seen in Figure 5.10.

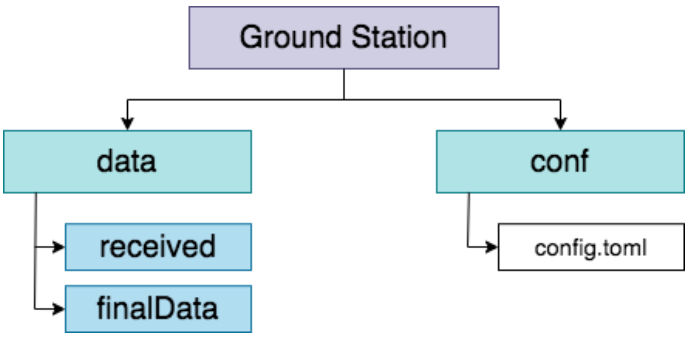


Figure 5.10: Ground Station S-Band Folder Schematic

The way in which the ground station deals with files is the following:

1. When the ground station is receiving frames, its function is to detect the frames, decode them and save them in files. This files will have the same filenames as in the satellite, since the filename will be encoded in the header of the frame, which is detailed in 5.3.3.2. These files are stored in the ”received” folder.
2. When the reception of frames has stopped, the file management starts. The first step is to join the files having the same timestamp and different parts into folders. Therefore folders named with the timestamp are created and the files coinciding with them are put inside.
3. Once the received folder only contains other folders with the data, the system checks folder by folder if each contains all the split files. If that is the case, the files are joined, and the resulting recovered file is decompressed and moved into the ”finalData” folder for the user to read.
4. If the previous process is done correctly, the folders are deleted and only remain those folders that miss parts, which will be received in the following satellite pass.

It should also be noted that the ground station contains a *”conf”* folder with a configuration TOML file. This file must contain the exact same configuration as the one in the satellite in order for the code to be able to decode the packets correctly.

5.3.3 Application layer design

When designing a satellite's communications link, it is necessary to define the messages that will be used in both the uplink and the downlink. Since these two links have different functionalities, to make the code more efficient, they have been designed separately.

5.3.3.1 Uplink telecommands

The uplink has been designed with three types of messages. These are as follows:

1. **Downlink activation**: In order to activate the downlink function of the satellite, the ground station will send a "Downlink Activation" packet. This is characterized by the code "00".

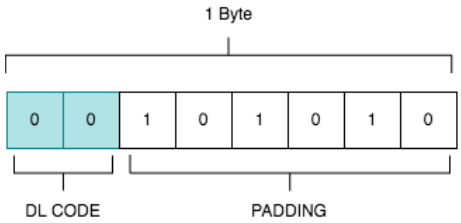


Figure 5.11: Downlink activation packet

2. **Relay activation**: In order to activate the relay function of the satellite, the ground station will send a "Relay Activation" packet. This is characterized by the code "01" coded in the first two bits of the first byte. The following byte contains the length of the instructions to be sent, and the following bytes contain the instructions, occupying as many bytes as indicated in the length.

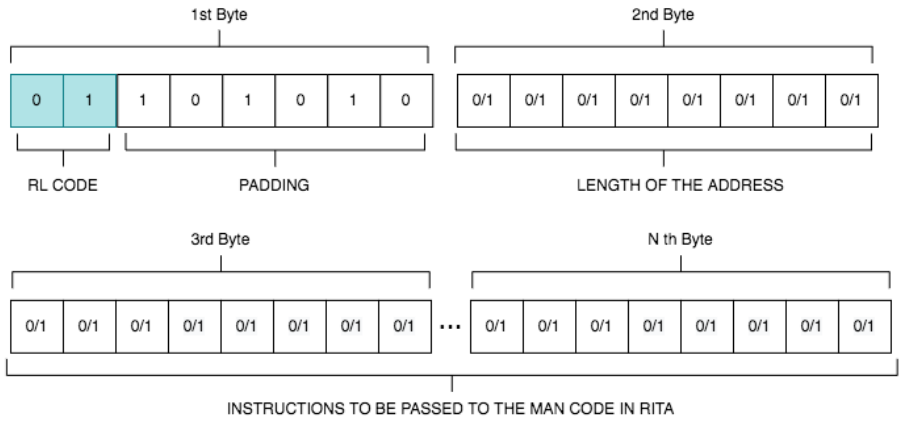


Figure 5.12: Relay activation packet

3. **ACK**: In order to carry out appropriately the ARQ protocol, and to make it more robust, after a packet is received in the ground station, an acknowledgement message will be sent. Its design is depicted below:

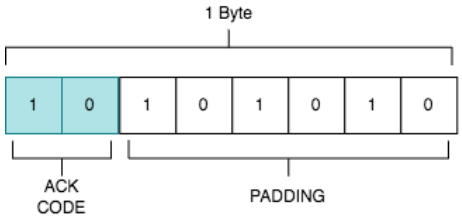


Figure 5.13: ACK packet

5.3.3.2 *Downlink telecommands*

The ground station will only receive one type of message, which will contain the data information and will have a fixed length, determined by the configuration parameters in the ”config.toml”, most specifically, the split size of the files.

This message will contain a header with the filename information. This will help ease the management of the files in the ground station. The header structure is depicted in Figure 5.14.

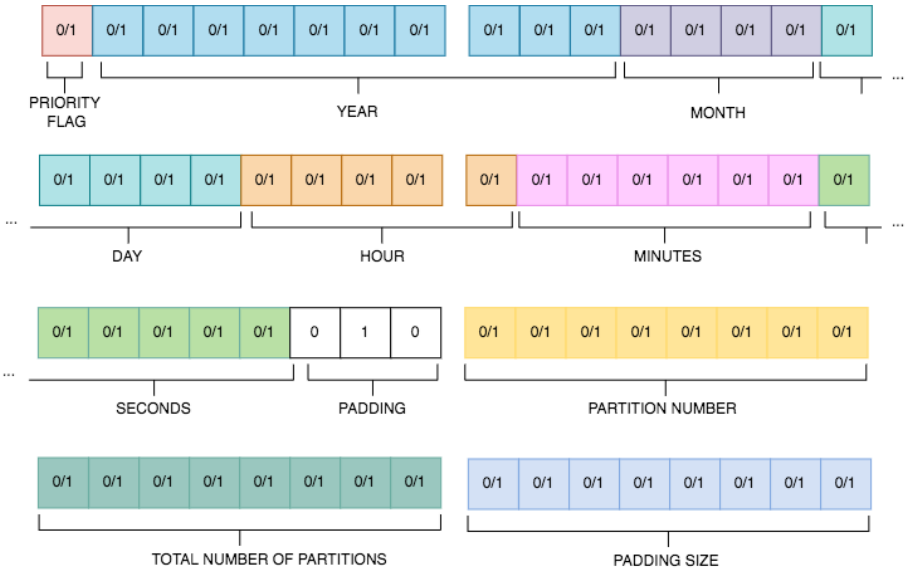


Figure 5.14: Header structure

As it can be seen, it contains the same information stored in the filename since it contains the priority flag (0 for the ”priority” folder and 1 for the ”results” folder data) and it is followed by the time the file has been created, as well as the partition number, the total number of partitions of the original file. Moreover, in this case, it also contains the padding size of the packet.

5.3.4 Software implementation

In order to implement the functions explained above, several files have been designed with appropriate code to carry them out. These are explained in appendix D.1, that depicts

all the files used and their respective functions. Moreover, it also indicates the external libraries used for this implementation.

5.4 Channel coding

The channel coding block, which is based on the CCSDS TM Synchronization and Channel Coding [62], is used to protect the transfer frames against errors that might occur on the physical channel.

This section explains in detail the channel coding block, which constitutes of all the algorithms and data processing that are done to the files.

5.4.1 Choosing the appropriate coding and decoding protocols

Figure 5.15 depicts the performance on a Gaussian channel of various recommended codes. All of them are included in the CCSDS Recommended Standard since they all provide a substantial coding gain over system without coding. This can be appreciated in the figure, since the blue line depicting the ”uncoded” results requires a much higher Eb/No than all of the other coded results.

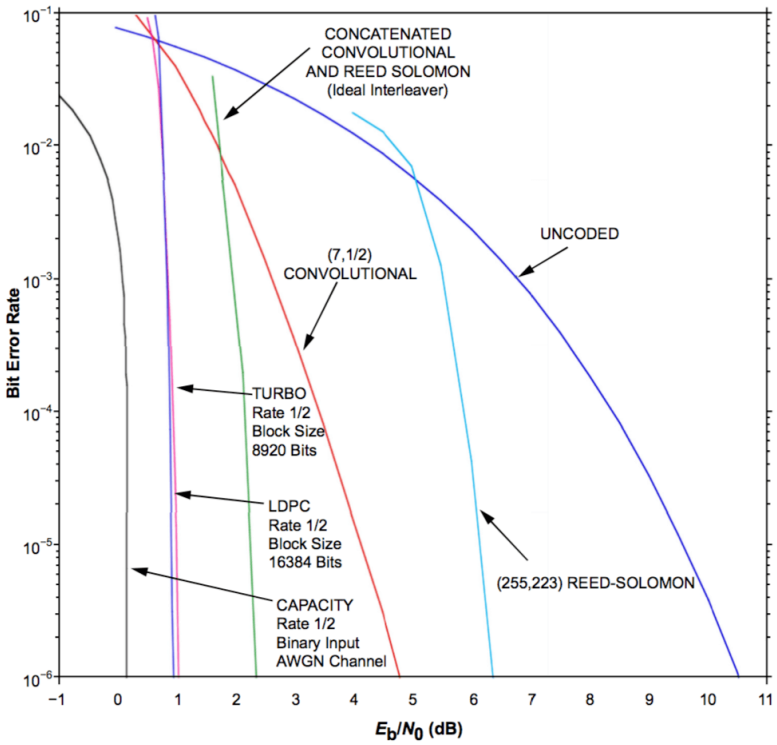


Figure 5.15: BER per Eb/No [62]

At first glance, it is clear that all of the options are adequate for the design, since the system should work with 11.3 dB Eb/No. Nevertheless, the lower the Eb/No (dB) is, the better, since it indicates that the system will work adequately with worst conditions.

A very typical implementation for simple space communication systems is the combination of the Concatenated Convolutional and Reed-Solomon with an ideal interleaver [62]. This fulfills the system's needs and therefore is the chosen design for the system's channel coding.

5.4.2 Channel coding architecture

Having determined that a combination of the Reed-Solomon codes and the Concatenated Convolutional codes will be implemented, the block structure for its correct functioning must be defined. This is depicted in Figure 5.16.

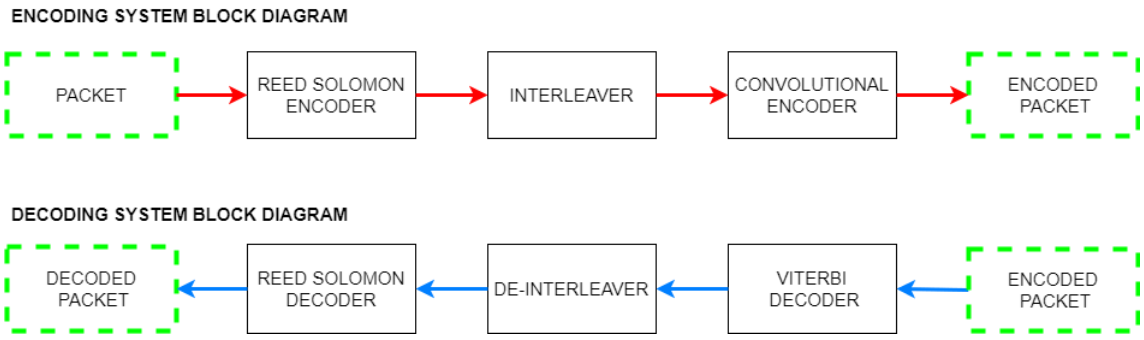


Figure 5.16: Channel coding system block diagram

These blocks will be explained with detail in the following sections.

5.4.3 Reed Solomon Codes

Irving S.Reed and Gustave Solomon published "Polynomial Codes Over Certain Finite Fields" in 1960. In this paper they presented an error-correcting code based on sampling points on a polynomial. The presented decoding algorithm was unfortunately unfeasible except for small codes[68]. This problem was resolved by modifying their original scheme to a BCH (Bose Chadhuri-Hocquengh) code like scheme based on a polynomial known to both encoder and decoder.

In 1968, Elwyn Berlekamp and James Massey introduced the Berlekamp-Massey decoding algorithm and improved the existing BCH scheme decoder. Furthermore, in 1975 Yasuo Sugiyama developed an improved BCH scheme decoder based on the extended Euclidean algorithm. Finally, in 1977, Reed Solomon concatenated error correction codes where implemented in the Voyager program. Nowadays these codes are widely used in digital communications standards and digital storage devices.

Reed Solomon (RS) are block codes. In the case of (255, k), they allow to input a fixed size block of $k = 255 - 2E$ symbols to be processed and encoded into 255 output symbols. CCSDS protocols recommend the used of (255,223) RS codes for space communications, which are able to correct up to 16 errors. The structure of this frame is represented in Figure 5.17.

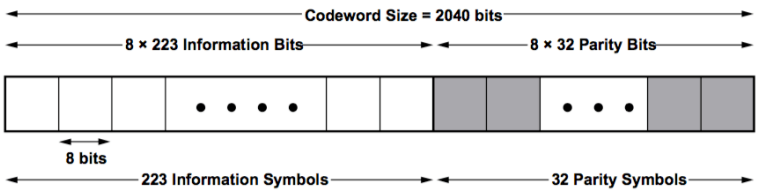


Figure 5.17: Reed Solomon Codeword (255,223) [62]

This linear codes are used effectively in concatenated coding schemes, where the encoded RS symbols are further encoded by a convolutional code. It should be noted that their error probability exponentially decreases with the block length. Moreover, its complexity is proportional to a small power of n-k, where *n* is the block length and *k* the data symbols.

RS codes are transparent codes, which means that in the case that the channel symbols are inverted, the decoders will still operate, and return the complement of the original data. With an exception commonly of the codeword in which the inversion has occurred. This is particularly useful for BPSK modulations such as the one that will be used in this communication link, since there is no way to avoid the inversion of the symbols [62].

5.4.3.1 Implementation

Figure 5.18 shows how the Reed-Solomon technique is implemented, requiring an input frame with a length multiple to 223 Bytes and returning an encoded frame with a length multiple to 255 Bytes.

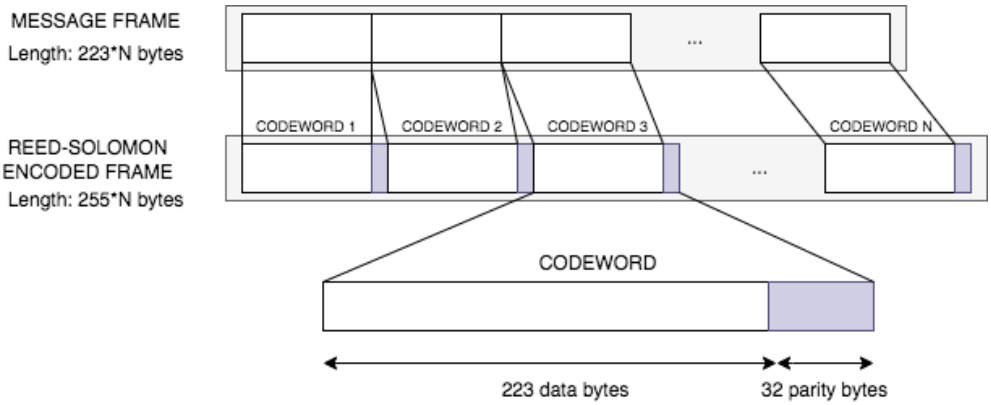


Figure 5.18: Reed Solomon Implementation schematic with RS (225,223) codeword

5.4.4 Interleaving

In space communications, transmission can be effected by deep fading and multipath. This leads to burst errors which tend to be localized in a specific frame, in this case, a Reed-Solomon code-word. Having a code-word with a great number of errors highly degrades the decoding, since there is a limit of errors that can be corrected. By using the interleaving technique, which alters the order of a sequence of bits, the code performance can be improved. Interleaving distributes uniformly burst errors across various code-words

in the data sequence, which enhances the Reed-Solomon code or the error correcting code used in the system [62].

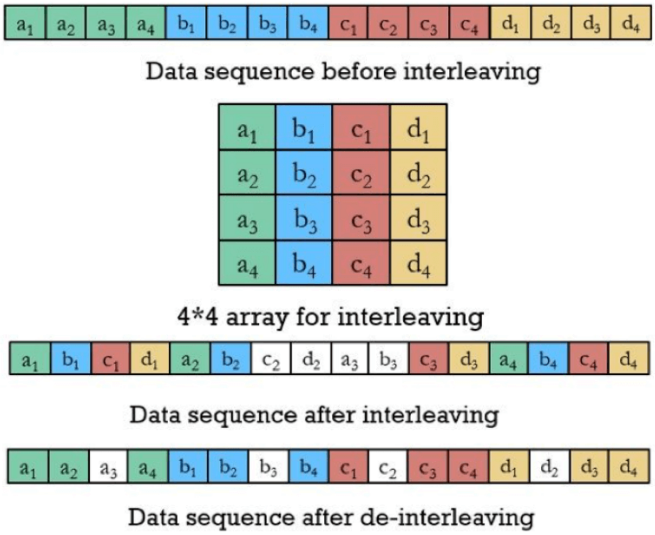


Figure 5.19: Interleaving [69]

Figure 5.19 depicts how the interleaving technique works, following the next steps:

1. There is an initial data sequence to interleave.
2. The data sequence is organized into a matrix by columns.
3. The interleaved data sequence is extracted by reading the matrix row by row

Additionally, it shows how if in transmission, an interleaved sequence of bits experiences an error burst, once de-interleaved, the error bits are divided uniformly along the data sequence, which as stated previously, is the goal of this technique.

5.4.5 Convolutional Coding

Convolutional codes are a type of error-correcting codes that were introduced by Peter Elias in 1955. By that time, these codes where thought to be decoded only at the expense of high computation and delay. Nevertheless, in 1967, Andrew Viterbi developed the Viterbi algorithm to decode them with reasonable complexity. Several algorithms exist now for convolutional codes decoding, nevertheless, the Viterbi algorithm remains universally used.

These codes are similar to block codes in that they involve the transmission of parity bits that are computed from message bits. Nevertheless, in convolutional codes, the message bits are not sent with the parity bits, but in this case, only the parity bits are. In order to do this, the encoder has a sliding window to compute the parity bits through a combination of various subsets of bits in the window. The size of the window (expressed in bits) is the length's constraint. The longer it is, the larger amount of parity bits that are influenced by

a message and therefore, the greater the resilience to bit errors. However, this comes with a trade-off, since larger lengths also imply higher computational costs for the decoding of the parity bits that come with delays. [70]

Convolutional codes are specified by the following parameters : n, k, R_c and K.

- **k:** The number of inputs.
- **n:** The number of outputs.
- R_c**:** The coding rate determines the number of data bits that are encoded per bit and is defined as $R_c = \frac{k}{n}$.
- **K:** The constraint length of the convolutional code.

The recommended CCSDS standard convolutional code for space applications is (7,1/2), which indicates a K=7 length constraint and a 1/2 bit rate. This specific code is known for it's transparency. This means that at a steady-state, should the input sequence of the decoder be inverted, the output will also be inverted. This is particularly useful when using BPSK modulation, since there is a 180º phase ambiguity and the demodulator can then produce the inverse transmitted symbols even when being in lock. Transparent codes allow that when the demodulator has produced the inverse of the transmitted symbols, the decoder returns the inverse of the encoded bits, and if specified in the header telemetry, it is easy to invert the bits back to the original state. [62]

Figure 5.20 depicts the diagram of an encoder for the aforementioned code. This encoder structure depends on how the adders are connected to the shift register. These connections are defined by a set of vectors. When the vector is 1 in position i, then the ith stage of the shift register is connected to the lth adder, and when it is 0, there is no connection.

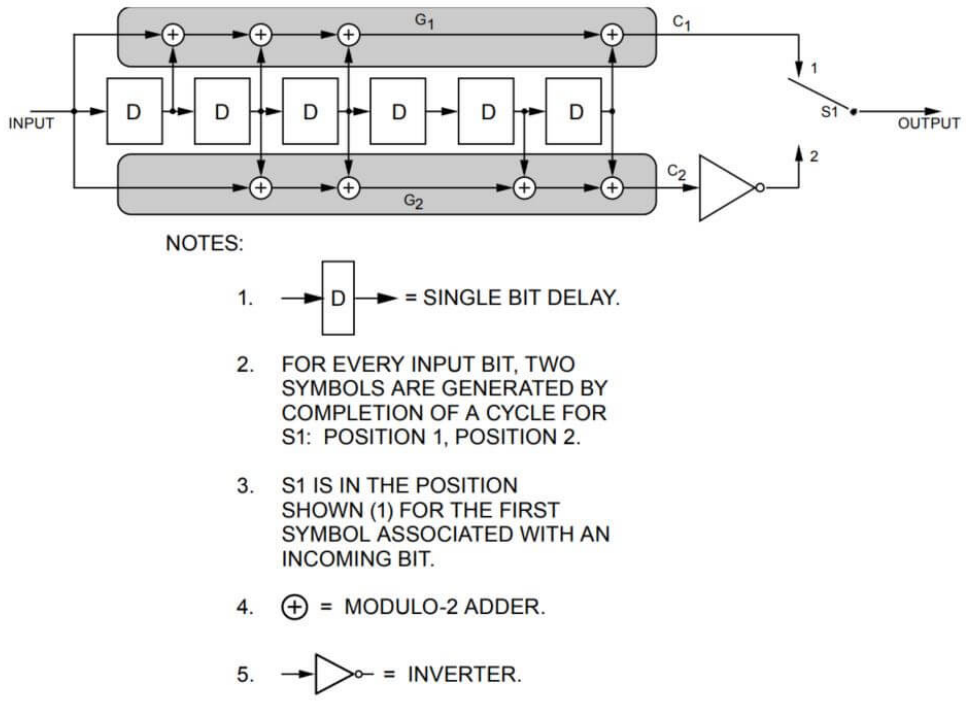


Figure 5.20: Example of a Convolutional Encoder (7,1/2) [62]

5.4.5.1 *Implementation*

Since the recommended configuration for space applications is the convolutional code (7,1/2), these are the specifications that will be implemented.

Any message length can be encoded with this system, and its resulting frame length can be computed with the following equation:

$$N_{\text{encoded bits}} = rate * (N_{\text{message bits}} + order + 1) \tag{5.1}$$

Therefore, for these characteristics, the length will be $2 * (N_{messagebits} * 7 + 1)$, dependent on the incoming message length. This can be seen in the following figure.

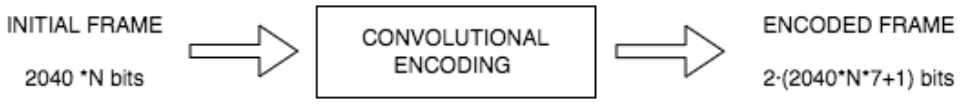


Figure 5.21: Convolutional code implementation

5.4.6 Functioning of the software codification channel

Even though the encoding and decoding for both uplink and downlink is based on the same techniques, its implementation in software slightly differs from each other.

5.4.6.1 *Downlink*

The following diagram depicts the functions carried our in the channel coding for the downlink:

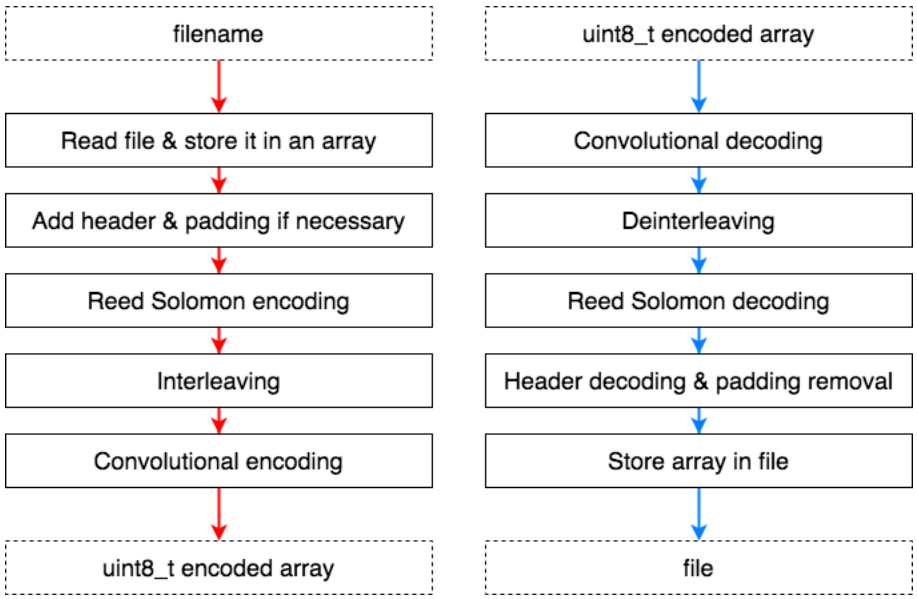


Figure 5.22: Downlink channel coding code block diagram

As it can be seen, in the satellite, the code is given a filename to be sent. This file is then read and stored into an array. A specific header is added to the beginning of the array (containing the filename), as well as padding at the end if required to fit the specifications of the encoding process (the incoming array must be a multiple of 223 bytes). After that the array undergoes the Reed-Solomon encoding, is interleaved and then it is convolutional encoded. The process returns a uint8_t array with the encoded frame to be sent to the physical layer of the code. Analogously, the decoding process follows the inverse steps, creating a file with the same filename as the file had in the satellite, containing the decoded message.

It should be noted that if a received packet has experienced more errors than can be solved by the previous techniques, the code will return a decoded file with a null filename (full of zeros). This way the software can detect that a frame is not adequate and discard it.

Even though this part of the software is essential as an error correction tool, it presents a main disadvantage: it adds a large overhead to the transmission, which means that the transmitter will have to send a much higher amount of bytes than those truly containing the desired information. This overhead can be minimized by setting the file split size in multiples of $(223 - 8) + N * 223$ bytes , where N ranges from 0, 1, 2, ... This way the padding is null.

The following table depicts the overhead for different packet sizes:

Table 5.1: Downlink transmission overhead due to channel encoding

Packet size	**Encoded packet size**	**Overhead**	**Overhead percentage**
215 Bytes	512 Bytes	297 Bytes	58
438 Bytes	1022 Bytes	584 Bytes	57
884 Bytes	1532 Bytes	648 Bytes	42

Even though the overhead is very high, it should not be forgotten that that the data contained in the packet file will have been previously compressed, meaning that it contains more information bytes than the ones present.

In addition, the packet files size has been left as a variable, and can be changed at any moment easily. As it can be seen, the overhead slightly decreases the bigger the size is, due to the header bytes. This would indicate that the bigger the packet file, the more efficient the transmission. Nevertheless, the bigger the packet file, the more probability of errors occurring in the transmission. Therefore, the size of the packet files should be chosen after some testing of the whole transmission system in order to define the best size, also taking into consideration the real amount of data that the communications link will have to transmit.

5.4.6.2 Uplink

The uplink system works very similarly to the downlink with the exception that no header must be added, since the telemetry messages contain all the required information themselves. Moreover, when decoded, the final result will be a command to be done by the satellite, and will not be stored as is done in the downlink. This is depicted in the following figure:

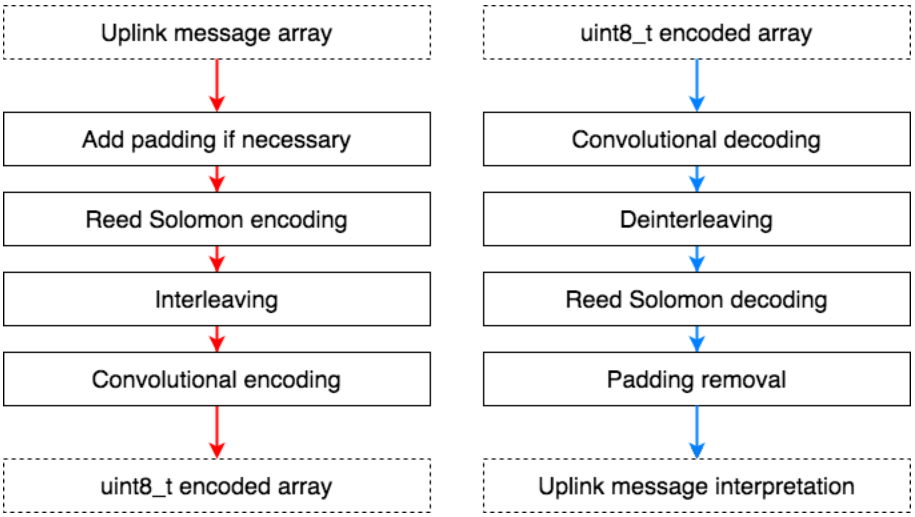


Figure 5.23: Uplink channel coding code block diagram

5.4.7 Code implementation

In order to implement the functions explained above, several files have been designed with appropriate code to carry them out. A detailed explanation can be found in appendix D.2.

5.5 Physical layer

As it is known, a radio frequency signal is an electromagnetic wave that communications systems use in order to transport information from two different points through air.

The physical layer is the part of the code that is in charge of transforming bits to an RF signal and recovering the bits from such a signal. Therefore, this section of the project depicts the design that has been implemented in order to deal with the transmission and reception of packets. It should be noted that this part of the software will work in tandem with a Software Defined Radio.

In order to carry out a correct design, the physical layer must first be understood correctly. This is detailed in the following subsection.

5.5.1 Physical layer architecture

The design of the physical channel must be different for transmission of the data and its reception.

In the transmission of bits, the data must be modulated in order to get a signal, then up-converted to the S-Band frequency and sent through the antenna, the up-conversion being necessary because of the lack of ADC/DAC functioning at the link's frequency. Analogously, on the reception side, the antenna will capture a signal, that has to be down-converted and later demodulated in order to obtain the data in bits.

Nevertheless, in order to be able to receive and interpret correctly the signal, more steps must be made.

The transmission block diagram is depicted below in Figure 5.24. As it can be seen, the block will take as an input an array of bits that have to be transmitted. These will be treated with the following steps:

- **Addition of a synchronization word**: This is necessary in order to be able to synchronize the signal at reception and detect the start of a packet. The synchronization functioning is detailed in subsection 5.5.5.
- **Modulation**: The modulation block is in charge of the conversion of bits (ones and zeros) into a signal. This is detailed in subsection 5.5.2.
- **Conformation Pulse**: In order to be able to sample correctly the symbols at reception, it is very important that the modulated signal is multiplied by a train of pulses. This block is explained with detail in subsection5.5.3.
- **Up-Conversion**: At this point of the chain, the code has the samples of a baseband signal. This has to be up-converted in order to be transmitted at the S-Band frequency. This up-conversion block is detailed in subsection 5.5.4.
- **Transmission**: The transmission is the process of sending the generated signal via an antenna. This is further detailed in subsection 5.5.7.

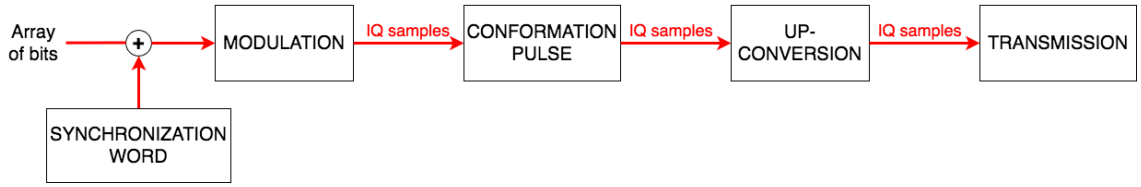


Figure 5.24: Physical channel transmission block diagram

Analogously, the reception block diagram is depicted below in Figure 5.25. The block will start with the reception of a signal that has to be processed and transformed into an array of bits. These will be treated with the following steps:

- **Reception**: The reception block is in charge of receiving a signal through an antenna. This is further detailed in subsection 5.5.7.
- **Down-Conversion**: The received signal is at a very high frequency, S-Band. Carrying out a demodulation at this frequency would require an incredible amount of efficiency and resources that make it impossible for the physical components of the design to handle. Therefore, the signal must be converted back into baseband to be processed. This process is explained in subsection 5.5.4.
- **Synchronization**: When the reception is activated, the antenna receives constantly, whether there is a signal present or only noise or interference's. Therefore, the receptor must have s system implemented to detect when the desired signal is present

and when this starts. This process is called synchronization and is further explained in subsection 5.5.5.

- **Tracking**: As has been previously explained, when dealing with satellite communications, the received signal experiences a Doppler shift in frequency. This results in the received and down-converted signal not being in baseband. Therefore, the receptor must implement a system to track the unknown frequency shift of the incoming signal to further down-convert the signal. There are several ways the tracking can be done, this project will use Phase Locked Loops. These are be detailed in subsection 5.5.6.

- **Sampling**: Once the baseband signal has been obtained, the sampling block is necessary to retrieve the samples from the conformation pulses that they have been modeled with in the transmission.

- **Demodulation**: The demodulation block is in charge of the conversion of samples of a signal into information, bits (one's and zero's). This is detailed in subsection 5.5.2.

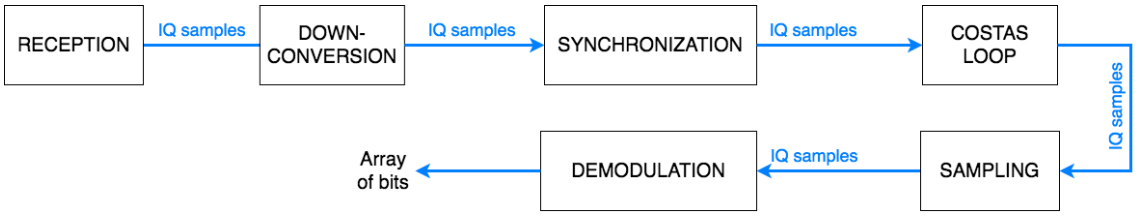


Figure 5.25: Physical channel reception block diagram

As it will be seen later on, the up-conversion and the transmission, as well as the reception and down-conversion and of the system will be carried out together in a SDR.

5.5.2 Modulation and Demodulation

Modulation is the technique used to convey data by changing one or several properties of a carrier signal (a periodic waveform). The resulting signal is referred to as the modulated signal. Demodulation consists on the inverse technique and allows to extract from a modulated signal its initial signal [71].

There are several types of modulations:

- **Amplitude-Shift Keying (ASK) :** This technique varies the amplitude of the signal proportionately to that of the modulating signal. The resulting modulated signal is expressed as:

$$v_c(t) = (A_c + A_m sin(w_m t))\, sin(W_c t + \theta) \tag{5.2}$$

where A_c is the amplitude of the carrier signal, A_m the amplitude of the modulating signal, w_c is the frequency of the carrier, w_m the frequency of the modulating signal.

- **Frequency-Shift Keying (FSK):** This technique varies the instantaneous frequency of the wave. The modulated signal can be expressed as:

$$v_c(t) = A_c sin\left[2\pi\left(f_c + \Delta f sin(2\pi f_m t)\right)t\right] \tag{5.3}$$

  where A_c is the amplitude of the carrier signal, f_c is the frequency of the carrier, f_m the frequency of the modulating signal and Δf is the frequency deviation, which is represents the maximum change in frequency that the modulated signal can undergo.

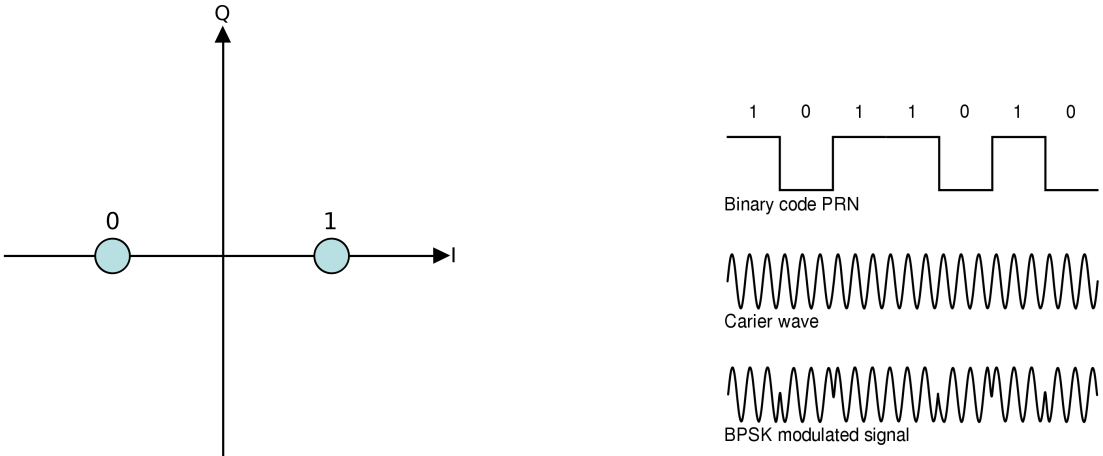
- **Phase-Shift Keying (PSK):** This technique encodes a message by using instantaneous phase variations in a carrier wave.

In order to determine which is the most adequate modulation for the communications system, the overall system must be taken into account. CubeSat communications are being developed with PSK, and in this case, Binary Phase-shift keying (BPSK) is used.

5.5.2.1 BPSK

Binary Phase Shift Keying, is a form on phase modulation that consists on varying the phase of the carrier signal between a discrete range of values. As its name indicates this is a binary shift, therefore, it uses two symbols to represent the signal.

Figure 5.26(a) represents a BPSK constellation. and Figure 5.26(b) depicts how the modulated signal presents a 180º shift in its phase when the bits of the code to be transmitted change.



(a) Constellation [72]

(b) Modulation [73]

Figure 5.26: BPSK Modulation

This modulation is appropriate for the communications link design since it is appropriate for long range transmissions. This is due to the fact that it requires less SNR than other PSK modulations for a given BER. This is because the transmitter sends only the ”in-phase” component of the signal, which results in symbols having higher energy than for modulations with both ”in-phase” and ”quadrature” components.

5.5.3 Conformation pulse

In order to send a signal, it is imperative to use a conformation pulse in order to ensure a correct reception at the end of the communications link. Nevertheless, not any conformation pulse can be used. In fact, this should be adequate to reduce as much as possible inter-symbol interference's and fulfill the Nyquist criteria.

The Nyquist ISI criteria describes the conditions for which a communication channel does not present inter-symbol interference's. These are created by the fact that when consecutive symbols are transmitted in a channel with a linear modulation, the frequency response of the channel causes the transmitted symbol to spread in the time domain. This leads to symbols interfering with each other.

In order to ensure that the response will not present ISI, the channel impulse response must fulfill the condition expressed in equation 5.4, which is equivalent to equation 5.4, the Nyquist ISI criterion.

$$h(nT_s) = \begin{cases} 1; & n = 0 \\ 0; & n \neq 0 \end{cases} \tag{5.4}$$

$$\frac{1}{T_s} \sum_{k=-\infty}^{+\infty} H\left(f - \frac{k}{T_s}\right) = 1 \quad \forall f \tag{5.5}$$

The raised cosine response meets this criterion, making it an adequate pulse to use. This can be seen in Figure 5.27, where a train of consecutive pulses all present zero values in the same coinciding locations.

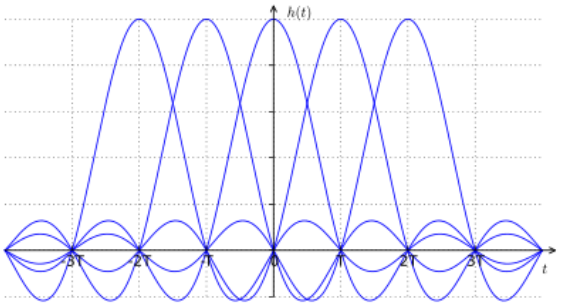


Figure 5.27: Raised cosine response [74]

In order to carry out this implementation, the Square-Root Raised Cosine Signals (SRRC) has been chosen. This waveform satisfies the previous criteria. Moreover, the BER is identical to that of a BPSK with NRZ pulses if the receiver samples at zero ISI locations.

The pulse has been chosen with a roll off factor of 0.35. This value is big enough that it presents a slim pulse, and small enough that it fulfills the bandwidth conditions of the channel.

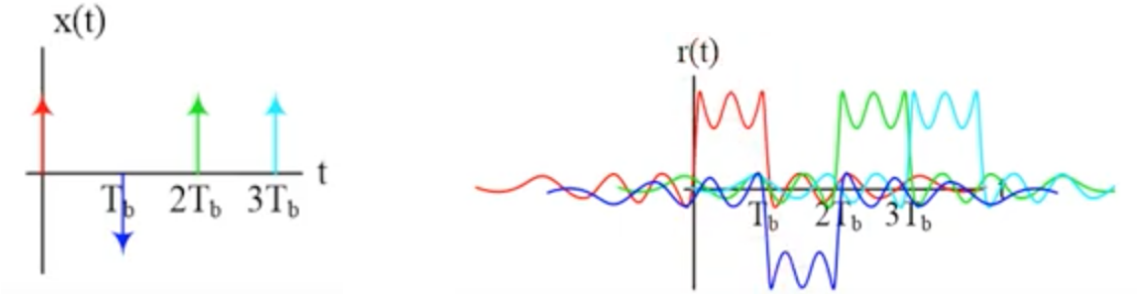
The square-root-raised cosine's power spectral density is given by the following equation:

$$G_{\mathrm{SRRC}}(\alpha, f) = \begin{cases} T_c & \|f\| \leq \frac{1-\alpha}{2T_c} \\ \frac{T_c}{2}\left\{1+\cos\left[\frac{\pi T_c}{\alpha}\left(\|f\|-\frac{1-\alpha}{2T_c}\right)\right]\right\} & \frac{1-\alpha}{T_c} \leq \|f\| \leq \frac{1+\alpha}{2T_c} \\ 0 & \|f\| > \frac{1+\alpha}{2T_c} \end{cases} \tag{5.6}$$

The time representation of SRRC pulses adopts the form depicted in equation 5.7.

$$s(t) = \frac{4\alpha}{\pi\sqrt{T_c}} \frac{\cos\left[\frac{(1+\alpha)}{T_c}\pi t\right] + \frac{T_c}{4\alpha t}\sin\left[\frac{(1-\alpha)}{T_c}\pi t\right]}{1-\left(\frac{4\alpha t}{T_c}\right)^2} \tag{5.7}$$

Figures 5.28 clearly represent how a train of pulses will look in the time domain after using a train of conformation pulses.



(a) Signal pulses [75]

(b) Pulse shaped signal [75]

Figure 5.28: Pulse shaping

5.5.4 Up and Down-Conversion

Up-conversion is a technique used to transform information contained in a low frequency into a higher frequency signal. Analogously, down-conversion consists on the inverse technique, allowing to recover the initial low frequency signal.

5.5.4.1 Introduction to I/Q signals

In order to understand the process on up and down conversion, some basic concepts regarding signals must be understood.

A pass-band signal can be seen in equation 5.8, where $a(t)$ is the envelope of the signal, f_c is the carrier signal and $\phi(t)$ is the phase of the signal.

$$s(t) = a(t)\cos\left(2pif_ct + \phi(t)\right) \tag{5.8}$$

The term "I/Q" stands for "in-phase" and "quadrature". These refer to two sinusoids that present the same frequency but a phase shift of 90º, being by convention "I" a cosine and "Q" a sine.

These emerge from the following process:

1) Equation 5.8 can be expressed as Equation 5.10 by using the Equation 5.9:

$$\cos(a+b) = \cos a \cos b - \sin a \sin b \tag{5.9}$$

$$s(t) = a(t)\left[\cos\left(\phi(t)\right)\cos\left(2pif_c t\right) - \sin\left(\phi(t)\right)\sin\left(2pif_c t\right)\right] \tag{5.10}$$

1) Considering that the in-phase component can be defined as $s_I = a(t)\cos\left(\phi(t)\right)$ and the quadrature component as $s_Q = a(t)\sin\left(\phi(t)\right)$ the signal can be expressed as:

$$s(t) = s_I(t)\cos\left(2pif_c t\right) - s_Q(t)\sin\left(2pif_c t\right) \tag{5.11}$$

By using the I/Q components, two baseband signals can be sent instead of one. Additionally, since the difference in phase between both phases is known, phase shifts can be detected.

Another important concept is that of the complex envelope of a signal. This complex tool contains all the information of a pass-band signal except that of the carrier frequency (f_c). This can be expressed with the following Equation 5.12.

$$\tilde{s}(t) = s_I(t) + js_Q(t) = a(t)e^{j\phi(t)} \tag{5.12}$$

5.5.4.2 Schematics

In order to carry out the up-conversion of a signal, an I/Q modulator is used. This device allows to modulate two baseband signals with the same carrier frequency, one in-phase and another in quadrature. This is done using the schematic depicted in Figure 5.29.

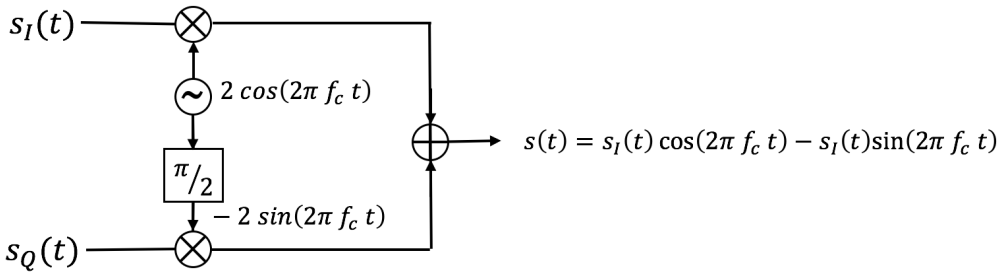


Figure 5.29: I/Q Modulator

While the previous figure depicts the required blocks for a hardware implementation, this same schematic can be simplified into the one depicted in Figure 5.30, which is better suited for a digital implementation.

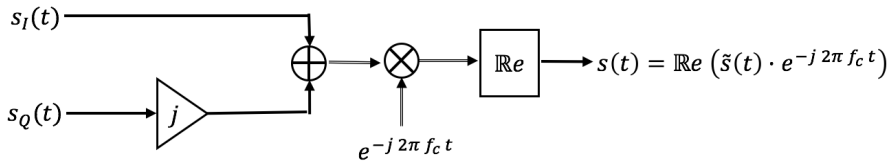


Figure 5.30: I/Q digital modulator

Analogously, the down conversion is done with an I/Q demodulator, which allows to recover the $s_I(t)$ and $s_Q(t)$ components from $s(t)$. This can be done both with the hardware model implementation depicted in Figure 5.31 and for the simplified model for a digital implementation in Figure 5.32.

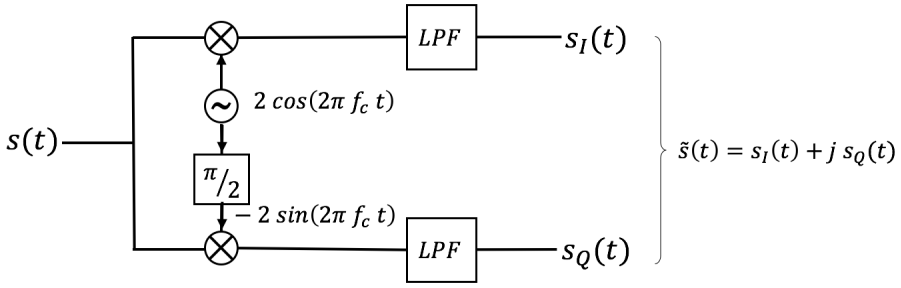


Figure 5.31: I/Q demodulator

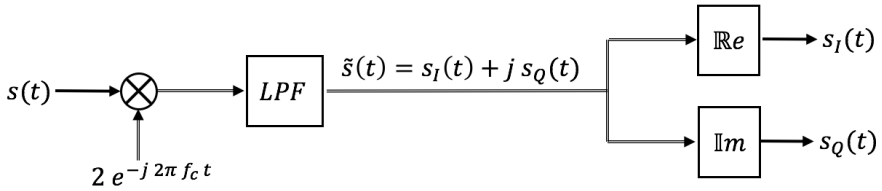


Figure 5.32: I/Q digital demodulator

It should be noted that the up and down-conversion of the signal is implemented in the ADALM-PLUTO SDR and therefore has not been included in the code.

5.5.5 Synchronization

In order to detect the start of a packet, it is imperative to synchronize the signal. Synchronization is the process with which a signal is detected amongst other signals or its surrounding noise. This process can be done both in the time domain or in the frequency domain. Nevertheless, this second case is much more efficient and the one commonly used in radio frequency communications systems.

In order to synchronize a signal, a synchronization word is added at the beginning of the packet to be transmitted. This word is known by the receiver and will be used to detect the packet. For this project the chosen synchronization word will be the one used in the CCSDS standard: 0x1ACFFC1C, which can also be written as 449838109 [76].

In order to detect this preamble or synchronization word, the software must carry out a correlation operation.

5.5.5.1 Correlation

Convolution, which is the basis of many signal processing techniques, can be defined as the mathematical method of combining two signals in order to form a third. Equation 5.13

shows the convolution of two continuous time signals $x(t)$ and $h(t)$ [77].

$$y(t) = x(t) \otimes h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \tag{5.13}$$

Correlation is a convolution operation that is used to compare signals. In this case, the convolution is done between a signal and the functional inverse of another signal. The resulting signal of the operation is know as the cross-correlation of the two input signals. The amplitude of the cross-correlation is a direct indicator of the resemblance between the two given signals, and its peak defines the location of the target. The correlation mathematical expression is shown in equation 5.14. [77]

$$c(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau = x(t) \otimes h(-t) \tag{5.14}$$

Using the correlation equation in the frequency domain, with the use of Fourier Transforms, the software is able to compute the correlation peak and detect packet's preamble.

5.5.6 Signal Tracking

Tracking a signal correctly is extremely important in satellite communications since in a coherent receptor, such as the one that has been designed, there cannot be a frequency nor phase shift between the transmitter and receiver for the data to be detected appropriately. Therefore it is imperative for this design that the Doppler frequency shift is corrected.

There are three methods that can be employed, which are as follows:

- Non-Data-Aided (NDA): These work without any knowledge of the modulation symbols, only using the incoming signal, and are commonly used in simple simple carrier recovery schemes.
- Data-Aided (DA): These methods work under the premise that the system has knowledge of transmitted symbols, for instance a preamble.
- Decision Directed (DD): These are used when the symbols are synchronized before the carrier is recovered. The output of the symbol decoder is introduced into a comparison circuit that comparing the phase between the received signal and the decoded symbol to manage the local oscillator. In particular, these methods are commonly used to synchronize frequency differences that are smaller than the symbol rate.

The system has been designed with a NDA method. There are a number of techniques that can be used in order to estimate this shift, one of which is through a Phase Locked Loop (PLL).

5.5.6.1 Phase Locked Loop

A PLL is a device that generates a clock and synchronizes it with an input signal. Its most relevant application in communication systems is the recovery of the transmitter clock,

since this is often required to time appropriately the processing of the incoming signal. This can be done through hardware or via software. In this project, its implementation will be digital.

Digital PLL's [78] are used for digital signal synchronization and present the components shown in Figure 5.33.

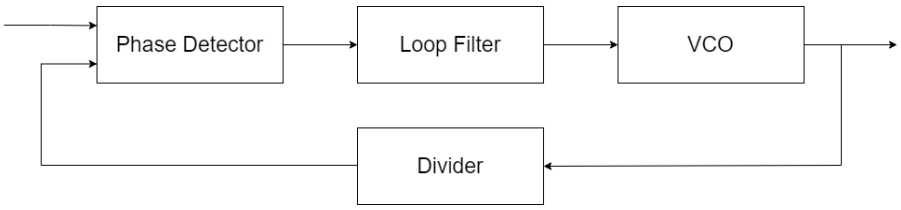


Figure 5.33: PLL Block Diagram

- **Phase Detector**: This block measures the differences in phase between input and output signal in order to correct the phase difference.
- **Loop Filter**: This filter is used to translate between the measurement of the phase detector and the Voltage Controlled Oscillator's control voltages, speeding it up or slowing it down as necessary.
- **Voltage Controlled Oscillator (VCO)**: This block generates a digital clock whose frequency is controlled by one or more voltage inputs.
- **Divider**: This performs a frequency division on the output signal in order to obtain the input's frequency with the output's phase.

The first signal tracking implementation was carried out with a PLL together with a Frequency Locked Loop. FLL are placed on the receiver's signal tracking loops in order to provide continuous frequency corrections. These are used specifically to track the Doppler frequency of the incoming signal without correcting the phase. The joined use of these locked loops provides an excellent tracking system, since the FLL is a fast estimator of the Doppler frequency and the PLL can fine tune the previous approximation.

Nevertheless, this is not the most efficient design for the systems requirements. Therefore, another tracker was implemented, using a Costas Loop.

5.5.6.2 Costas Loop

A Costas loop is a PLL circuit used for carrier frequency recovery of suppressed carrier modulation and phase modulation signals. It's main us is in wireless receivers and it presents much more sensitivity to Doppler shifts than other models, making it excellently suited for GPS receivers.

As it has been previously stated, BPSK is one of the most efficient modulations when it comes to noise immunity per unit bandwidth. Nevertheless, its demodulation is much trickier than that of a Frequency Shift Keying (FSK), and the most appropriate design for its receiver design is to implement a Costas-loop to perform both phase tracking and data demodulation [79].

The following diagram, shown in Figure 5.34, depicts the typical schematic of the Costas Loop for the BPSK case.

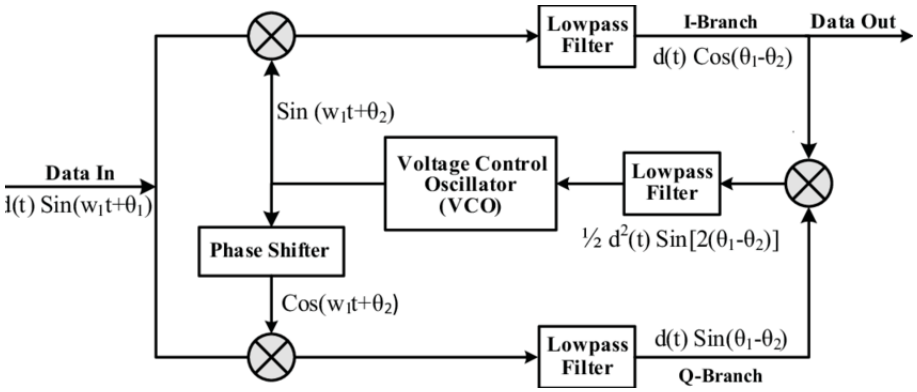


Figure 5.34: Costas Loop Block Diagram [80]

There are several discriminator algorithms that are appropriate for this configuration. These are depicted in table 5.2.

Table 5.2: Costas Loop Discriminators [81]

Discriminator Algorithm	Output Frequency Error	Computational burden
$Q_{PS} * I_{PS}$	$sin(2\phi)$	Moderate
$Q_{PS} * Sign(I_{PS})$	$sin(\phi)$	Least
$\frac{Q_{PS}}{I_{PS}}$	$tan(\phi)$	High
$atan(\frac{Q_{PS}}{I_{PS}})$	ϕ	Highest

It should be noted, that even though it may appear as a very simple system, the Costas Loop has several details that if unknown, make its implementation particularly difficult. For instance, during its implementation, it was noticed that if the frequency at which the bits were modulated is higher than the carrier frequency (in this case the Doppler frequency), its typical design does not work.

After much research to fully understand the functioning of the Costas Loop, a feasible design for this scenario has been devised. The following schematic depicts the structure of the Costas Loop implementation.

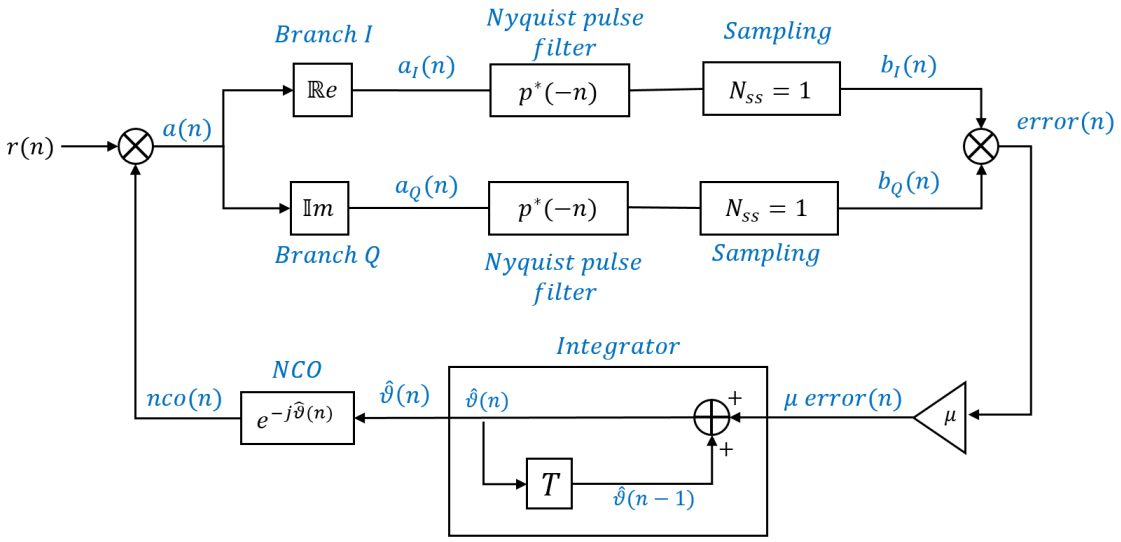


Figure 5.35: Costas Loop Schematic

As it can be seen, the signal $r(n)$ enters the loop and is multiplied by the NCO. The resulting signal $a(n)$ is expressed in equation 5.15, where A_o represents the amplitude of the signal, a_k the symbol, N_{ss} the number of symbols, $\theta(n)$ the phase of the input signal and $\theta(n)$ the phase of the NCO.

$$a(n) = A_o * \sum_k a_k \phi(n - N_{ss}k) * e^{j[\theta(n) - \hat{\theta}(n)]} \tag{5.15}$$

This signal is divided into its "in-phase" and "quadrature" components on two separate branches. The resulting $a_I(n)$ and $a_Q(n)$ undergo a Nyquist pulse filter and are later sampled. This sampling process is carried out due to the fact that the signal has been multiplied in the transmitter by a train of conformation pulses, therefore, several samples represent one symbol. The sampling will extract the symbol from its respective samples. At the end of this process the signal is represented by $b_I(n)$ and $b_Q(n)$, that are represented by Equation 5.16. It should be noted that $b_I(n)$ will be the resulting demodulated and sampled signal that will be used in the rest of the receiver chain.

$$\begin{aligned} b_I(n) &= A_o a_k \cos(\theta(n) - \hat{\theta}(n)) \\ b_Q(n) &= A_o a_k \sin(\theta(n) - \hat{\theta}(n)) \end{aligned} \tag{5.16}$$

In order to obtain the phase error, the two signals are multiplied, which as has been noted earlier, a discriminator algorithm with a moderate computational burden that returns a very small error in comparison to others. The result of this multiplication $error(n)$ is expressed in equation, where $\theta_e(n) = \theta(n) - \hat{\theta}(n)$.

$$error(n) = \frac{A_o}{2} * sin[2\theta_e(n)] \approx A_o \theta_e(n) \tag{5.17}$$

This error is then multiplied by a gain μ, which is a small value that helps to control the NCO. The higher the value, the faster the system will approximate the phase, but also the

more overshooting. And on the other hand, the smaller it is, the slower the system will approximate the phase error.

After going through the amplifier, the signal error enters an integrator, that helps the system to better control the phase error that is then passed to the NCO. This block produces an exponential wave that approximates the carrier frequency of the incoming signal through the error phase that it receives. This wave is multiplied by the incoming signal and the process is repeated again for the next incoming sample and continues until the last sample of the incoming signal.

As it can be seen, not only does this design eliminate the Doppler frequency shift of the signal, but samples and demodulates the signal at the same time, being a very efficient design.

5.5.7 Transmission and reception

The transmission and reception of the signal is coded using the iio library [82] to connect the software to the hardware. This is possible because this library is designed to interface with Linux IIO (Linux Industrial Input/Output) devices.

Moreover, in order to use this library in a more efficient manner, another library has been used: IIOHandler. This library has been developed in NanoSat Lab and has been used, with some modifications, in order to configure the physical receiver and transmitter more easily.

In order for the physical SDR device to transmit and receive the symbols, a specific configuration must be used, which has been included in the software. This is available in appendix E. Moreover, this appendix also contains the code corresponding to the connection between software and physical device for the reception and transmission.

5.5.8 Functioning of the physical channel code

Having defined all the steps that the physical channel must undergo, the software can be designed accordingly. The hardware that will be used, which is explained in further detail in 6.1.1 already has implemented the functions of transmission, reception, up and down-conversion, low pass filtering and DAC and ADC. Knowing this, the software has been designed to carry out the functions shown in the schematic below. It should be noted that the Costas Loop design implemented also carries out the sampling of the signal.

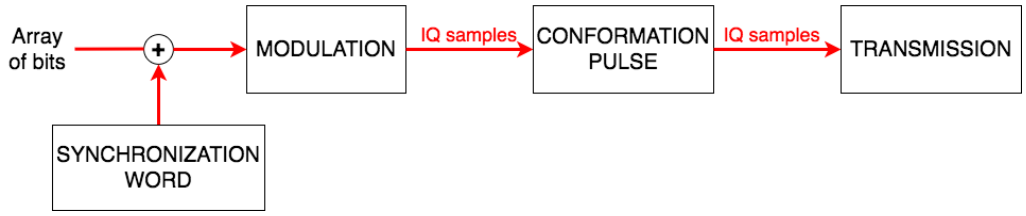


Figure 5.36: Block diagram of the transmission software

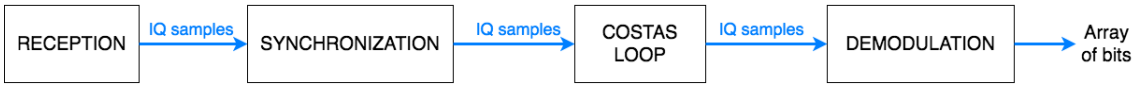


Figure 5.37: Block diagram of the reception software

5.5.9 Code implementation

In order to implement the functions explained above, several files have been designed with appropriate code to carry them out. These are explained in appendix D.3, that depicts all the files used and their respective functions. Moreover, it also indicates the external libraries used for this implementation.

CHAPTER 6. TESTING AND VERIFICATION

This chapter elaborates on the device used for testing and the tests that have been carried out to check the correct functioning of the code.

6.1 Testing interface

Both RITA payload and Montsec ground station will communicate through a Software Defined Radio (SDR) connected to the respective antenna. In order to test the functioning of the code, two SDR devices have been used.

A SDR is a radio in which signal processing tasks that are commonly processed by hardware are carried out using software or firmware. It should be noted that this chip is controlled by an FPGA.

An SDR is a radio communication system that has been implemented in a software on a computer or an embedded system. Therefore, it allows to carry out many functions that traditionally required hardware (such as mixers, filters and modulators amongst others) by software.

6.1.1 ADALM-PLUTO SDR

In order to test the functioning of the code, a ADALM-PLUTO Software-Defined Radio Active Learning Module has been used.

This module has the following characteristics that make it appropriate to use in this project [83]:

- It's portable, which makes it easier to work with.
- It's based on Analog Devices AD9363, highly integrated RF Agile Transceiver and Xilinx Zynq Z-7010 FPGA. Since RITA works with a very similar FPGA and SDR pair, this is an appropriate testing environment.
- It has an RF coverage from 325 MHz to 3.8 GHz, which includes the S-Band downlink frequency (2.279 GHz) and the uplink frequency (2.028 GHz).
- Its bandwidth can be up to 20 MHz, which includes the bandwidth to be used (2.4MHz).
- It has a 12-bit ADC and DAC .
- It has a receiver and a transmitter chain, half or full duplex, which allows to do all the necessary testing.
- It has a USB 2-0 Interface powered with a Micro-USB 2.0 connector, that can be easily connected to any computer.
- Can run software cross-compiled for ARM and written in C++.

As stated, this SDR will run the code. It should be noted that each device has a reception interface and a transmission interface, making it possible to receive and transmit in the satellite side and the ground station side. To better understand the functioning of the device, the block diagram shown in the following figures can be used. Each represents the specific blocks of the ADALM-PLUTO device used for transmitting (Figure 6.1) and receiving (Figure 6.2).

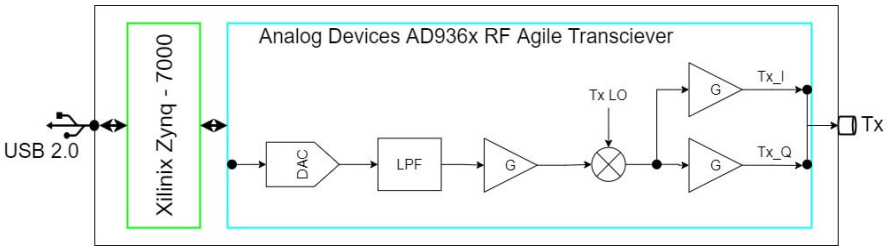


Figure 6.1: ADALM-PLUTO Transmitter block diagram

In the transmitting scenario, the code is run in the Xilinix module, that controls the AD9363. In order to transmit, the digital signal passes through a Digital to Analog Converter (DAC), a low pass filter, an amplifier, and later on, the analog signal is mixed with a local oscillator. The function of this is to convert the base-band signal resulting from the DAC to a pass-band signal in a much higher frequency, which is the S-Band frequency for this communication scenario. Then, the resulting signal is modulated in phase and quadrature, and transmitted through the transmission port.

The reception side works very similarly to the transmission. The main difference being that in this case, the reception antenna detects band-pass signals, that are down-converted to baseband with the LO mixer. Later on, the baseband signal is filtered with a low pass filter and converted into digital with the Analog to Digital Converter (ADC).

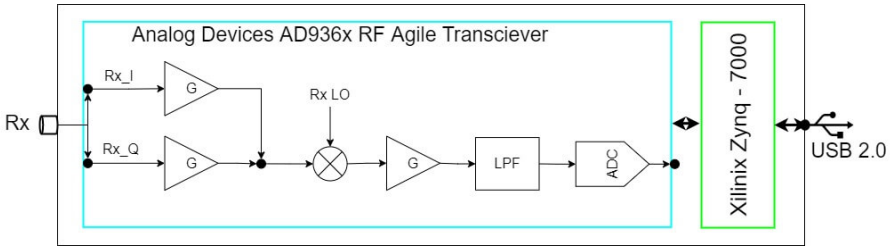


Figure 6.2: ADALM-PLUTO Receiver block diagram

6.1.2 Compilation in SDR

In order to be able to run the code in the ADALM-PLUTO, there are several steps that must be made. The most important being ”Cross Compilation”, which is the act of compiling a code for another processor architecture, in this case, the ARM® Cortex™-A9 MPCore [84]. This is a very useful and common technique when working with embedded systems. Appendix C.2 ”Guide to run the project in ADALM-PLUTO contains a thorough description of the steps that have to be followed to cross compile the project and be able to run it in the SDR.

6.2 Application layer testing

A correct implementation of the files management is crucial to ensure that no information is lost in the process and no files are forgotten. This section carries out several tests to verify that the system works as designed.

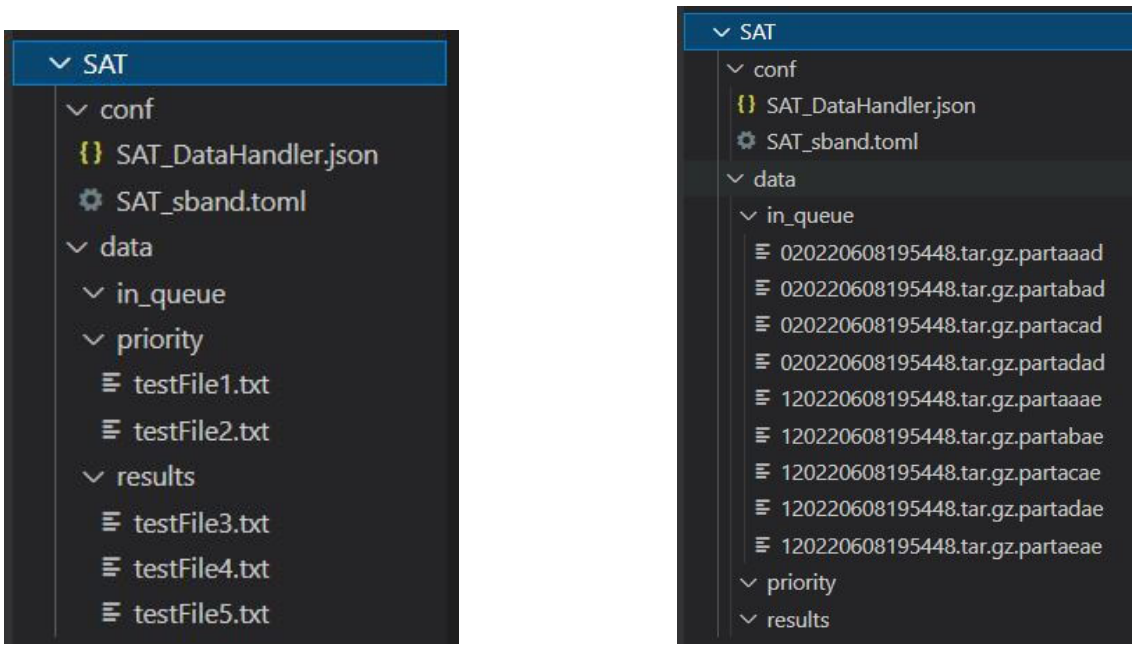
6.2.1 RITA downlink files preparation

The first test to be carried out is regarding the preparation of the files for the downlink transmission.

As it has been explained in the implementation section, the satellite will have a data folder and a configuration folder. The main code of the satellite will have access to the data folder, and will save the files to be transmitted in the folders ”results” and ”priority”. In order to simulate a real case of file preparation, several test files have been saved in these folders: two files in the priority folder and three in the results folder. This can be seen in Figure 6.3(a), that depicts the initial disposition of the files.

To activate the test, the command ”rita-comms -p” is used. This uses flag ”p”, that stands for preparation of the files.

In order to determine that the downlink files preparation has worked correctly, at the end of the process, the files from the folders should have been split and renamed with the timestamp and the priority flag. Moreover, these packet files should only be present inside the ”in_queue” folder in split tar.gz files indicating their split contents.



(a) Initial files disposition

(b) Final files disposition

Figure 6.3: File management in the satellite

Figure 6.3(b) depicts the disposition of the files after the process has been carried out. As it can be seen, the final files present the expected filename, with the priority flag and the

timestamp of the moment of their creation (the 8th of June 2022 at 19 hours, 54 minutes and 48 seconds). Moreover, they are located at the ”in_queue” folder as expected, and the initial files have been erased from the ”results” and ”priority” folders.

The following code depicts the *SAT_DataHandler.json* file. This data handler should keep track of all the files that are present in the folder ”in_queue” to be sent. As it can be seen, this field indeed contains all the files that have been created with the characteristic ”to send”. With these results, it can be concluded that this system is tested and verified.

```
{
    "020220608195448.tar.gz.partaaad": "to send",
    "020220608195448.tar.gz.partabad": "to send",
    "020220608195448.tar.gz.partacad": "to send",
    "020220608195448.tar.gz.partadad": "to send",
    "120220608195448.tar.gz.partaaae": "to send",
    "120220608195448.tar.gz.partabae": "to send",
    "120220608195448.tar.gz.partacae": "to send",
    "120220608195448.tar.gz.partadae": "to send",
    "120220608195448.tar.gz.partaeae": "to send"
}
```

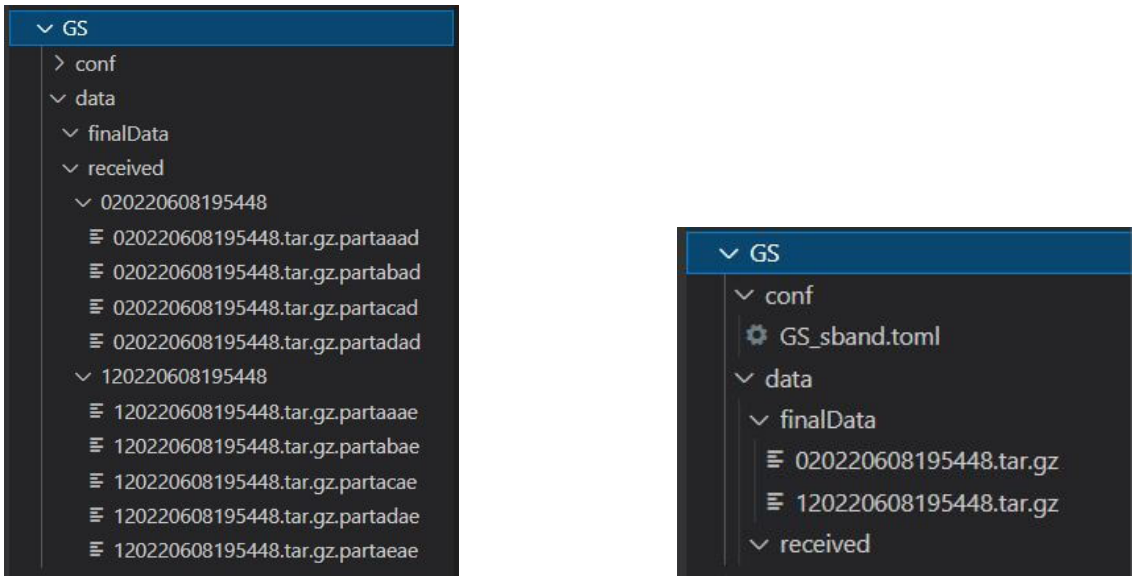
6.2.2 Ground Station downlink files management

Having tested the file management in the satellite, the next step is to test the functioning of the ground station implementation.

In the same way that the satellite must manage the files to be sent, the ground station must manage the packets received. This test is designed to verify that the system is able to extract the initial information from the packets received.

It should be noted that this test is carried out from the previously obtained packet files, assuming that the ground station code has carried out the packet reception and the packets have been decoded and saved. Figure 6.4(a) depicts the disposition in which the received packet files would be after a downlink reception.

In order to determine that the received files have been managed correctly, at the end of the process, the packet files in the *received* folder should have been deleted and the original compressed files of information should be located in the *finalData* folder. As it can be seen in Figure 6.4(b), this is the case, since there are two files present in the *finalData* folder: ”020220608195448.tar.gz”, containing the information of the priority folder from the satellite, and ”120220608195448.tar.gz”, containing the information of the results folder from the satellite.



(a) Initial files disposition (b) Final files disposition

Figure 6.4: File management in the ground station

6.2.2.1 Results and conclusions

In order to check that the files have been joined correctly and that the compressed files handed to the user are correct, the files have been decompressed using the following commands ”tar -xvf 020220608195448.tar.gz” and ”tar -xvf 120220608195448.tar.gz in the *finalData* folder.

Figure 6.5 shows how the files compressed in the satellite correspond to the ones obtained with this command. Therefore, the correct functioning of the system is verified.

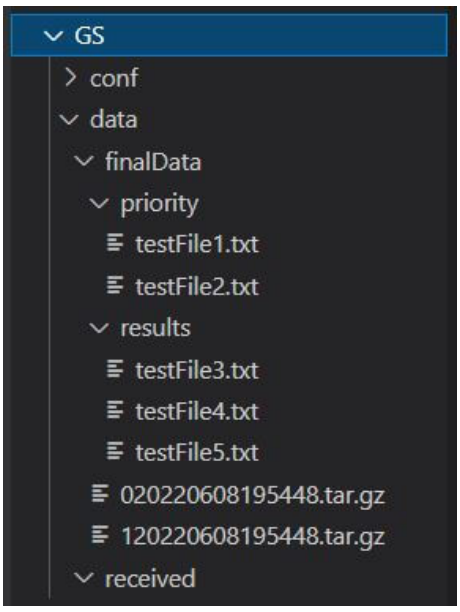


Figure 6.5: Decompressed received files in the ground station

It should be noted that if this process is robust. The code checks that no errors have

occurred in any commands before erasing the previous file in order to make sure no information is lost. For instance, if an error occurred when joining the files of a folder and the resulting joined file was damaged, this file would be erased and the command carried out again.

6.3 Channel Coding testing

It is of utmost importance that the encoding and decoding of the frames that are sent and received work correctly and are able to correct possible errors that might occur in the physical channel. For this reason, several tests have been carried out.

6.3.1 Basic encoding and decoding test

This test has been designed to verify that after a message is encoded and decoded, the original information can be recovered correctly. The test consists of the following steps:

1. An array of 200 bytes is created and assigned to a specific filename.
2. The array and filename are entered into the encoding process, where the following steps are carried out:
   (a) Padding addition
   (b) Header addition
   (c) Reed Solomon encoding
   (d) Interleaving
   (e) Convolutional encoding
3. The physical transmission and reception is simulated by the addition of errors in the encoded message.
4. The encoded message with errors is entered into the decoding process, where the following steps are carried out:
   (a) Convolutional decoding
   (b) De-interleaving
   (c) Reed Solomon decoding
   (d) Header decoding
   (e) Padding removal
5. The original message and filename are recovered

6.3.1.1 Results and conclusions

The chosen array of bytes for the test and the filename are depicted in the following figure:

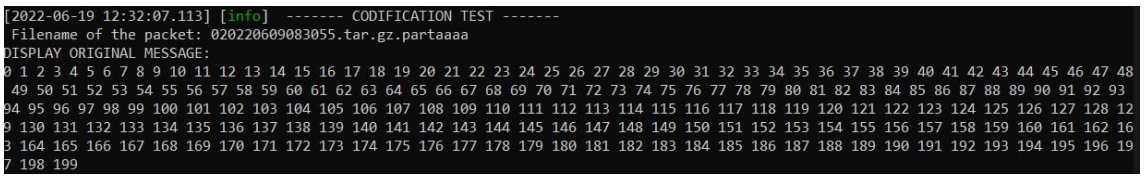


Figure 6.6: Initial test filename and message displayed in console

This test has been carried out for several types of errors, to ensure that the process will be able to function in different cases. The tests and their results step by step are accessible in Appendix G. The obtained results verify the correct functioning of the encoding and decoding functions up to a number of errors. For instance, the test is able to recover the data perfectly for recurring byte errors every 20 and 10 bytes, but is unable to do so for errors every 5 bytes, as is to be expected, since that surpasses the correction capacity of the code. Moreover, the test is also able to recover errors simulating ”deep fading” up to a point.

As it can be seen, when there are too many errors for the system to correct, the recovered message will be an array of 0, indicating then that the message is too damaged to be used.

6.3.2 File recovery after errors test

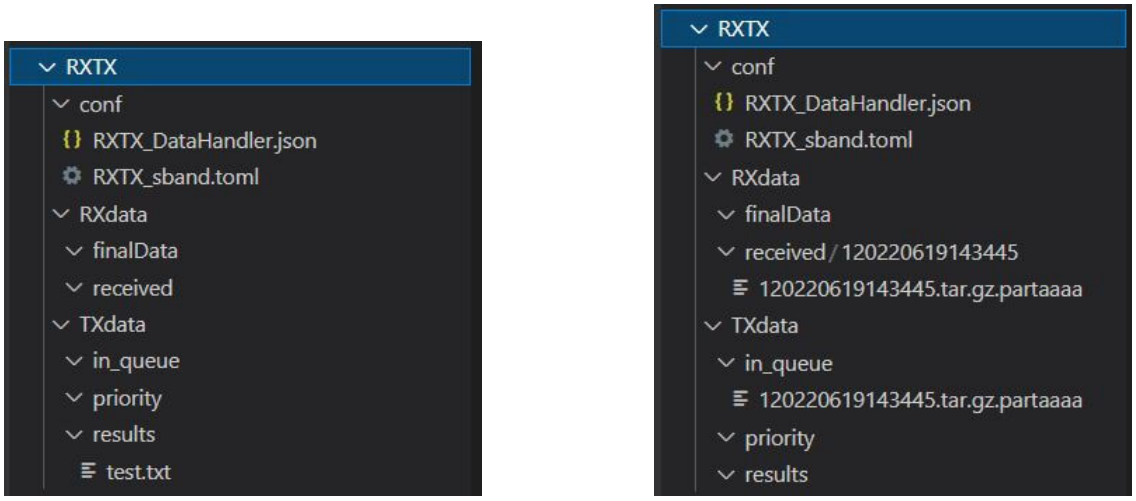
Having tested that the channel codification software is functional for a simple message, the next step is to test it with a file.

This test will simulate both transmitter and receiver scenarios. In order to do this, a specific folder has been created for the test. This contains a configuration folder, with the data handler JSON file and the TOML configuration file. Moreover, there are two data folders, one with the data folder distribution of the satellite (*TXdata*) and another with the data folder distribution of the ground station (*RXdata*). This setup can be seen in Figure 6.7.

This test consists of the following steps:

1. A file has been created and is saved in the *TXdata* folder, inside the results folder.
2. The application layer function to prepare the files is used and the stored file is processed and later converted into a packet file to be sent, located in the *in_queue* folder.
3. The packet file is read, stored into an array and encoded.
4. The resulting array is modified with the addition of errors simulating those that might appear in the physical layer of the system.
5. The resulting array with errors is then decoded and the filename of the packet file is recovered.

6. The file management of the ground station is activated for the reception of files, and the recovered message is stored into a file with its corresponding filename within a folder also containing its filename.

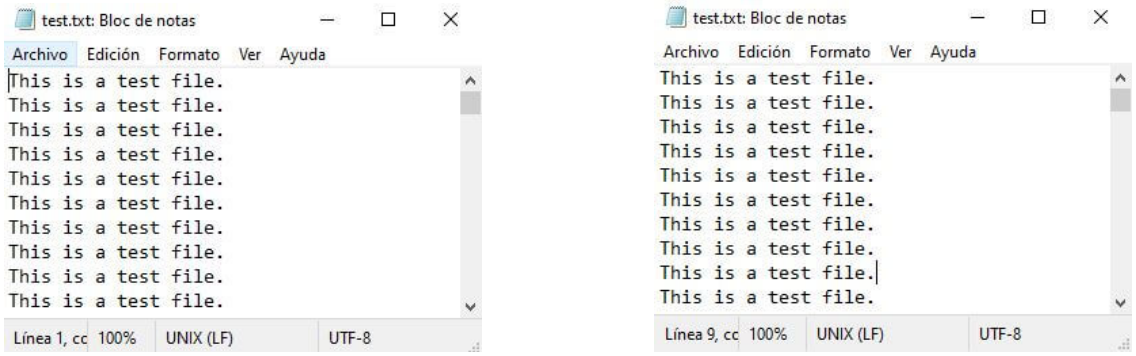


(a) Initial files disposition (b) Final files disposition

Figure 6.7: File distribution for the test

6.3.2.1 Results and conclusions

As it can be seen in Figure 6.7(b), the file has been recovered correctly, and the files have been managed as desired, verifying then all the aforementioned processes. Nevertheless, in order to check that the file has been indeed recovered correctly it must be decompressed and compared with the original file. The original and recovered files are depicted in the following figures.



(a) Original file (b) Recovered file

Figure 6.8: Initial and recovered file

As it can be seen, the files are identical, verifying that the whole process works correctly.

6.4 Physical channel testing

The correct union of software with hardware is one of the most important and trickiest parts of a project. For this reason, it is key to test the functioning of both.

6.4.1 Transmission test

The transmission test is done in order to check that the ADALM-PLUTO is transmitting the bits as expected. This test will consist of sending an array of bits and detecting it in a spectrum analyzer.

This is a very simple test and requires only of a few components:

- Spectrum analyzer. For this test, the Rode & Schwarz Spectrum Analyzer has been used. This equipment can measure signals from 9 kHz to 40 GHz, and is therefore appropriate for the S-Band signal.
- SDR device. As has been stated previously, this test will use the ADALM-PLUTO.
- Coaxial cable to join the transmitter port of the SDR with the RF input of the spectrum analyzer.

In order to carry out the test, after the code is written it must be cross-compiled in the ADALM-PLUTO, where it can be run. Figure 6.9 depicts the testing scenario, where the SDR contains the appropriate code.

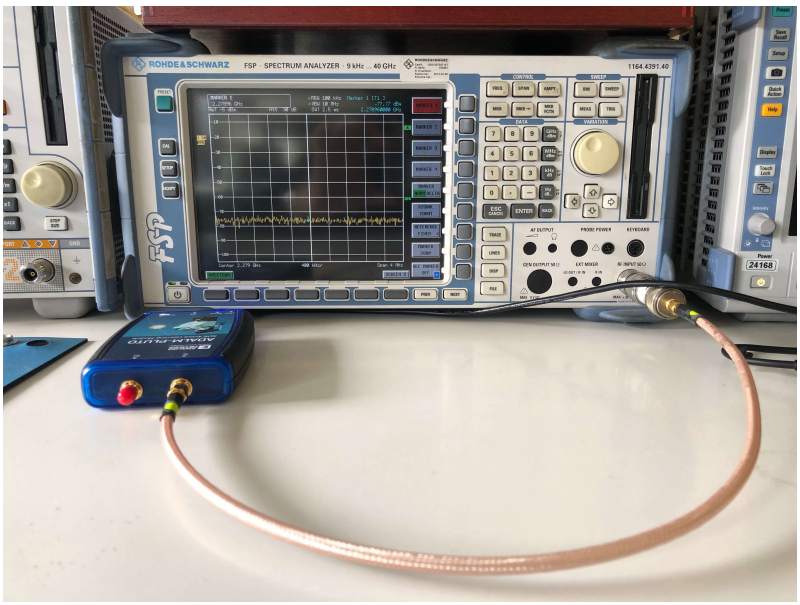


Figure 6.9: Transmission test setup

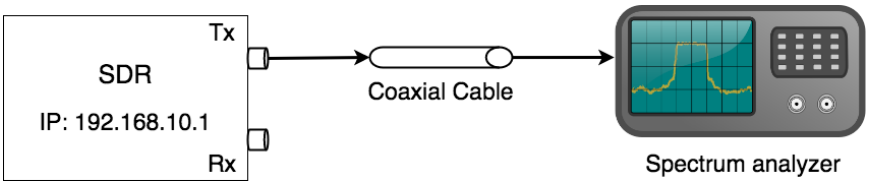


Figure 6.10: Transmission test setup schematic

6.4.1.1 Results and conclusions

The test has been configured to send a pulse at 2.279 GHz, which is the downlink S-Band central frequency. Therefore, should the transmission be correct, a pulse should appear at that frequency in the spectrum analyzer.

It should be taken into consideration that the ADALM-PLUTO has a local oscillator to elevate the transmitting signal to the appropriate frequency. In this case, the local oscillator has been set to 2.279GHz, and therefore, there will be a low peak of power in the frequency analyzer. This peak is depicted in Figure 6.11, and it should not be mistaken for the transmitted signal.

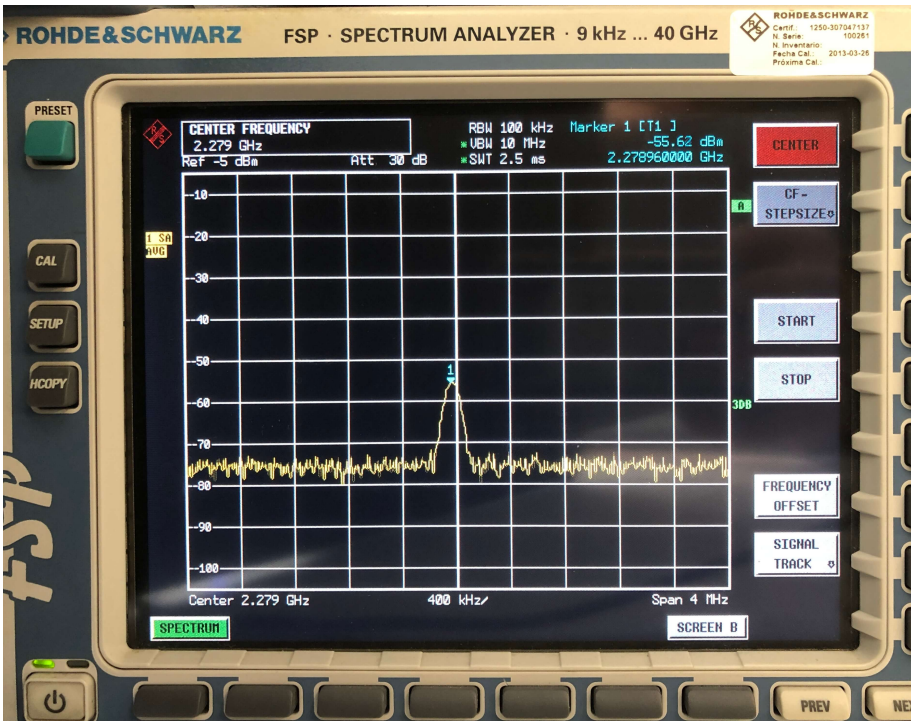


Figure 6.11: Transmission test local oscillator

The local oscillator peak shouldn't be seen, therefore, an attenuator should be added to reduce its amplitude until it cannot be appreciated in the following tests. This has been tested and can be done with a 30 dB attenuator.

Figure 6.12 shows the transmission pulse, which is much higher than the previous by approximately 32 dB. This pulse is indeed as expected and therefore, it can be concluded that the transmission code works correctly. In addition, this figure depicts the shift in frequency

introduced by the error in the local oscillator since the peak is at 2.27896 GHz instead of 2.279 GHz.

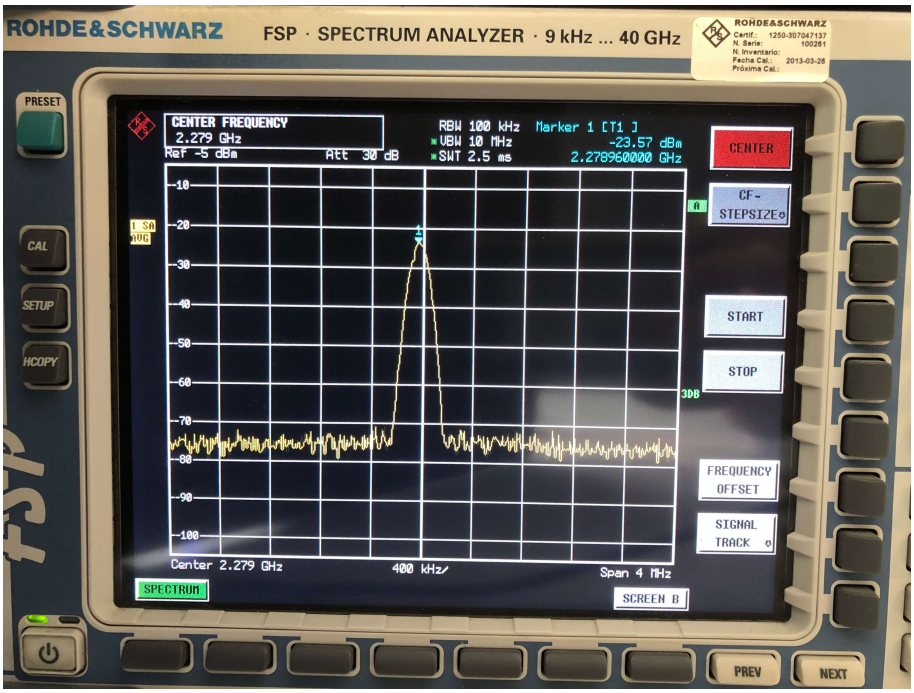


Figure 6.12: Transmission test local signal

6.4.2 Reception test

A basic reception test is done in order to check that the ADALM-PLUTO can receive bits at a given frequency. This test will consist of generating a signal at the central S-Band downlink frequency of 2.279 GHz and displaying the reception of samples in the ADALM-PLUTO through a laptop.

This is a very simple test and requires only of a few components:

- Signal generator. For this test, the Rode & Schwarz SM300 Signal generator has been used. This equipment can generate signals from 9 kHz to 3 GHz, and is therefore appropriate to generate a signal in the desigred S-Band frequency.
- SDR device. As has been stated previously, this test will use the ADALM-PLUTO.
- Coaxial cable.

The configuration parameters of the signal generator and the SDR are shown in 6.1.

Table 6.1: Reception test parameters

Parameters	Value	Units
Signal generator		
Frequency	2.279	GHz
Power	-28	dBm
ADALM-PLUTO receiver		
Frequency	2.279	GHz
Bandwidth	2.4	MHz

Figure 6.13 depicts the testing scenario diagram, where a signal is generated in the signal generator and transmitted via a coaxial cable connected to the receiver port of the SDR.

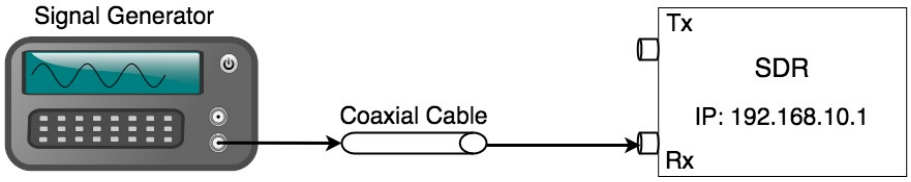


Figure 6.13: Reception test setup

Figure 6.14 depicts the signal generator configuration used in the test.

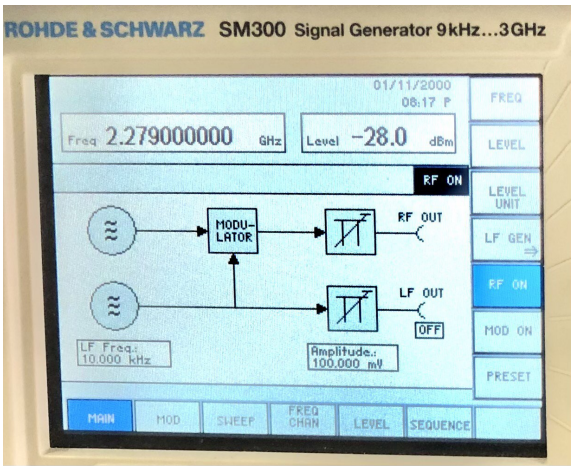


Figure 6.14: Signal generator configuration

6.4.2.1 Results and conclusions

In order to determine that the ADALM-PLUTO is able to receive a signal at the given frequency, the ”in-phase” and ”quadrature” samples displayed by the laptop when the signal generator is transmitting should have much higher amplitudes than when the transmission is disabled.

In the test, when the ADALM-PLUTO reception is activated without a transmitting signal, the received samples commonly present samples from approximately -2 to +2, these val-

ues are to be expected and result from noise. Nevertheless, when the signal generator is activated, the receiver stores much higher samples in this frequency, with amplitudes ranging from 500 and -500 approximately, indicating the presence of a signal. These results indicate that the ADALM-PLUTO is able to receive a signal in the desired frequency, and that the reception code works properly.

6.4.3 Transmission and reception test

Having determined that both transmission and reception configurations work correctly, the next step is to test that both configurations can be used together, and that what is sent by one device is received in the other.

In order to test the transmission and reception capabilities using the ADALM-PLUTO, both CPU cores can be used in order to maximize the efficiency. This configuration is detailed in appendix C.4.

The setup can be seen in Figure 6.15.

Figure 6.15: Pluto Testing Setup

As it can be seen in the block diagram, a 30 dB attenuator has been placed in the transmitter connection to attenuate the local oscillator signal that appears when the device is not sending any information.

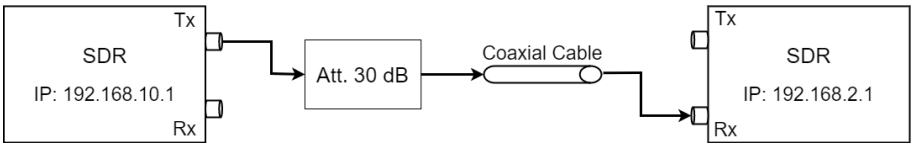


Figure 6.16: Pluto Testing Block Diagram Setup

The test itself consists in sending a signal from the transmitting SDR and checking if this is received it in the receiving SDR. It should be noted that the transmission has been carried out with constant amplitude samples in the quadrature and null in samples in the in-phase branch of the transmitter.

6.4.3.1 *Results and conclusions*

Figure 6.17 depicts the received signal. As it can be clearly seen, approximately the first 3300 samples correspond to the sent exponential signal, and the following, which have a much lower amplitude, correspond to the local oscillator. This demonstrates the correct transmission and reception of the SDR devices.

Furthermore, it can also be noted that the signal has been modulated in frequency. This is because the local oscillator of the devices are not perfect and each introduce an error. Therefore, the signals are down-converted with an error in frequency, that is the frequency that can be appreciated in the Figure 6.17.

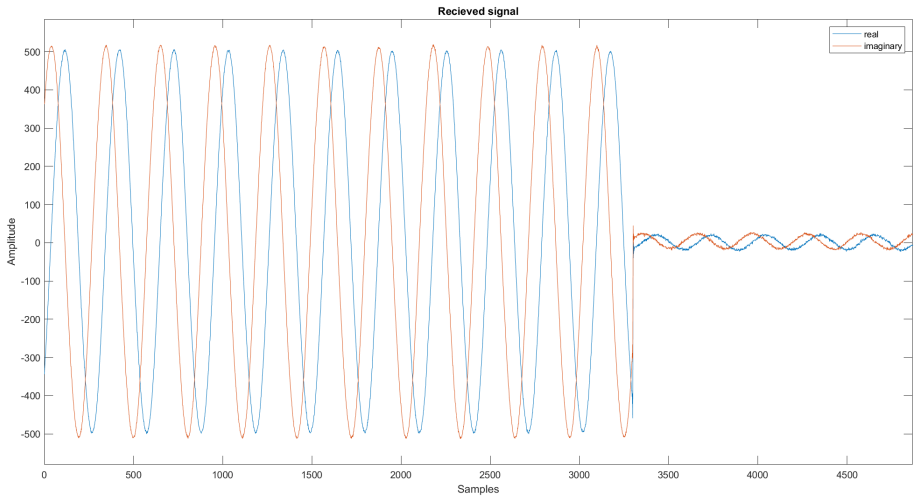


Figure 6.17: Received signal

6.4.4 Costas Loop test

In space communications, it is imperative that the receptor is able to detect appropriately the received signal even when it arrives with a frequency shift, due to Doppler, or a phase shift due to the transmission. Therefore, a very important test to perform is to check that should the received signal be modulated in amplitude due to the Doppler frequency shift, the detector will be able to detect the shifts.

The Costas Loop has been designed and tested in Matlab initially and then translated and tested in C++. First testing it with a generated signal an then with a signal obtained in the ADALM-PLUTO after synchronization.

6.4.4.1 *Generated signal test*

In order to test the correct functioning of the Costas Loop, a BPSK signal has been generated. This has been done by creating an array of random bits and later on modulating them with a BPSK modulator. In order to simulate the Doppler frequency effect that the satellite and ground station will experience, the signal has been then up-converted with a simulated Doppler frequency of 20 kHz. The obtained signal is represented in Figure 6.18.

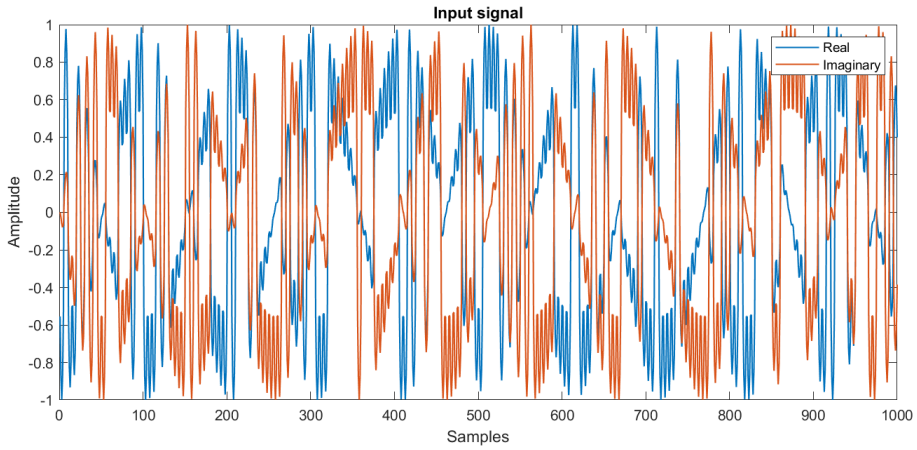


Figure 6.18: Input signal in Costas

As it can be seen, this signal cannot be demodulated and it has to undergo a carrier tracking system. After the Costas Loop is applied, the plot in Figure 6.19 is obtained. As it can be seen, the signal is no longer modulated in amplitude by the Doppler frequency shift, but instead, now presents a constant amplitude, which in the plot has been normalized to 1. It should be noted that theoretically, the quadrature (”Q”) branch of the signal should be null, since this test has been carried out without noise. The small amount of signal that can be seen is due to miss-adjustments on the parameters. This is not ideal because some power is lost in the ”Q” branch, since it will be discarded. Nevertheless, this recovered signal validates the design.

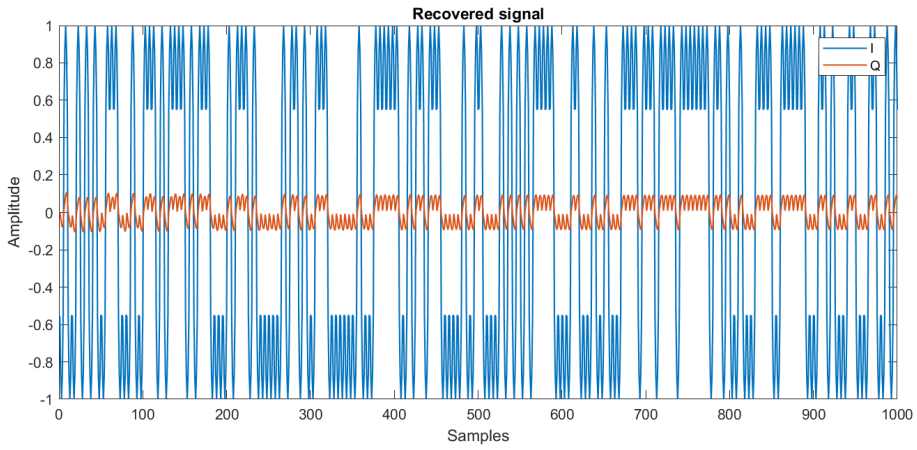


Figure 6.19: Recovered signal in Costas

The following figure depicts the sampled and demodulated signal. This plot contains the initial data signal and the recovered data, which has been displaced minimally in order to be able to check that the recovered data corresponds to the initial. As it can be seen, this is the case, which indicates that the Costas Loop has worked properly.

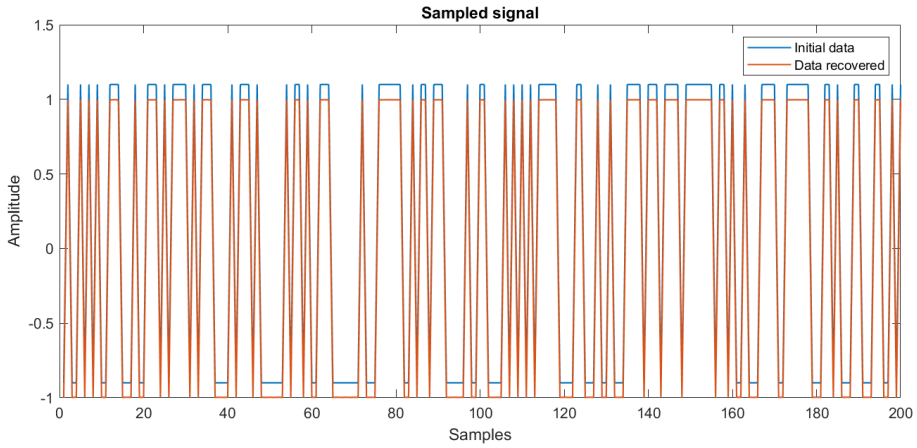


Figure 6.20: Recovered data in Costas

Moreover, the following Figure 6.21 shows the constellation of the received signal. This corresponds as expected to the BPSK constellation.

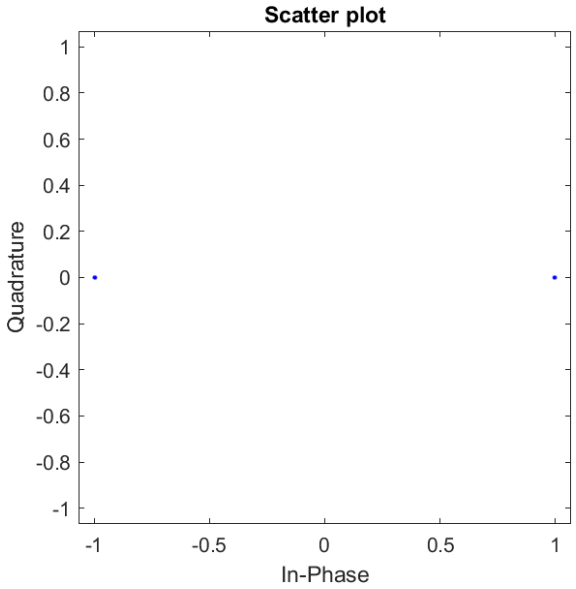


Figure 6.21: Scattered plot in Costas

Having demonstrated that the Costas Loop design and implementation is correct, several other tests have been carried out simulating the noise that the system is expected to experience. These are available in Appendix F, and verify that at the worst SNR condition expected (9 dB), the system will continue to function adequately.

6.4.4.2 Signal test in ADALM-PLUTO

In order to test the functioning of the Costas Loop in the system two ADALM-PLUTOs are required: one to act as the transmitter, and another as the receiver, connected via a coaxial cable. The transmitter is set to send an array of random bytes and the receiver is set to activate the Costas Loop after the signal synchronization.

The input signal received is shown in Figure 6.22. This particular test is set to correct the local oscillator frequency shift between the two devices, which as it can be seen, is considerable.

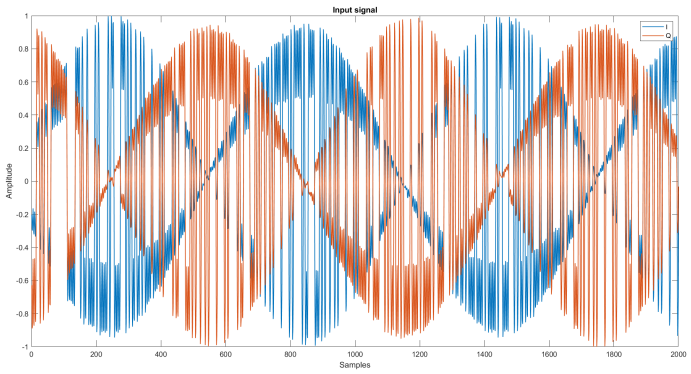


Figure 6.22: Input signal in the receiver

The Costas Loop approximates the frequency shift with the NCO. It's resulting signal is displayed in Figure 6.23, and as it can be noted, it coincides with the incoming frequency depicted in the previous figure.

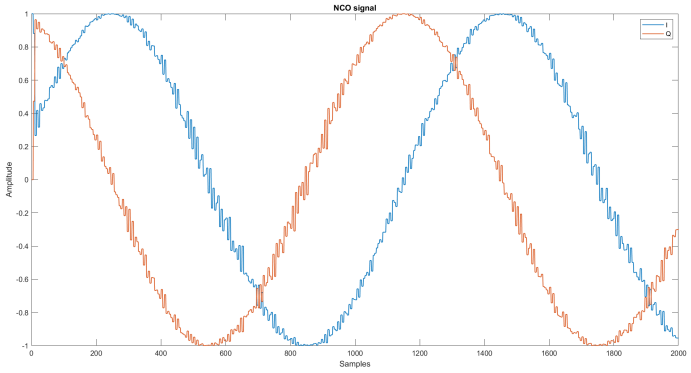


Figure 6.23: NCO signal

After the loop, the Costas recovers the signal. This is shown in Figure 6.24. Moreover, Figure 6.25 represents the sampled signal.

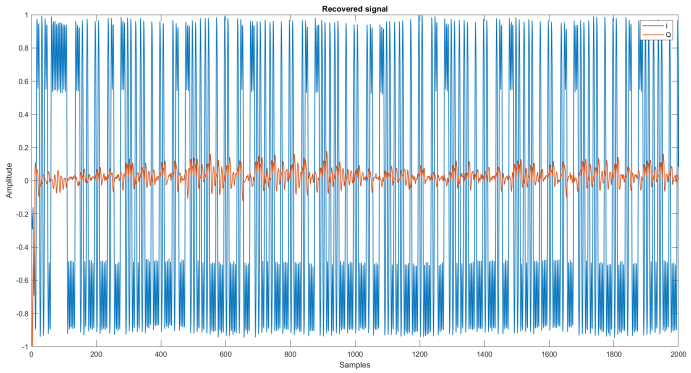


Figure 6.24: Recovered signal

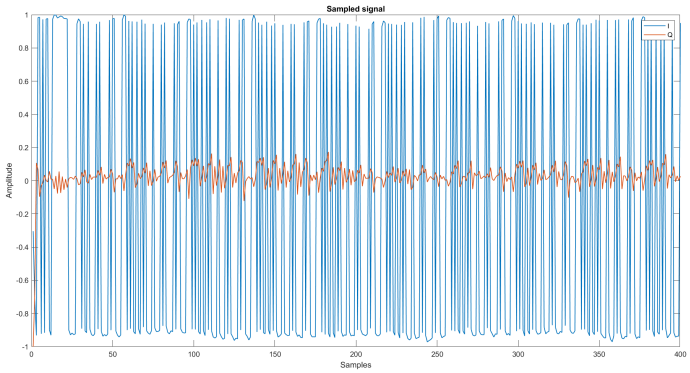


Figure 6.25: Sampled recovered signal

Finally, the constellation is correspondent to that of a BPSK as expected.

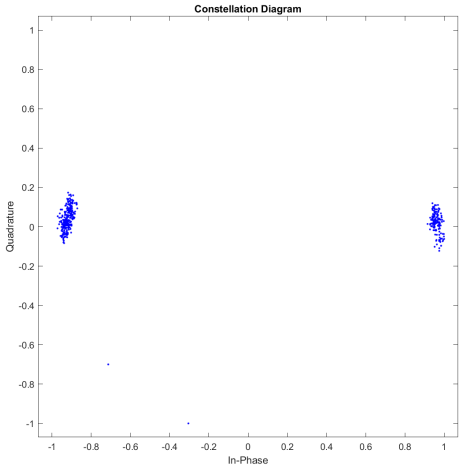


Figure 6.26: Constellation diagram

This test has also been carried out by adding an extra 30dB attenuation to the transmitter. With this scenario, the received signal is very degraded, as it can be seen in Figure 6.27.

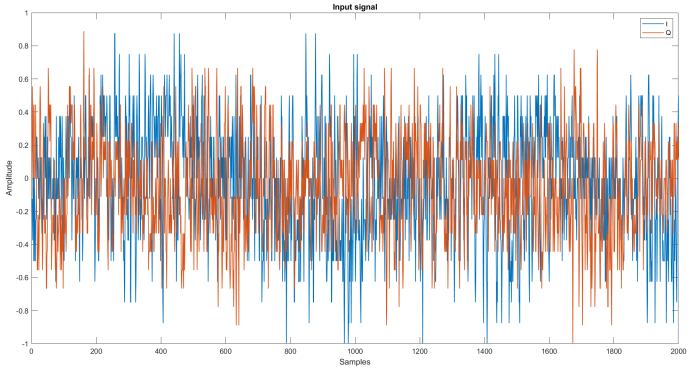


Figure 6.27: Input signal in the receiver with added attenuation

Even when the incoming signal is detected with very low power, the following plot demonstrates that the Costas Loop is able to correct the phase shift, even if it takes the loop

several samples to do so (approximately 25), whereas in the previous test it was almost instantaneous (2 samples).

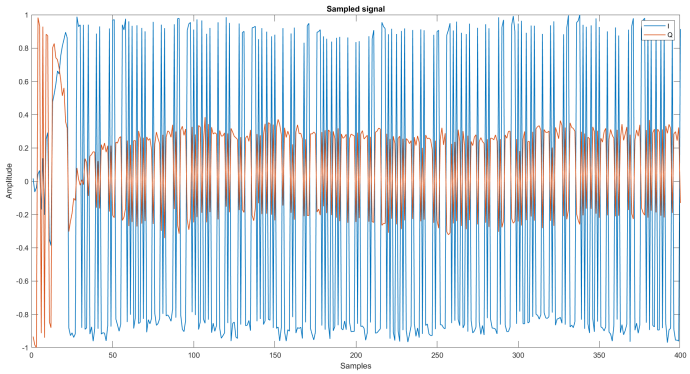


Figure 6.28: Sampled recovered signal

The constellation diagram is depicted in Figure 6.29. This diagram is particularly useful to understand the necessity of a Costas loop. As it can be seen, the constellation presents a number of samples following a circle around the centre. These are the initial 25 samples recovered by the system that have not been corrected by the loop. If there was no Costas Loop, the constellation would rotate and finally be a circle instead of two clutters, making impossible to recover the bits. The Costas loop, by tracking the frequency shift, stops this rotation and the bits can be recovered.

Moreover, it should also be noted that the clutters are a little deviated from their expected position. This is a result of the loop not being fully adjusted for this scenario.

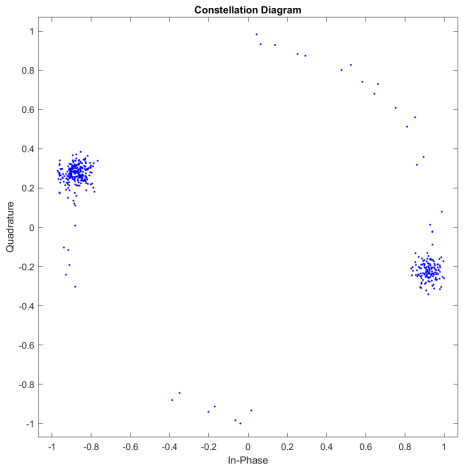


Figure 6.29: Constellation diagram

6.4.4.3 Conclusion

After carrying out the previous tests, it can be concluded that the system works correctly for the desired SNR and that the implementation in the ADALM-PLUTO is functional.

6.4.5 Message transmission and reception test

This test aims to verify the correct functioning of the transmission and reception functions together with all the physical channel chain that has been developed in software.

This test requires two ADALM-PLUTOs, one to act as the transmitter, and another as the receiver, connected via a coaxial cable. In order to control both of them and check the results, two consoles are required, one for each, as Figure 6.30 illustrates.

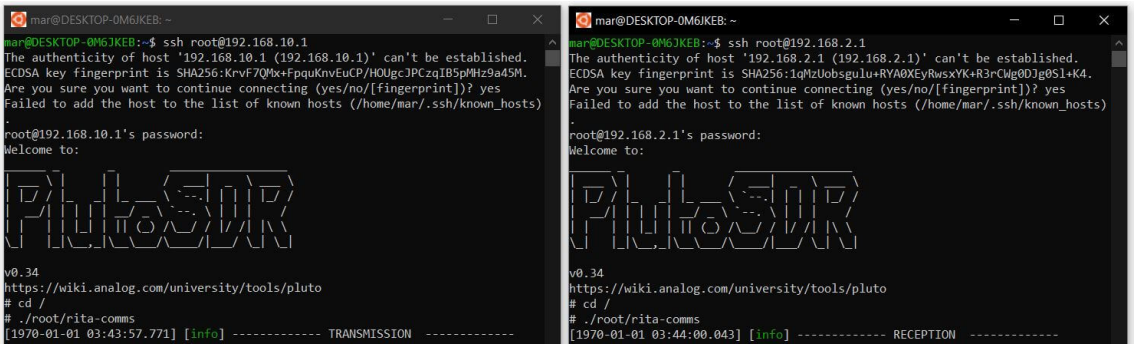


Figure 6.30: Console configuration

Two tests have been carried out for this configuration, the first to transmit an array of bytes, and the second, to encode, transmit, receive and decode a message. These tests will verify the modulation and demodulation of bytes, the synchronization function, the Costas loop and the transmission and reception in the physical channel, and therefore, are extremely important and representative.

The first and most simple test has consisted in sending a simple array of bits. For this, the following steps have been carried out automatically by the code:

- At the transmitter:
    1. Generate the synchronization word and add it to the array of bytes to be transmitted.
    2. Modulate the bytes in BPSK.
    3. Multiply the resulting signal by a train of conformation pulses.
    4. Configure the transmitter.
    5. Transmit.
- At the receiver:
    1. Generate the synchronization word modulate it in BPSK and multiply it by a train of conformation pulses.
    2. Configure the receiver.
    3. Activate the reception of samples.
    4. Activate the synchronization of the signal.

5. Activate the Costas loop.
6. Display demodulated bytes.

The second test includes the functions of encoding and decoding processes that have been tested in subsection 6.3.1.

6.4.5.1 Results and conclusions

The results of these tests are shown in Appendix H. The first test, based on the transmission of a message has been successful, and the receiver has been able to recover the original message. This indicates that the whole physical layer software design is correct and that the synchronization, the Costas Loop and the demodulation are correct. Moreover, two other tests where the message has been encoded and later decoded have been carried out, the second having more attenuation than the first, presenting more errors in the transmission. However, both tests have been able to recover the original message sent. This verifies the adequate functioning of all the physical layer software design as well as the channel codification.

CONCLUSION

This project has been carried out with a clear objective: to design and implement an S-Band communications link for the RITA Payload. Nevertheless, this main task has consisted of a number of smaller tasks.

The first step that has been carried out has been the study of the communications scenario. This has involved a theoretical research of satellite communications followed by an orbital study through simulations. These have been successful in providing the necessary information regarding the communication's scenario for which the software has to be designed. Particularly, a relevant parameter that has been obtained in this step is the estimated time of LOS (7 minutes), which was necessary later on for the design.

The following step was to carry out the link budget. This has provided a deeper understanding of the link conditions and all the parameters that have a direct effect on it. The obtained results have been crucial in defining the software requirements such as the minimum SNR (9 dB) and the expected maximum Doppler frequency shift that the software should be able to compensate ($\pm$47kHz). Moreover, it also successfully provided information regarding the characteristics of the link, such as the fact that the link will have an approximate bit rate of 1.7 Mbps.

Having understood the software requirements, its design and implementation has been carried out. In order to do so, the CCSDS standard has been followed, which has allowed for a more guided design and ensured that the blocks and protocols used are adequate for space communications. These has been divided into three main blocks: the application layer, the channel codification and the physical channel.

The application layer has been designed to fit the needs of the link, by including a specific design of headers and telemetry messages fit for an efficient management of packets. Moreover, the Stop and Wait ARQ protocol has been chosen amongst the others due to its simplicity given the time constraints of the project. It would be interesting to revisit this design in the future to implement a more efficient protocol to provide faster communication between satellite and ground station.

The second part, channel codification, has been implemented with a more fixed design, following one of the CCSDS encoding and decoding recommendations. With this implementation the transmitted messages are protected against errors produced along the transmission such as deep-fading amongst others.

The third part, the physical channel, has taken the longest to develop, since the union of software with hardware can be trickier than it appears. Particularly, the Costas Loop implementation has been problematic, since the standard design only works for some cases, and not when the carrier frequency is lower than the modulation bandwidth. This realization was made quite late after a long implementation process. Nevertheless, once redesigned, the whole implementation of the physical channel has been successful.

Finally, the testing of the system has verified the functioning of all of the above using two ADALM-PLUTO devices. However, in order to fully verify the design, the initial software requirements must be reviewed. These are clearly depicted in Table 6.2.

Table 6.2: Requirements verification

Requirements	**Verification**
Programming language must be appropriate for embedded systems.	Fulfilled by using C++.
Software must communicate with the main code of the satellite.	Fulfilled by its designed to act upon commands from the main code and return other parameters.
Must be used in both the satellite and the ground station.	Validated thanks to the software's modular design.
High level of error correction	Fulfilled with the implementation of the channel coding, as seen in section 5.4
Must have the ability to correct the Doppler frequency shift of incoming signals	Achieved with the Costas Loop implementation, as seen in subsection 6.4.4.
Codification and modulation scheme must function properly with a SNR of 9 dB.	Has been tested and confirmed.
Must be very efficient.	Requires further development.
Must have a high level of robustness.	Requires further development.

As this indicates, all the requirements have been fulfilled in exception of the last two. Having a robust and efficient code is of great importance in space systems, nevertheless, due to the time restrictions in the development of the project, functionality has been prioritized. Therefore, this project requires further work to ensure the system is as efficient as possible and that it has a higher level of robustness.

Future work

Even if a large amount of work has been carried out, due to time restrictions, there are still several tasks to be done before the communication software can be deemed final. These consist of the testing and verification of the whole system. This should be done as follows:

A first test should consist of running the software in the flight hardware. This would allow to check that the software works as expected.

Another test should involve using the S-Band payload antenna with the ADALM-PLUTO in order to carry out some transmission tests. These would verify the antenna functioning and would allow to gather real data of the signal that the satellite would transmit from space.

Furthermore, having in mind that each part of the software has been tested on its own, the software should be tested next in a real scenario. Therefore, the following tests to be carried out are to begin a testing campaign with the real hardware: using the S-Band payload antenna and the S-Band parabolic antenna in Montsec. This should provide the final verification of the system.

Nevertheless, the final verification is not the only thing left to be done. As stated previously, once the software is fully verified, the next step is increasing its efficiency and robustness, which are two very important characteristics in space applications. This would involve further testing for possible errors that might arise in situations that have not been considered, a revision of large functions to check if there are faster methods to carry them out, etc.

Highly related to increasing the efficiency and speed of the program, another thing that could be done is be to implement the Costas Loop and demodulator in the FPGA. This would have a great impact in the system's efficiency and make it faster. Nevertheless, this implementation would require a deeper understanding of FPGA and programming knowledge in VHDL, which is the language used in digital circuits. This idea emerged at the beginning of the design implementation but was paused due to the added difficulty considering the time restrictions of the project.

BIBLIOGRAPHY

[1] EU Science Hub. Earth observation. https://joint-research-centre.ec.europa.eu/scientific-activities-z/earth-observation_en. 3

[2] NASA Earth Observatory. Remote sensing methods. https://earthobservatory.nasa.gov/features/RemoteSensing/remote_08.php. 3

[3] Ashutosh Agrawal. Agile methodology: Incremental and iterative way of development. https://medium.com/@ashutoshagrawal1010/agile-methodology-incremental-and-iterative-way-of-development-a6614116ae68. 4

[4] NASA. What is a satellite? <https://www.nasa.gov/audience/forstudents/5-8/features/nasa-knows/what-is-a-satellite-58.html>. Last accessed 18 April 2022. 7

[5] European Space Agency. A man - and an equation. <https://blogs.esa.int/rocketscience/2012/10/14/a-man-and-an-equation/>. Last accessed 18 April 2022. 7

[6] Justin Pollard. The eccentric engineer: Herman potocnik and his forgotten space station. <https://eandt.theiet.org/content/articles/2018/01/the-eccentric-engineer-herman-potocnik-and-his-forgotten-space-station/>. Last accessed 18 April 2022. 7

[7] Joan Lisa Bromberg. Nasa and the space industry. JHU Press, 2011. 8

[8] Greelane. Sputnik 1: La pequeña esfera brillante que desencadenó la carrera espacial. https://www.greelane.com/es/. Last accessed 5 April 2022. 8

[9] Ashley Campbell. August 1960 - project echo launched. https://www.nasa.gov/directorates/heo/scan/images/history/August1960.html. Last accessed 11 April 2022. 9

[10] JJ Velasco. Telstar 1, el primer satélite de comunicaciones comerciales. https://blogthinkbig.com/telstar-1-historia. Last accessed 11 April 2022. 10

[11] Gunter D Krebs. Syncom 1, 2, 3. https://space.skyrocket.de/doc_sdat/syncom-1.htm. Last accessed 11 April 2022. 10

[12] DAVID S. F. PORTREE. Lm relay experiment laboratory (1966). https://www.wired.com/2012/08/lem-relay-experiment-laboratory-1966/. Last accessed 11 April 2022. 10

[13] Noticias de la Ciencia y Tecnología. Gran enciclopedia de la astronáutica (268): Intelsat-i. https://noticiasdelaciencia.com/art/9673/articulo. Last accessed 11 April 2022. 11

[14] Wikipedia. Molniya (satellite). https://en.wikipedia.org/wiki/Molniya_(satellite). Last accessed 11 April 2022. 12

[15] Gunter D Krebs. Molniya-1s (11f658). https://space.skyrocket.de/doc_sdat/molniya-1s.htm. Last accessed 11 April 2022. 12

[16] The European Space Agency. Earth observation missions. https://www.esa.int/Enabling_Support/Operations/Earth_observation_missions. 12

[17] Aeroespacial. El satélite smos, de la esa, y su instrumento miras, superan una década en órbita. https://actualidadaeroespacial.com/el-satelite-smos-de-la-esa-y-su-instrumento-miras-superan-una-decada-en-orbita 12

[18] The European Space Agency. Smos, la misión de la esa para el estudio del agua. https://www.esa.int/Space_in_Member_States/Spain/SMOS_la_mision_de_la_ESA_para_el_estudio_del_agua. 12

[19] The European Space Agency. Miras. https://earth.esa.int/eogateway/instruments/miras. 12

[20] Wikipedia. Soil moisture and ocean salinity. https://en.wikipedia.org/wiki/Soil_Moisture_and_Ocean_Salinity. 12

[21] The European Space Agency. Smos celebra su primer año en órbita. https://www.esa.int/Space_in_Member_States/Spain/SMOS_celebra_su_primer_ano_en_orbita. 12

[22] Mclee Kerolle. Newspace - is this the advent of the second space age? <https://www.spaceboard.eu/articles/space-out/newspace-is-this-the-advent-of-the-second-space-age->. Last accessed 14 April 2022. 13

[23] ISISPACE. Cubesat information. <https://www.isispace.nl/cubesat-information/>. Last accessed 19 February 2022. 13

[24] Alen Space. A basic guide to nanosatellites. https://alen.space/basic-guide-nanosatellites/. Last accessed 19 February 2022. 14, 16

[25] nanosats. Nanosats database. <https://www.nanosats.eu/>. Last accessed 20 March 2022. 14, 15

[26] Alen. 10 advantages of cubesats vs. conventional satellites. <https://info.alen.space/advantages-of-cubesats-vs-conventional-satellites?hsLang=en>. 15

[27] Adriano Camps. Nanosatellites and applications to commercial and scientific missions. In Vladislav Demyanov and Jonathan Becedas, editors, *Satellites Missions and Technologies for Geosciences*, chapter 9. IntechOpen, Rijeka, 2020. 16

[28] Tony Greicius. Nasa tests tiny satellites to track global storms. <https://www.nasa.gov/feature/jpl/nasa-tests-tiny-satellites-to-track-global-storms>. Last accessed 24 March 2022. 16

[29] Sara Blumberg. Tiny satellite's first global map of ice clouds. <https://www.nasa.gov/feature/goddard/2018/tiny-satellites-first-global-map-of-ice-clouds>. Last accessed 24 March 2022. 16

[30] Solar System Exploration NASA Science. 10 things: Cubesats — going farther. <https://solarsystem.nasa.gov/news/834/10-things-cubesats-going-farther/>. Last accessed 24 March 2022. 17

[31] NanoSat Lab. 3cat-1. <https://nanosatlab.upc.edu/en/missions-and-projects/3cat-1>. Last accessed 03 April 2022. 17

[32] NanoSat Lab. 3cat-3. <https://nanosatlab.upc.edu/en/missions-and-projects/3cat-3>. Last accessed 03 April 2022. 17

[33] NanoSat Lab. 3cat-4. <https://nanosatlab.upc.edu/en/missions-and-projects/3cat-4>. Last accessed 03 April 2022. 17

[34] NanoSat Lab. Fsscat. <https://nanosatlab.upc.edu/en/missions-and-projects/fsscat>. Last accessed 03 April 2022. 18

[35] NanoSat Lab. Clean room. <https://nanosatlab.upc.edu/en/facilities-folder/clean-room>. Last accessed 03 April 2022. 18

[36] NanoSat Lab. Thermal and vacuum chamber (tvac). <https://nanosatlab.upc.edu/en/facilities-folder/thermal-and-vacuum-chamber-tvac>. Last accessed 03 April 2022. 19

[37] NanoSat Lab. Vibration table. <https://nanosatlab.upc.edu/en/facilities-folder/shake-table>. Last accessed 03 April 2022. 19

[38] L. Fernandez; M. Sobrino; A. Rodriguez; A. Gonga; C. Molina; L. Rayon; M. Badia; P. Fabregat; A. Perez-Portero; J. Ramos-Castro; A. Ruiz de Azua; A. Calveras; A. H. Jallad and Z. Abdul Aziz. Sdr-based lora enabled on-demand remote acquisition experiment on-board the alainsat-1. <https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/stamp/stamp.jsp?tp=&arnumber=9553020>. Last accessed 19 April 2022. 22

[39] A. Perez-Portero; P. Fabregat; M. Badia; M. Sobrino; C. Molina; L. Fernandez; L. Rayon; A. Rodríguez; J.F. Munoz-Martin; A. Gonga; J. Ramos-Castro; A.H. Jallad and Z. Abdul Aziz. Rita: A 1u multi-sensor payload for the grsssat contributing soil moisture, vegetation analysis and rfi detection. *IGARSS Conference*, 2021. 24

[40] European Space Agency. Satellite frequency bands. <https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Satellite_frequency_bands>. Last accessed 19 April 2022. 27

[41] Cuadro nacional de atribuciÓn de frecuencias. <https://portal.mineco.gob.es/RecursosArticulo/mineco/ministerio/participacion_publica/audiencia/ficheros/210422-CNAF-2021.pdf>. 28

[42] Institut d'Estudis Espacials de Catalunya IEEC. Observatori astronòmic del montsec. <http://www.ieec.cat/content/206/what-s-the-oadm/>. Last accessed 23 March 2022. 30

[43] NanoSat Lab. Ground segment. <https://nanosatlab.upc.edu/en/facilities-folder/ground-segment>. Last accessed 16 March 2022. 30

[44] UTCC Asean Economic Community Strategy Center. Thaicom-cat tie-up for low earth orbit satellite services for cambodia and other countries. <http://aec.utcc.ac.th/thaicom-cat-tie-up-for-low-earth-orbit-satellite-services-for-cambodia-and-oth>. Last accessed 18 April 2022. 31

[45] European Space Agency. Types of orbits. <https://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits>. Last accessed 21 March 2022. 32, 33

[46] Yu.F. Kolyuka; N.M. Ivanov; T.I. Afanasieva and T.A. Gridchina. Examination of the lifetime, evolution and re-entry features for the "molniya" type orbits. https://issfd.org/ISSFD_2009/CollisionRiskII/Kolyuka.pdf. 32

[47] Heimdal Satellite Technologies. Satellites: Sun synchronus orbits. <https://hsat.space/satellites-sun-synchronus-orbits/>. Last accessed 19 March 2022. 34

[48] Russian Zarya Soviet and International Space Flight. Sun-synchronous satellites - by satellite age. <http://www.zarya.info/Diaries/Launches/GeoSS/ss-Age.php>. Last accessed 23 March 2022. 35

[49] N2YO. Saudisat 5b. <https://www.n2yo.com/satellite/?s=43833>. Last accessed 23 March 2022. 35

[50] Ansys Government Initiatives (AGI). Systems tool kit. <https://www.agi.com/products/stk>. Last accessed 24 April 2022. 37

[51] Nasir Saeed; Ahmed Elzanaty; Heba Almorad; Hayssam Dahrouj; Tareq Y. Al-Naffouri and Mohamed-Slim Alouini. Cubesat communications: Recent advances and future challenges. <https://arxiv.org/pdf/1908.09501.pdf>, April 2020. Last accessed 13 March 2022. 43, 45

[52] Nikolova. Lecture 5: Polarization and related antenna parameters. <https://www.ece.mcmaster.ca/faculty/nikolova/antenna_dload/current_lectures/L05_Polar.pdf>. Last accessed 12 March 2022. 45

[53] Keith Willey. Antenna selection to minimize pointing requirements. <https://www.microwavejournal.com/articles/3371-antenna-selection-to-minimize-pointing-requirements>. Last accessed 30 April 2022. 46

[54] CelesTrak. Norad general perturbations (gp) element sets. <http://celestrak.com/NORAD/elements/>. Last accessed 24 April 2022. 46

[55] Kathleen Riesing. Orbit determination from two line element sets of iss-delpoyed cubesats. <https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=3222&context=smallsat&httpsredir=1&referer=>. Last accessed 30 April 2022. 46

[56] Atmospheric transmittance. https://www.usna.edu/Users/oceano/pguth/md_help/remote_sensing_course/atmos_transmit.htm. Last accessed 12 March 2022. 47

[57] Spectral calc. https://www.spectralcalc.com/info/about. Last accessed 12 March 2022. 47

[58] Modtran. http://modtran.spectral.com/. Last accessed 12 March 2022. 47

[59] MIT Lincoln Laboratory. Introduction to radar systems - propagation effects. https://www.ll.mit.edu/sites/default/files/outreach/doc/2018-07/lecture%203.pdf/. Last accessed 30 April 2022. 48

[60] Wikipedia. Psk ber curves.svg. https://es.m.wikipedia.org/wiki/Archivo:PSK_BER_curves.svg. Last accessed 30 April 2022. 49

[61] Battista; Umberto Landini; Alberto Golebiowski; W. Michalczyk; A. Duda; K. Sochaczewska. Design of net ejector for space debris capturing. <https://www.researchgate.net/publication/316789217_Design_of_net_ejector_for_space_debris_capturing>. Last accessed 19 April 2022. 53

[62] The Consultative Comittee for Space Data Systems. *TM SYNCHRONIZATION AND CHANNEL CODING — SUMMARY OF CONCEPT AND RATIONALE, INFORMATIONAL REPORT*. 2020. 60, 75, 76, 77, 78, 79

[63] CMake. About cmake. https://cmake.org/. 61

[64] Alberto Leon-Garcia and Indra Widjaja. *Communication networks: Fundamental concepts and key architectures*. McGraw-Hill, 2004. 66, 67, 68

[65] Toml, tom's obvious minimal language. https://toml.io/en/. Last accessed 5 June 2022. 71

[66] Wikipedia. Hash table. https://en.wikipedia.org/wiki/Hash_table. Last accessed 5 June 2022. 71

[67] JSON. Introducing json. https://www.json.org/json-en.html. Last accessed 5 June 2022. 71

[68] Matt Keti. Reed-solomon codes and the deep hole problem, 2015. 76

[69] Electronics Desk. Interleaving in communication. https://electronicsdesk.com/interleaving.html. Last accessed 6 February 2022. 78

[70] MIT. Lecture 8 - convolutional coding. http://web.mit.edu/6.02/www/f2010/handouts/lectures/L8.pdf. Last accessed 20 April 2022. 79

[71] Kai Chang. *RF and Microwave Wireless Systems*. John Wiley & Sons, Inc., 2000. 84

[72] Wikipedia. Archivo: Bpsk gray coded.svg. https://es.m.wikipedia.org/wiki/Archivo:BPSK_Gray_Coded.svg. Last accessed 13 April 2022. 85

[73] Wikimedia Commons. File:phase modulation bpsk gps.svg. https://commons.wikimedia.org/wiki/File:Phase_modulation_BPSK_GPS.svg. Last accessed 13 April 2022. 85

[74] Wikipedia. Nyquist isi criterion. https://en.wikipedia.org/wiki/Nyquist_ISI_criterion. 86

[75] Ali Muqaibel. Pulse shaping and nyquist criteria for zero isi. https://www.youtube.com/watch?v=--0z_iUNAN0. 87

[76] The Consultative Comittee for Space Data Systems. *TM SYNCHRONIZATION AND CHANNEL CODING - RECOMMENDED STANDARD*. 2017 2020. 89

[77] Debesh Choudhury. Teaching the concept of convolution and correlation using Fourier transform. In Xu Liu and Xi-Cheng Zhang, editors, *14th Conference on Education and Training in Optics and Photonics: ETOP 2017*, volume 10452, pages 183 – 188. International Society for Optics and Photonics, SPIE, 2017. 90

[78] Devon Fernandez and Sanjeev Manandhar. Digital phase locked loop. https://ece.umaine.edu/wp-content/uploads/sites/203/2012/05/Devon_Sanjeev.pdf, December 2003. 91

[79] Jeff Feigin. Practical costas loop design. https://ez.analog.com/cfs-filesystemfile/__key/communityserver-discussions-components-files/333/Costas-Loop.pdf?_=637218102685428736. Last accessed 20 May 2022. 91

[80] Devi K G Gayathri and Shamla B. Design and implementation of costas loop for bpsk demodulator. https://www.researchgate.net/publication/344217695_Design_and_Implementation_of_Costas_loop_for_BPSK_Demodulator. Last accessed 20 May 2022. 92

[81] Elliott D. Kaplan and Christopher J. Hegarty. *Understanding GPS - Principles and Applications*. Atrech House, 2006. 92

[82] Analog Devices. libiio. https://github.com/analogdevicesinc/libiio. 94

[83] ANALOG DEVICES. Adalm-pluto, software-defined radio active learning module. https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html. Last accessed 01 April 2022. 97

[84] Analog devices. Adalm-pluto detailed specifications. <https://wiki.analog.com/university/tools/pluto/devs/specs>. 98

[85] Annie Wilson. 2021 year in review: Statistics and satellites. <https://cosmoquest.org/x/2022/01/2021-year-in-review-statistics-and-satellites/>. Last accessed 14 March 2022.

[86] International Astronautical Federation (IAF). Intersputnik international organization of space communications. https://www.iafastro.org/membership/all-members/intersputnik-international-organization-of-space-communications.html. Last accessed 11 April 2022.

[87] Nagel Evlyn Márcia Leão de Moraes Novo Gustavo Willy and Milton Kampel. Nanosatellites applied to optical earth observation: a review. <https://doi.org/10.4136/ambi-agua.2513>. Last accessed 19 February 2022.

[88] NanoSat Lab. 3cat-2. <https://nanosatlab.upc.edu/en/missions-and-projects/3cat-2>. Last accessed 03 April 2022.

[89] Ibtissam Latachi; Mohammed Kari; Ahmed Hanafi; Tajjeeddine Rachidi; Ahmed Khalayoun; Nasser Assem; Sanae Dahbi and Smail Zouggar. Link budget analysis for a leo cubesat communication subsystem. *3rd International Conference on Advanced Technologies for Signal and Image Processing - ATSIP'2017*, 2017.

[90] Holli Riebeek. Catalog of earth satellite orbits. <https://earthobservatory.nasa.gov/features/OrbitsCatalog>. Last accessed 21 March 2022.

[91] CelesTrak. Norad two-line element set format. <https://www.celestrak.com/NORAD/documentation/tle-fmt.php>. Last accessed 24 March 2022. 3

[92] The Consultative Comittee for Space Data Systems. *CCSDS FILE DELIVERY PROTOCOL (CFDP)*. 2020.

[93] The Consultative Comittee for Space Data Systems. *CCSDS FILE DELIVERY PROTOCOL (CFDP)— PART 1: INTRODUCTION AND OVERVIEW*. 2021.

[94] The Consultative Comittee for Space Data Systems. *CCSDS FILE DELIVERY PROTOCOL (CFDP)— PART 2: IMPLEMENTERS GUIDE*. 2021.

[95] libcorrect. https://github.com/quiet/libcorrect, 2018. 16

[96] RF Wireless World. What is interleaving — advantages of interleaving in data communication. https://www.rfwireless-world.com/Terminology/Advantages-of-Interleaving-in-Data-Communication.html. Last accessed 6 February 2022.

[97] Tutorials Point. What is interleaving? https://www.tutorialspoint.com/what-is-interleaving. Last accessed 6 February 2022.

[98] S. Benedetto, Guido Montorsi, and Dariush Divsalar. Concatenated convolutional codes with interleavers. *Communications Magazine, IEEE*, 41:102 – 109, 09 2003.

[99] El M. Soudi M.I.Garcıa-Planas and L. E. Um. Concatenated convolutional codes: Analysis of control properties under linear systems theory point of view.

[100] Wikipedia. Hexspeak. https://en.wikipedia.org/wiki/Hexspeak. Last accessed 12 April 2022.

[101] NanoSat Lab. Ground segment. https://nanosatlab.upc.edu/en/facilities-folder/ground-segment. Last accessed 6 February 2022.

[102] IEEC. Montsec observatory. http://www.ieec.cat/content/206/what-s-the-oadm. Last accessed 6 February 2022.

[103] toml++. https://marzer.github.io/tomlplusplus/. 14

[104] Json for modern c++. https://github.com/nlohmann/json. 13

[105] Crc++. https://github.com/d-bahr/CRCpp.

[106] Ne10. https://github.com/projectNe10/Ne10. 19

[107] Project ne10. https://projectne10.github.io/Ne10/. 19

[108] Spdlog: a fast c++ logging library. https://github.com/gabime/spdlog. 13

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

APÈNDIXS

TÍTOL DEL TFG: S-Band communications design and implementation for 3Cat-6

TITULACIÓ: Double bachelor's degree in Aerospace Systems Engineering and Telecommunications Systems Engineering

AUTOR: Mar Munuera Vilalta

DIRECTORS: Hyuk Park
Adrián Pérez

DATA: July 4, 2022

APPENDIX A. RITA'S TARGET AREAS

Table A contains all the areas that will be studied by RITA payload and indicates their respective purposes.

Target Areas	**Operational Mode**	**Main purpose**	**Secondary purpose**
African Evergreen Forest	MWR + RFI	Vegetation	-
Alps	MWR + RFI	Vegetation	-
Amazonas	CAM + MWR + RFI	Vegetation	Soil moisture
Arabian Sea	CAM + MWR + RFI	Algae blooms	-
Australian Forests East	CAM + MWR + RFI	Vegetation	-
Australian Forests North	CAM + MWR + RFI	Vegetation	-
Bangladesh	CAM + MWR + RFI	Soil moisture	Vegetation
Borneo	CAM + MWR + RFI	Vegetation	-
Cocos Island	Calibration	Calibration	-
East Asia	MWR + RFI	RFI detection	-
Gulf Mexico	CAM + MWR + RFI	Oil spills	Algae blooms
Japan and Coast	MWR + RFI	Vegetation	-
Lake Aral	CAM + MWR + RFI	Soil Moisture	Algae blooms
Lake Chad	CAM + MWR + RFI	Soil Moisture	Algae blooms
Lake Chott El Djerid	CAM + MWR + RFI	Soil Moisture	Algae blooms
Lake Erie	CAM + MWR + RFI	Soil Moisture	-
Lake Etosha	CAM + MWR + RFI	Soil Moisture	Algae blooms
Lake Taihu	CAM + MWR + RFI	Soil Moisture	Algae blooms
Lake Urmia	CAM + MWR + RFI	Soil Moisture	Algae blooms
Lake Victoria Natron	CAM + MWR + RFI	Soil Moisture	Algae blooms
Lorentz National Park	CAM + MWR + RFI	Soil Moisture	-
Madagascar	MWR + RFI	Vegetation	-
Malaysia	MWR + RFI	Vegetation	-
Middle Asia	MWR + RFI	Vegetation	-
Netherlands	CAM + MWR + RFI	Soil moisture	-
Nile Sinai	CAM + MWR + RFI	Soil moisture	-
Okavango	CAM + MWR + RFI	Soil moisture	Vegetation
Panama	CAM + MWR + RFI	Oil spills	Algae blooms
Persian Gulf	CAM + MWR + RFI	Oil spills	Algae blooms
Plata Mouth	MWR + RFI	Soil moisture	-
Sahara Desert	Calibration	Calibration	-
Salt Lake City	MWR + RFI	Vegetation	-
Spain	CAM + MWR + RFI and LoRa	Soil moisture	Vegetation
Uyuni Salt Lake	Calibration	Calibration	-

Table A.1: Target areas and purposes

APPENDIX B. TWO LINE ELEMENT DECODIFICATION

This appendix contains the description of how NORAD Two-Line Elements can be decoded. This information has been extracted from Celestrak website accessible at [91].

Each satellite's data consists of 3 lines that can be expressed the format below, where line 0 is a 24 character name and lines 1 and 2 are the standard Two-Line Orbital Element Set Format identical to that used by NORAD and NASA. The format description is:

AAAAAAAAAAAAAAAAAAAAAAA

1 NNNNNU NNNNNAAA NNNNN.NNNNNNNN +.NNNNNNNN +NNNNN-N +NNNNN-N N NNNNN

2 NNNNN NNN.NNNN NNN.NNNN NNNNNNN NNN.NNNN NNN.NNNN NN.NNNNNNNNNNNNN

The first line contains the following information:

- 1st number: Line number of the element data
- 3rd to 7th number: Satellite number
- 8th number: Classification
- 10th to 11th number: International Designator (Last two digits of launch year)
- 12th to 14th number:International Designator (Launch number of the year)
- 15th to 17th number: International Designator (Piece of the launch)
- 19th to 20th number: Epoch Year (Last two digits of year)
- 21th to 32th number: Epoch (Day of the year and fractional portion of the day)
- 34th to 43th number: First Time Derivative of the Mean Motion
- 45th to 52th number: Second Time Derivative of Mean Motion (Leading decimal point assumed)
- 54th to 61th number: BSTAR drag term (Leading decimal point assumed))
- 63th number:Ephemeris type
- 65th to 68th number: Element number
- 69th number: Checksum

The second line contains the following information:

- 1st number: Line number of the element data
- 3rd to 7th number: Satellite number
- 8th number: Classification

- 9th to 16th number: Inclination [Degrees]
- 18th to 25th number: Right Ascension of the Ascending Node [Degrees]
- 27th to 33th number: Eccentricity (Leading decimal point assumed)
- 35th to 42th number: Argument of Perigee [Degrees]
- 44th to 51th number: Mean Anomaly [Degrees]
- 53th to 63th number: Mean Motion [Revs per day]
- 64th to 68th number: Revolution number at epoch [Revs]
- 69th number: Checksum

APPENDIX C. GUIDE TO RUN THE PROJECT IN ADALM-PLUTO

This appendix describes how to use the software designed in the ADALM-PLUTO SDR.

C.1 Necessary materials

In order to use the software and test it several components are required. These are as follows:

- A computer or laptop with linux
- 2 ADALM PLUTO SDR devices

Figure C.1: ADALM PLUTO SDR

- 2 connectors USB to Micro-USB 2.0

Figure C.2: USB to Micro-USB connector

- 1 or 2 coaxial cables

Figure C.3: Coaxial Cable

- 2 or 4 ADALM-PLUTO antennas

Figure C.4: ADALM PLUTO Antennas

These cannot be used to transmit and receive in S-Band since it would be illegal, but can be used to carry out tests with other frequencies that can be used legally in the laboratory.

- An attenuator

Figure C.5: Attenuator

C.2 Cross Compilation

It should be noted that the project is compiled with make files. For it to be able to be used in the device, it must be cross-compiled with a specific make file. This is located in the highest level of the rita-comms folder with the filename ”pluto-crosscompile.cmake” and contains the following commands:

```
1  set(CMAKE_SYSTEM_NAME Linux)
2  set(CMAKE_SYSTEM_PROCESSOR arm)
3
4  # Specify the sysroot from which to obtain libraries and headers
5  set(CMAKE_SYSROOT "/opt/pluto-0.34")
6
7  # Specify the compiler and linker to use (they need to be in the ...
       PATH)
8  set(CMAKE_CXX_COMPILER /opt/linaro/bin/arm-linux-gnueabihf-g++)
9  set(CMAKE_CXX_LINKER   /opt/linaro/bin/arm-linux-gnueabihf-ld)
10
11 set(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
12 set(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
13 set(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
14 set(CMAKE_FIND_ROOT_PATH_MODE_PACKAGE ONLY)
15
```

```
16 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -mfloat-abi=hard ...
       -mfpu=neon")
```

In order to use this file, the cmake must be called with the following commands:

```
1 mkdir build && cd build
2 cmake -DCMAKE_TOOLCHAIN_FILE=../pluto-crosscompile.cmake ..
3 make
```

C.3 Execution of the project

In order to run the code in ADALM-PLUTO several steps must be made. Assuming that the previous cross compilation steps have been done and the computer contains the compiled libraries, the steps to follow are:

1. Connect the ADALM-PLUTO to the computer via USB.
2. Open a linux terminal and execute:

```
1 cd /opt/pluto-0.34/
2 ssh root@192.168.2.1
```

3. Copy the dynamic libraries and the executable file into the SDR. The project has several libraries that are necessary for its correct functioning. These are the following:

   - External libraries:
     - cppzmq
     - iiohandler
     - libcorrect
     - libzmq
     - Ne10
     - spdlog
   - Source libraries:
     - applayer
     - channel
     - phy

   All of the above are contained in dynamic library files (.so). There are several ways to copy the files, however, the one that has been used in the project has been to create a expect script. This can be called through the Linux command line with the following code (being copyFiles.exp the script):

```
1 sudo expect copyFiles.exp
```

Its code is the following:

```
1
2  #!/usr/bin/expect
3
4  # ./copyFiles.exp <password>
5  set password "analog"
6
7  #MUST MOVE IMPORTED LIBRARIES:
8  cd build
9  cd lib
10 cd iiohandler
11 #sshpass -p 'analog' scp libiiohandler.so ...
       root@192.168.2.1:/usr/lib/
12 #sshpass -o StrictHostKeyChecking=no scp libiiohandler.so ...
       root@192.168.2.1:/usr/lib/
13 #pscp -pw 'analog' libiiohandler.so root@192.168.2.1:/usr/lib/
14 spawn scp libiiohandler.so root@192.168.2.1:/usr/lib/
15 expect "password"
16 send "$password\r"
17 interact
18
19 cd ..
20 cd libcorrect/lib
21 spawn scp libcorrect.so root@192.168.2.1:/usr/lib/
22 expect "password"
23 send "$password\r"
24 interact
25
26 cd ../../
27 cd spdlog
28 spawn scp libspdlog.so root@192.168.2.1:/usr/lib/
29 expect "password"
30 send "$password\r"
31 interact
32
33 spawn scp libspdlog.so.1 root@192.168.2.1:/usr/lib/
34 expect "password"
35 send "$password\r"
36 interact
37
38 spawn scp libspdlog.so.1.9.2 root@192.168.2.1:/usr/lib/
39 expect "password"
40 send "$password\r"
41 interact
42
43 cd ..
44 cd Ne10/modules
45 spawn scp libNE10.so root@192.168.2.1:/usr/lib/
46 expect "password"
47 send "$password\r"
48 interact
49
50 spawn scp libNE10.so.10 root@192.168.2.1:/usr/lib/
51 expect "password"
52 send "$password\r"
53 interact
```

```
54
55 spawn scp libNE10_test.so root@192.168.2.1:/usr/lib/
56 expect "password"
57 send "$password\r"
58 interact
59
60 spawn scp libNE10_test.so.10 root@192.168.2.1:/usr/lib/
61 expect "password"
62 send "$password\r"
63 interact
64
65 cd ..
66 cd ..
67 cd libzmq/lib
68
69 spawn scp libzmq.so root@192.168.2.1:/usr/lib/
70 expect "password"
71 send "$password\r"
72 interact
73
74 spawn scp libzmq.so.5 root@192.168.2.1:/usr/lib/
75 expect "password"
76 send "$password\r"
77 interact
78
79 spawn scp libzmq.so.5.2.5 root@192.168.2.1:/usr/lib/
80 expect "password"
81 send "$password\r"
82 interact
83
84 cd ..
85 cd ..
86 cd ..
87
88 # MUST MOVE SRC LIBRARIES:
89 cd src
90 cd applayer
91 spawn scp libapplayer.so root@192.168.2.1:/usr/lib/
92 expect "password"
93 send "$password\r"
94 interact
95
96 cd ..
97 cd channel
98 spawn scp libchannel.a root@192.168.2.1:/usr/lib/
99 expect "password"
100 send "$password\r"
101 interact
102
103 cd ..
104 cd phy
105 spawn scp libphy.a root@192.168.2.1:/usr/lib/
106 expect "password"
107 send "$password\r"
108 interact
109
110 cd ..
111 cd ..
```

```
112
113 # MUST MOVE rita-comms LIBRARIES:
114 cd app
115 spawn scp rita-comms root@192.168.2.1:/root
116 expect "password"
117 send "$password\r"
118 interact
```

4. Run the code in the ADALM-PLUTO with following commands:

```
1 cd /
2 cd ./root/rita-comms
```

5. Repeat for the other ADALM PLUTO device.

C.4 CPU configuration of ADALM-PLUTO

In order to use more efficiently the ADALM-PLUTO, both CPU cores can be used. This device does not have this configuration activated by defect, therefore, the user must change its predetermined configuration to the dual core usage. This can be done with the following code, which results in the configuration depicted in figure C.6.

```
cat /proc/cpuinfo
> 1 CPU
fw_setenv maxcpus
pluto_reboot reset
...
cat /proc/cpuinfo
> 2 CPUs
```

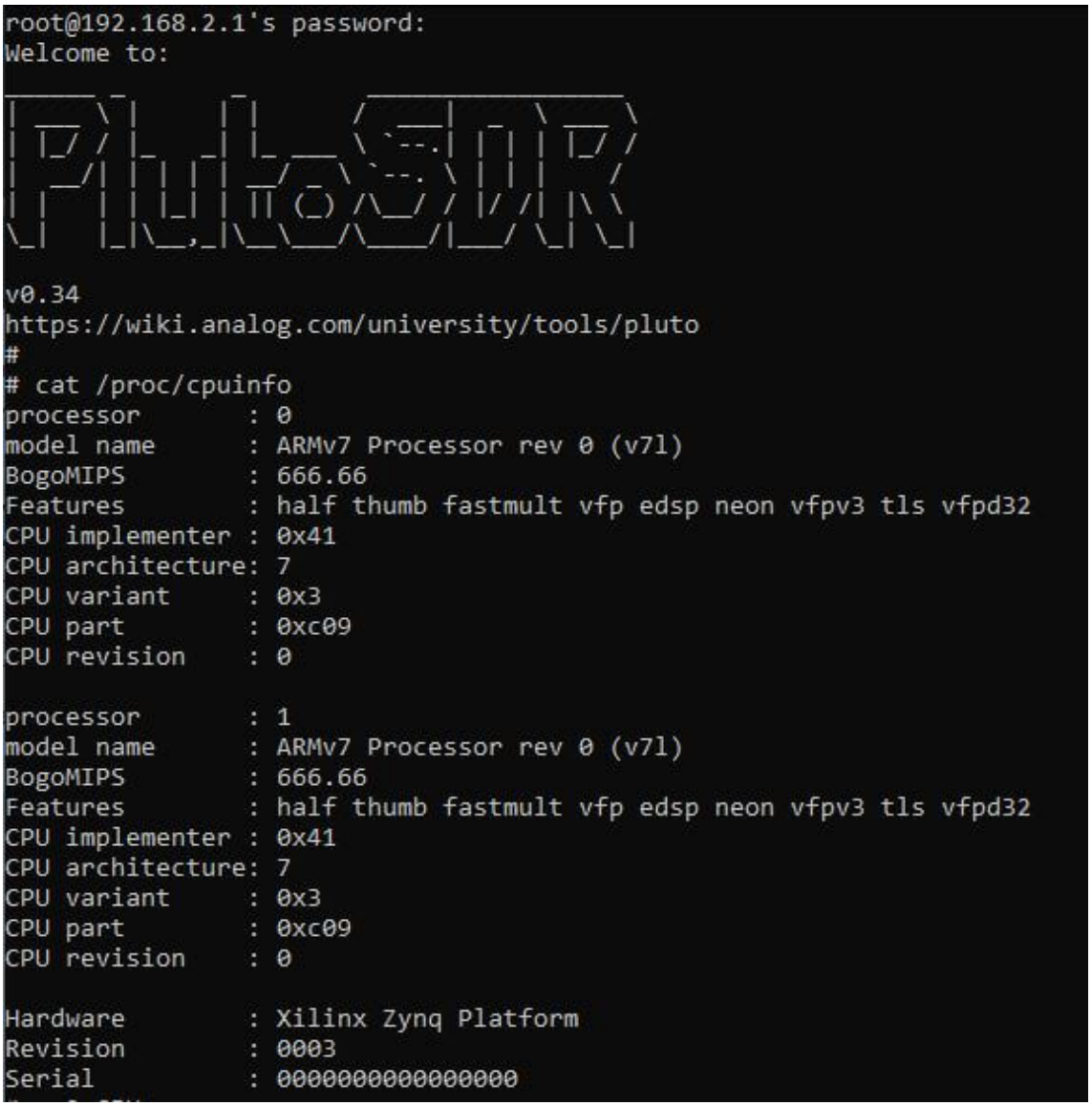


Figure C.6: ADALM-PLUTO 2 CPU configuration

APPENDIX D. SOFTWARE IMPLEMENTATION

This document contains an schematic explanation of the code developed for the implementation (contained in the *src* folder). These can be split into three main code blocks:

- Application Layer and Network Code
- Channel Coding Code
- Physical Channel Code

It should be noted that, along the code, the logging library ”spdlog” has been used to keep track of the processes carried out and inform of errors that might occur. This library can be accessed in [108].

D.1 Application Layer and Network Code

In order to implement the application layer and network, several files have been created. These are as follows:

- **JsonUtils.cpp and JsonUtils.hpp**

  These files have been implemented to deal with the JSON data handler file. Some of the functions that have been designed involve:

  - Getting a vector with the names of the files to be sent
  - Getting a vector with all the names of the files that have received an ACK message
  - Adding a given filename to a JSON file.
  - Changing the status of a given filename of a JSON file.
  - Erasing a given filename from a JSON file.
  - Reading a JSON file and comparing it with the existing files in a given folder to detect the new filenames to be added.
  - Test functions to check the correct functioning of the previous functions

  In order to implement this language in the code, a JSON library have been used. There are several JSON open source libraries available in GitHub, however, the ”JSON for Modern C++” has been chosen for it’s intuitive syntax and the testing that it has undergone. Moreover, it’s a fast and memory efficient library . This can be accessed in GitHub [104].

- **fileUtils.cpp and fileUtils.hpp**

  These files contain functions that are used to deal with the managing of files. Some of the functions are the following:

  - Get a vector containing the files present in a given folder.

- Get the number of files present in a folder.
- Split a file with a given split size into several smaller files.
- Compress a file and split it into several smaller files and move them to a specific folder.
- Join several files.
- Decompress a file.
- Check if a directory is empty.
- Delete a given file.
- Delete the files that have been stored in a JSON file with the attribute ACK.
- Save an array of uint8_t data into a file, stored in a given folder.
- Check if a file is present in a folder given its filename.
- Test functions to check the correct functioning of the previous functions

Moreover, this files also contain functions to deal with the TOML configuration file. The functions that deal with TOML are stated below. It should be noted that in order to implement this language in the code, a TOML configuration parser and serializer for C++ has been used. The particular library used is "toml++", and can be found on Github [103].

- Print the information of a TOML file.
- Get parameters from a TOML file.

- **HeaderUtils.cpp and HeaderUtils.hpp**

  These files contain the functions that deal with the message header, which has been detailed in the previous sections. Some of the implemented functions are:

  - **addHeader**: This function creates the header of the message from a given filename and adds it to the message.
  - **createHeader**: This function creates the header of the message from a given filename.
  - **decodeHeader**: As its name states, this function carries decodes the header of a given message.
  - **removeAndDecodeHeader**: As its name states, this function removes the header of a message and decodes it.
  - **codificationPart_separateNum and codificationPart_unicNum**: This functions are used when creating the headers. They particularly codify split part of the initial file. Both of them work the same way but take the input in different manners.
  - **decodificationPart_separateNum and decodificationPart_unicNum**: Just like the previous functions, these decode the split part of the initial file.

- **ULUtils.cpp and ULUtils.hpp**

  These files contain the functions that are used for the uplink telemetry. Some of the implemented functions are the following:

- **createDLrequestFrame**: As its name states, this function creates the downlink request frame.
- **createRELAYrequestFrame**: As its name states, this function creates a relay request frame
- **createACKFrame**: This function creates an ACK frame.
- **createNACKFrame**: This function creates a NACK frame. It is not used, but it has been designed should the ARQ protocol be changed.
- **readULCommand**: This function determines the type of command that has been received in the satellite.
- **getRelayCommand**: This function saves the relay command in a uint8_t array.

- **appUtils.cpp and appUtils.hpp**

  These files contain the most important functions of the application layer and call the functions contained in JsonUtils and fileUtils. Some of the implemented functions are the following:

  - **prepare_files**: This function is designed to be used in the satellite. It accesses the folders ”prioriy” and ”results” of the satellite. If they are not empty, it compresses them. Then it splits them and moves them to the folder ”in_queue”. Finally, it saves the filenames that have been added in the folder to the JSON data handler file.
  - **activate_prepare_downlink_files**: This function is designed to be used in the satellite. It gets the files from a directory, compresses them and splits them into new smaller files with a given size. Then, it moves the resulting files to the ”in_queue” folder. Finally, it adds the new files to the data Handler JSON file.
  - **GS_save_Packet_in_File**: This function is designed to be used in the ground station. It saves the files in the adequate reception data folder. Moreover, it creates a specific folder with the name of the file it it doesn't exist. Therefore, all the received files that have been split into parts will be stored in their particular general folder.
  - **GS_join_decompress_and_store_files**: This function is designed to be used in the ground station to join files, which had been split in the satellite previous to the transmission of the data. Finally, it decompresses the file and moves it into a given folder.

D.2 Channel Coding Code

In order to implement the channel codification, several files have been created. These are as follows:

- **BasicUtils.cpp and BasicUtils.hpp**

  These files contain functions that carry out small tasks that are useful and have been used along the channel coding development and have been implemented in some higher level functions. Some of the functions are the following:

  - Obtain a vector of uint8_t with the contents of a given file.
  - Save an array of uint8_t in a tar file.
  - Convert from uint8_t to uint16_t.
  - Convert from uint16_t to uint8_t.
  - Convert from binary to hexadecimal.
  - Convert from hexadecimal to binary.
  - Convert from string to char.

- **ConvolutionalUtils.cpp and ConvolutionalUtils.hpp**

  These files contain functions that carry out the convolutional endoding and decoding. This process has a very complex software implementation, therefore, a library has been used in order to save development time as well as to ensure coding efficiency and robustness.

  The convolutional concatenation has been implemented by making use of the ”libcorrect” library. This has an efficient implementation that is adequate for fast-processing. This library can be accessed in [95].

  Using the aforementioned library requires several steps. Therefore, two functions have been designed in order to ease its use. These are the following:

  - **activate_convolutional_encoding**: As its name states, this function carries out the convolutional codification.
  - **activate_convolutional_decoding**: As its name states, this function carries out the convolutional decodification.

- **InterleavingUtils.cpp and InterleavingUtils.hpp**

  These files contain functions that carry out the interleaving and deinterleaving of a message. Since the implementation of the this technique does not require extensive algorithms, it has been coded without the use of an external library, and can be used with the following two functions:

  - **interleave**: As its name states, this function carries out the interleaving of a given message.
  - **deinterleave**: As its name states, this function carries out the deinterleaving of a given message.

- **ReedSolomonUtils.cpp and ReedSolomonUtils.hpp**

  These files contain functions that carry out the Reed-Solomon coding and decoding. Just like with the Convolutional Concatenated codification, this process has a very complex software implementation, therefore, a library has been used in order to save development time as well as to ensure coding efficiency and robustness. The chosen library is ”libcorrect”, and can be accessed in [95].

  In order to use this library easily two main functions have been created using several functions provided by the library. These are as follows:

- **activate_reed_solomon_encoding**: As its name states, this function carries out the Reed-Solomon codification of a given message.
- **activate_reed_solomon_decoding**: As its name states, this function carries out the Reed-Solomon decodification of a given message.

- **ChannelCodingUtils.cpp and ChannelCodingUtils.hpp**

  These files contain the most important functions of the codification and use the functions contained in the previous files. Some of the implemented functions are the following:

  - **encodeFile_Down**: This is a very important function that encodes the file to be sent. It carries out the following steps:
    1. From a filename, obtains an array of uint8_t containing the data in the file.
    2. The array is transformed into a vector and padding is added if needed.
    3. The header of the packet is added.
    4. The Reed-Solomon codification is carried out.
    5. The message is interleaved.
    6. The convolutional coding is carried out.
  - **encodeFile_Up**: This, analogously to the previous function encodes the uplink telemetry frame to be sent without adding a header.
  - **decodeFile_Down**: This is a very important function that decodes a received message. It carries out the following steps:
    1. The decoding of the convolutional coding is carried out.
    2. The message is deinterleaved.
    3. The Reed-Solomon decoding is carried out.
    4. The header is decoded
    5. The padding is removed
  - **decodeFile_Up**: This is a very important function that decodes a received message from the uplink, returning the transmitted original telemetry message command.
  - **add_channel_errors**: This function has been implemented for testing purposes and adds errors to a given uint8_t array simulating errors that might occur in the physical channel.

D.3 Physical Channel Code

In order to implement the physical layer codification, several files have been created. These are as follows:

- **CostasUtils.cpp and CostasUtils.hpp** These files contain the functions used to carry out the Costas Loop.

  - **activateCostasLoop**: This function carries out the Costas Loop.

  - **recoverbits**: This function is used in ”activateCostasLoop” to help manage the conversion from signals to bits by joinig individual recovered bits from the loop into a bytes array.

- **SynchroUtils.cpp and SynchroUtils.hpp** These files contain the functions used to carry out the synchronization of the signal.

  - **detectSynchWord**: This function is used for synchronizing a signal and requires the ”normalizedCrossCorrelation” function.
  - **normalizedCrossCorrelation**: This function computes the correlation between to vectors and returns the correlation normalized value.
  - **multiply_conjugate_dot_product**: This function computes the conjugate dot product of two vectors. To do so, it uses the ”sumValuesVector” and ”conjugate” function.
  - **sumValuesVector**: This function returns the sum of all the values in a vector.
  - **conjugate**: This function conjugates a vector.
  - **energy_normalization**: This function uses the ”sumValuesVector” and ”vector_magnitude_squared” to normalize the respective ”multiply_conjugate_dot_product”.
  - **accumulator**: Sums the values of one array with the same index values of another array and returns a final array.
  - **vector_magnitude_squared**: This function computes an array containing the squared magnitude of a real and a complex initial arrays.
  - **division_absolute**: This function computes the correlation value from the results of the ”multiply_conjugate_dot_product” and the ”energy_normalization”.

- **ModDemod.cpp and ModDemod.hpp**

  These files contain the functions to modulate and demodulate a signal.

  - **mod_BPSK**: This function carries out the BPSK modulation, converting an array of bits into an array of samples.
  - **demod_BPSK**:This function carries out the BPSK demodulation, converting an array of samples into an array of bits.

- **ConfPUtils.cpp and ConfPUtils.hpp** These files contain the function ”confPulse_SRRC” that allows to multiply a signal by a train of SRRC conformation pulses.

- **SamplingUtils.cpp and SamplingUtils.hpp**

  These files contain the functions to sample the received signal once it has been processed.

  - **sample**:This function samples a given array of samples depending on the Baud rate of the transmitted signal.
  - **sample_IQ**:This function samples a given two arrays of samples, a real one and an imaginary one, depending on the Baud rate of the transmitted signal.

- **sample_complex**: This function samples a given complex array of samples depending on the Baud rate of the transmitted signal.

- **TransmitUtils.cpp and TransmitUtils.hpp**

  These files contain the functions used for the transmission of frames. Moreover, it should be noted that they require the use of the IIOHandler library, as has been stated previously.

  - **TX_test**: This function has been designed to test the transmission functioning. It configures the iio library and fills the transmission buffer with a given array.
  - **activateTransmisionCOSINE_test**: This function has been designed to test the transmission functioning. It works in the same manner as the previous but transmits a cosine.
  - **activateTransmision**: This function is used for the transmission of a given array. This is done with the following steps:
    1. Get synchronization word, modulate it and multiply it by a train of conformation pulses.
    2. Get the message to be sent, modulate it and multiply it by a train of conformation pulses.
    3. Configure iio library parameters and fill the transmission buffer
    4. Transmit
  - **confPulse**:This function multiplies an array of samples with a train of conformation pulses.
  - **GetSynchWordMod**: This function has been designed for practical purposes. It defines the CCSDS Synchronization word and returns an array with the modulated word.

- **ReceiverUtils.cpp and ReceiverUtils.hpp**

  These files contain the functions used for the reception of frames. Moreover, it should be noted that they require the use of the IIOHandler library, as has been stated previously.

  - **RX_test**: This function has been designed to test the reception functioning. It configures the iio library and fills the reception buffer, storing it in an array unit it is stopped.
  - **activateReceptionSamples_Costas**: This function is designed to test the costas loop implementation. It configures the iio library and fills the reception buffer. Then the synchronization is carried out and if a packet is detected, the costas loop is carried out, returning the demodulated bits. The whole process is carried out as many times as specified by the user.
  - **activateReception**: This function carries out the reception of signals with all the necessary steps, returning an array of bits.

It should be noted that the most part of these functions require the use of the external library NE10 [107], which can be accessed in GitHub in [106]. This is a library that has been

optimized for ARM architectures, such as the one that the code will be run in. It contains useful functions that have been optimized and are much more efficient that if they where implemented directly on C++. Therefore, by using this library, the code's computations are faster and more efficient.

APPENDIX E. TRANSMISSION AND RECEPTION SOFTWARE CONFIGURATION

In order to configure the physical device (ADALM-PLUTO SDR) the following the following code is used, where the gain of the receiver and transmitter can be adapted as well as the size of the buffers.

```
IIOHandler handler;
IIOHandler::stream_cfg config[2];

config[0].bw_hz = MHZ(2.048);  // RX Bandwidth
config[0].fs_hz = MHZ(2.048 * 4); // RX sampling frequency
config[0].gain = 50; // RX gain
config[0].lo_hz = MHZ(2279); // RX local oscillator
config[0].port = IIOHandler::PORT_RXA; // RX port definition
config[0].buffer_size = (2048 * 10); // RX buffer size

config[1].bw_hz = MHZ(2.048); // TX Bandwidth
config[1].fs_hz = MHZ(2.048 * 4); // TX sampling frequency
config[1].gain = 0; // RX gain
config[1].lo_hz = MHZ(2279); // RX port definition
config[1].port = IIOHandler::PORT_TXA; // TX port definition
config[1].buffer_size = (2048 * 10); // TX buffer size


int result = handler.init(1, 1, true, LOCAL_CONTEXT); // ...
    initialize the Handler
if (result != 0)
{
    spdlog::info("Error initializing handler"); // print error
}

bool load_my_config = handler.loadConfig(config); // load the ...
    configuration
if (!load_my_config)
{
    spdlog::info(" Error loading handler configuration."); // ...
        print error
}
```

Since it is particularly tricky to get the ADALM-PLUTO SDR to transmit and receive as desired from a C++ code, the functions used to do so are written below.

- Transmit function

  In order to transmit from the ADALM-PLUTO, the following function can be used, as long as it is configured with the previous code.

```
bool IIOHandler::pushTXBufferAsComplex(Ports port, ...
    int_complex *samples_complex, size_t sample_count)
{
    char *p_dat, *p_end;
```

```
    ptrdiff_t p_inc;
    iio_channel *chan;
    switch (port)
    {
    case PORT_TXA:
        chan = txa_i;
        break;
    case PORT_TXB:
        chan = txb_i;
        break;
    default:
        return false;
    }

    int cont = 0;

    ssize_t nbytes_tx = iio_buffer_push(tx_buf);
    if (nbytes_tx < 0)
    {
        printf("Error pushing bufer \n");
    }

    // WRITE: Get pointers to TX buf and write IQ to TX buf ... port 0
    p_inc = iio_buffer_step(tx_buf);
    p_end = (char *)iio_buffer_end(tx_buf);

    for (p_dat = (char *)iio_buffer_first(tx_buf, txa_i); ... p_dat < p_end; p_dat += p_inc)
    {
        ((int16_t *)p_dat)[0] = samples_complex[cont].real; ... // Real (I)
        ((int16_t *)p_dat)[1] = samples_complex[cont].imag; ... // Imag (Q)
        cont++;
    }
    if (sample_count == 0)
        return true;
    else
        return false;
}
```

- Receive function

  In order to receive from the ADALM-PLUTO, the following function can be used, as long as it is configured as stated previously.

```
bool IIOHandler::fillRXBufferAsRI(Ports port, int16_t ... *samples_real, int16_t *samples_imag)
{
    switch (port)
    {
    case PORT_RXA:
        m_rx_chan = rxa_i;
        break;
    case PORT_RXB:
        m_rx_chan = rxb_i;
```

```
10          break;
11      default:
12          return false;
13      }
14
15      // Refill RX buffer
16      if (iio_buffer_refill(rx_buf) < 0)
17          return false;
18
19      m_samp_cnt = 0;
20      m_rx_buffer_increment = iio_buffer_step(rx_buf);
21      m_rx_buffer_end = iio_buffer_end(rx_buf);
22
23      for (char *p_dat = (char *)iio_buffer_first(rx_buf, ...
            m_rx_chan); p_dat < (char *)m_rx_buffer_end;
24           p_dat += m_rx_buffer_increment)
25      {
26          samples_real[m_samp_cnt] = ((int16_t *)p_dat)[0];  ...
                // Real (I)
27          samples_imag[m_samp_cnt] = ((int16_t *)p_dat)[1];  ...
                // Imag (Q)
28          m_samp_cnt++;
29      }
30      return true;
31  }
```

APPENDIX F. COSTAS LOOP TEST RESULTS

This appendix contains some of the results of several tests that were carried out in order to verify the correct functioning of the Costas Loop implementation.

These tests have been carried out with Matlab and consist of the following:

1. A random array of bits is declared
2. The bits are modulated with BPSK. This can be seen in figure F.1, which shows clearly how the *in-phase* signal contains the bit information.

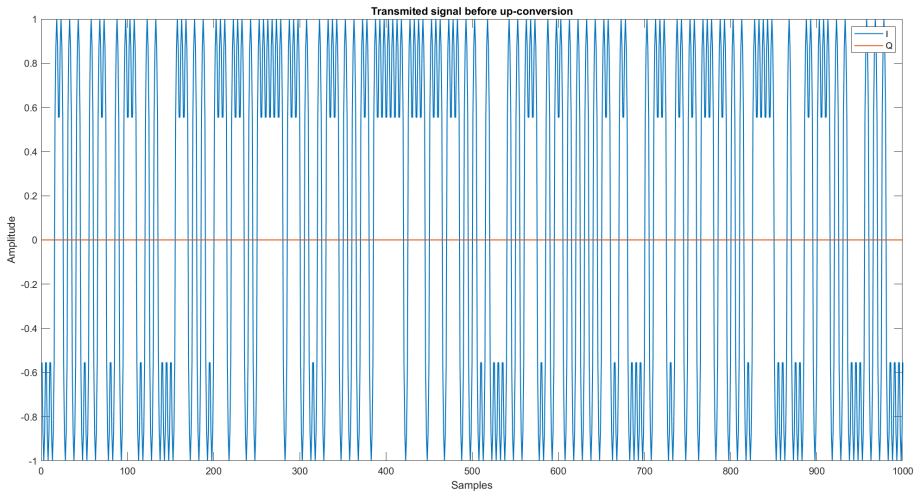


Figure F.1: Transmitted modulated signal

3. This signal has been up-converted with a 20kHz and a 40 kHz frequency. These way, the Doppler frequency shift that the receiver will detect is simulated.

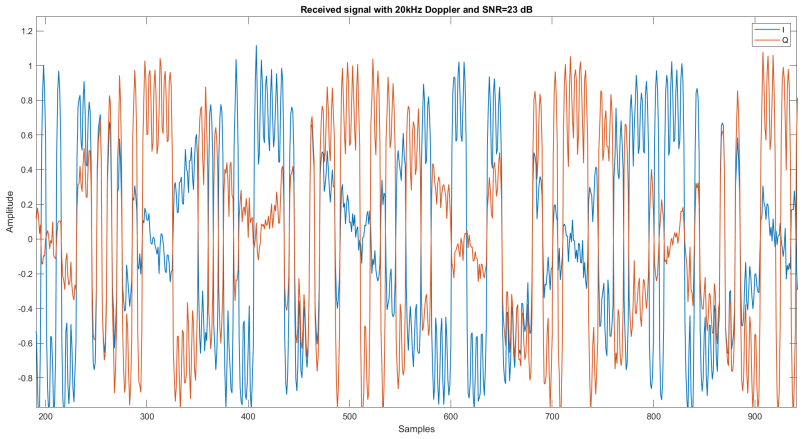
In order to test how the Costas Loop will behave in real conditions, noise must be added to the signal. This experiment considers this and has carried out 3 tests per Doppler frequency, generating random white noise signals correspondent to 9, 16 and 23 dB of Signal-to-Noise Ratio. This specific SNR have been chosen because, as has been computed in the link budget, the worst case scenario corresponds to getting a SNR of 9 dB, and in the best case (when the satellite is on top of the ground station), of SNR 23 dB.

With these two maximum and minimum values and by adding an intermediate, a better understanding of the functioning of the system can be gained.

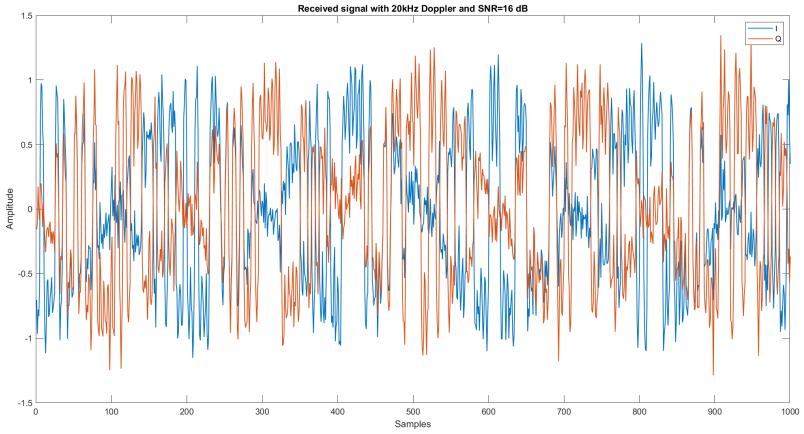
The results of the test are contained in the following sections:

F.1 Test for a Doppler frequency of 20 kHz

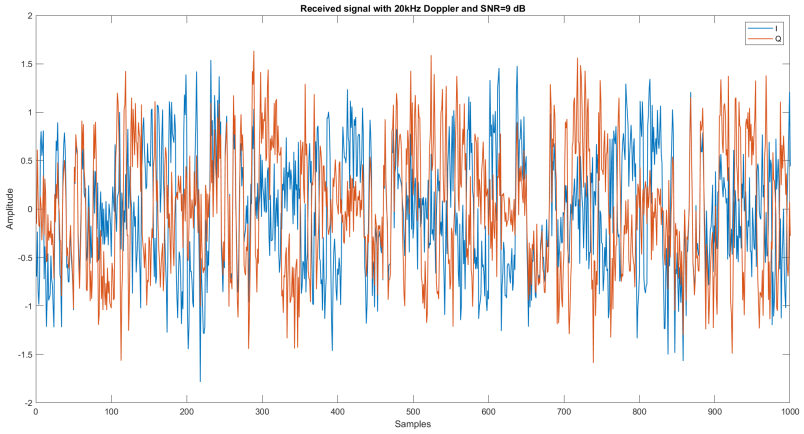
The following images depict the signal that the Costas Loop would receive for all SNR cases. As it can be seen, and is to be expected, the lower the SNR is, the more distorted the signal is by the noise.



(a) SNR=23 dB



(b) SNR=16 dB



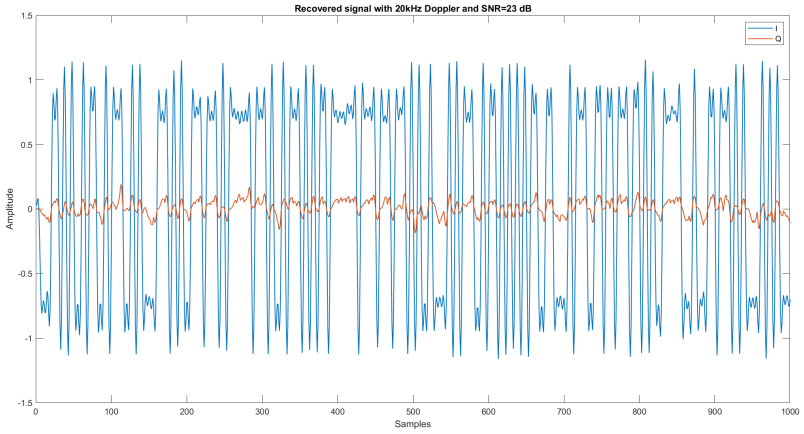
(c) SNR=9 dB

Figure F.2: Costas Loop received signal for a Doppler frequency shift of 20 kHz

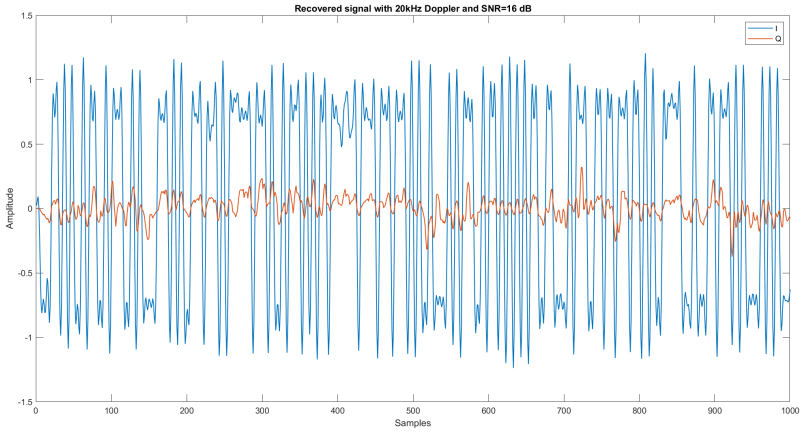
The following images depict the recovered signal by the Costas Loop for all SNR cases. This signal should match that of figure F.1. However, since this system is not ideal and it has noise, only an estimation can be expected.

As it can be seen, the quadrature component of the signal is not null as it should be, and instead it contains noise mixed with the in-phase component. The smaller this component is, the better the system. For these reason, the following figures will present better results

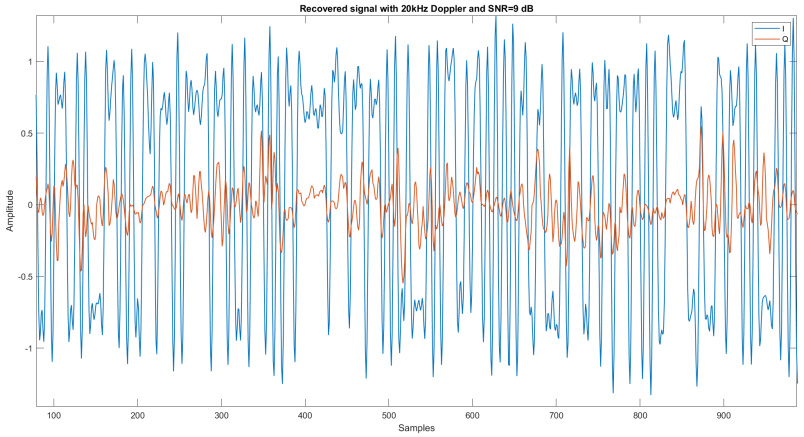
for higher values of SNR. Nevertheless, these results are all acceptable and adequate.



(a) SNR=23 dB



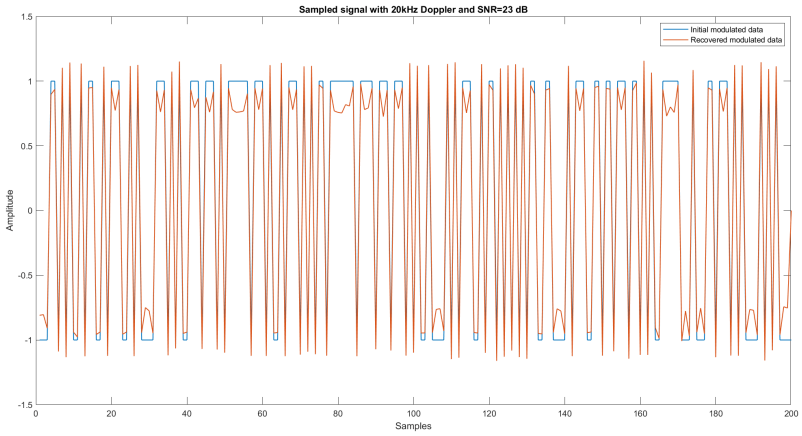
(b) SNR=16 dB



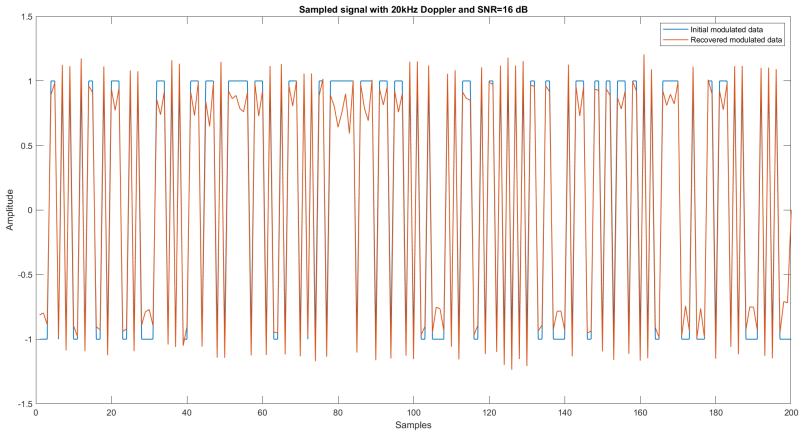
(c) SNR=9 dB

Figure F.3: Costas Loop recovered signal for a Doppler frequency shift of 20 kHz

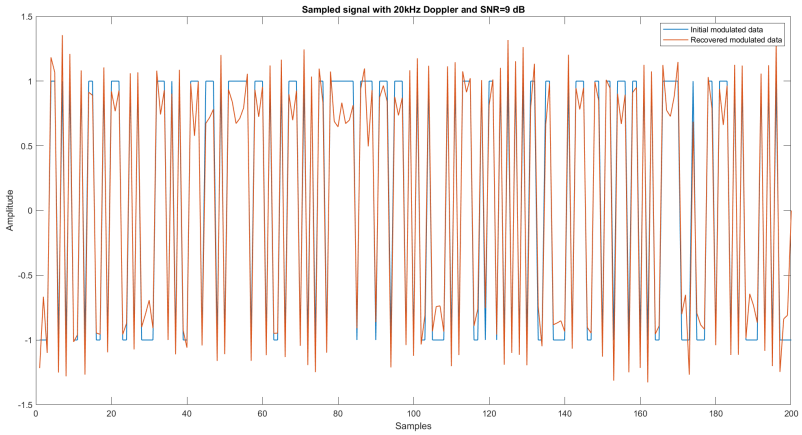
The following figures present a comparison between the modulated data and the recovered signal after sampling. As it can be seen, the recovered data is affected by noise, but follows clearly the same course as the transmitted signal.



(a) SNR=23 dB



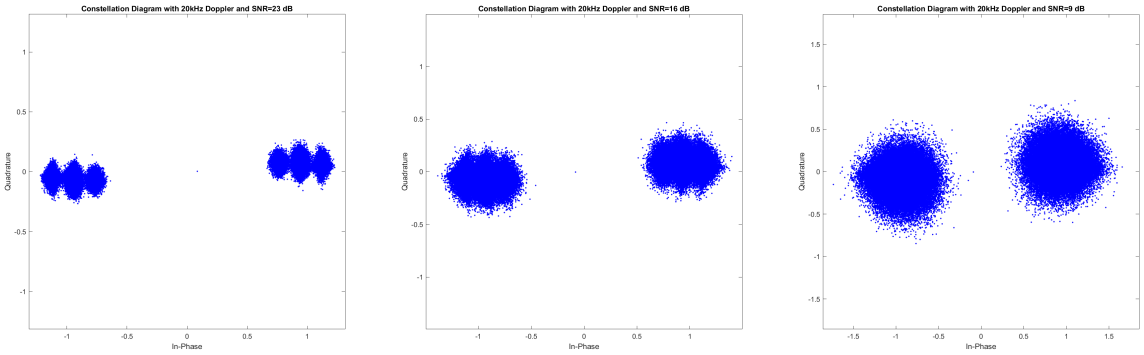
(b) SNR=16 dB



(c) SNR=9 dB

Figure F.4: Costas Loop sampled signal for a Doppler frequency shift of 20 kHz

Once the signal is sampled, as has been done in the previous step, the signal must be then demodulated to recover the bits. Figures F.5 present the constellations obtained, which have the expected BPSK format. Moreover, as it is to be expected, the higher the noise, the more disperse the two clutters are. However, all the figures present two clear and differential clutters that indicate a good reception and demodulation of the received signal.

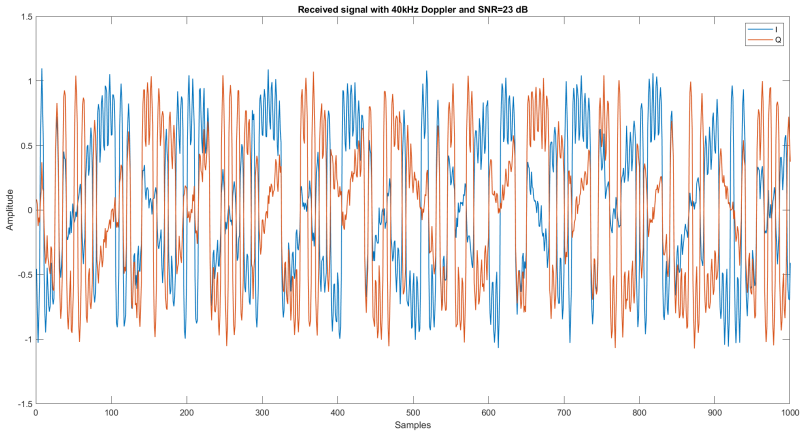


(a) SNR=23 dB (b) SNR=16 dB (c) SNR=9 dB

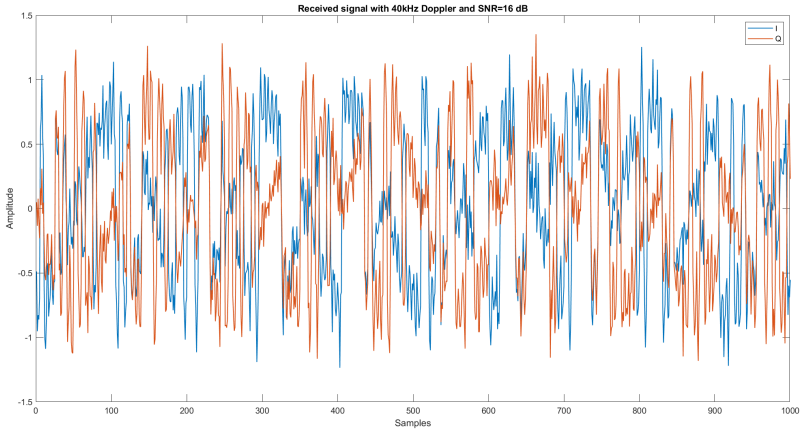
Figure F.5: Costas Loop constellation for a Doppler frequency shift of 20 kHz

F.2 Test for a Doppler frequency of 40 kHz

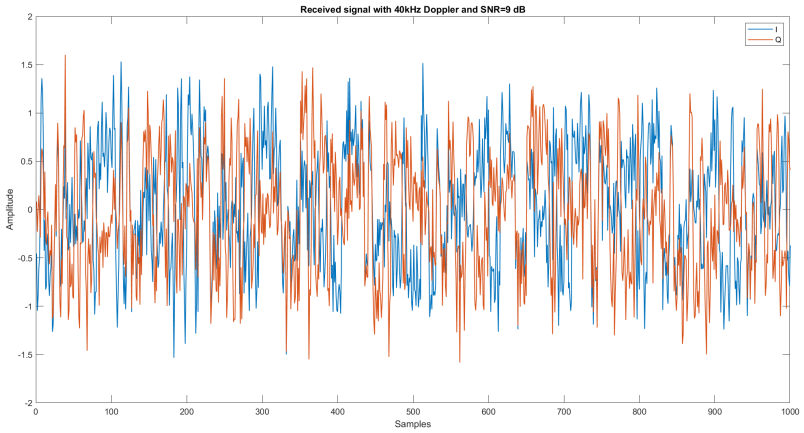
This test presents the same results as the previous, but a higher Doppler frequency shift. These results then show how the Loop is functional for very high shift frequencies that the communication link will have to deal with.



(a) SNR=23 dB

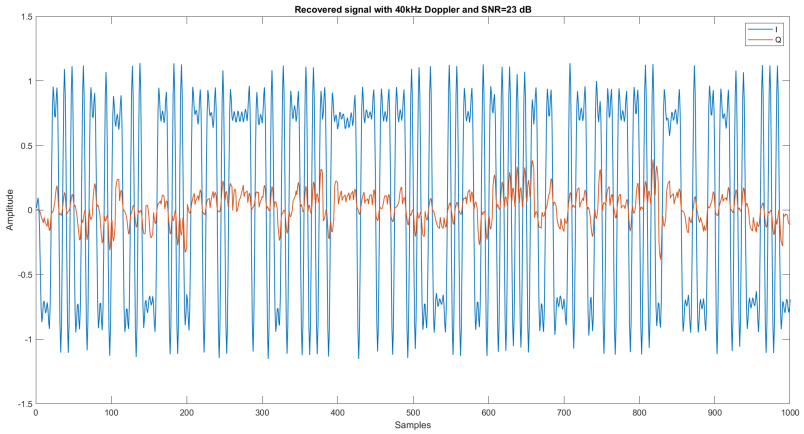


(b) SNR=16 dB

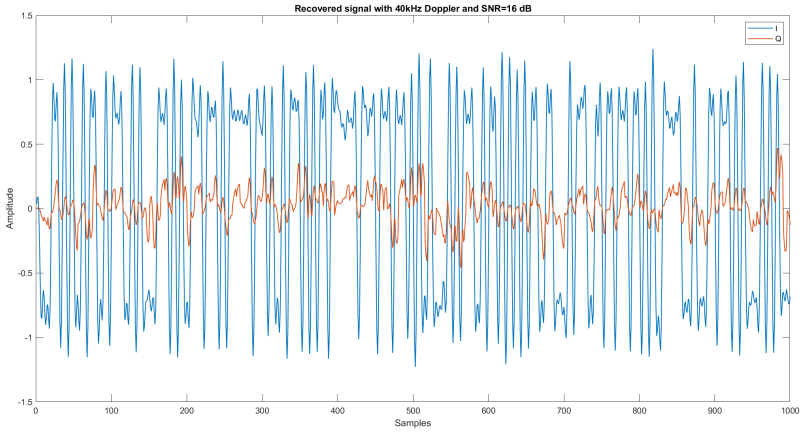


(c) SNR=9 dB

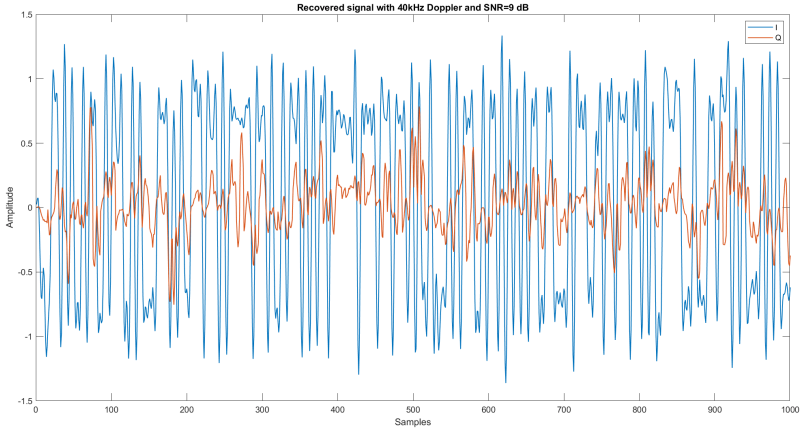
Figure F.6: Costas Loop received signal for a Doppler frequency shift of 40 kHz



(a) SNR=23 dB

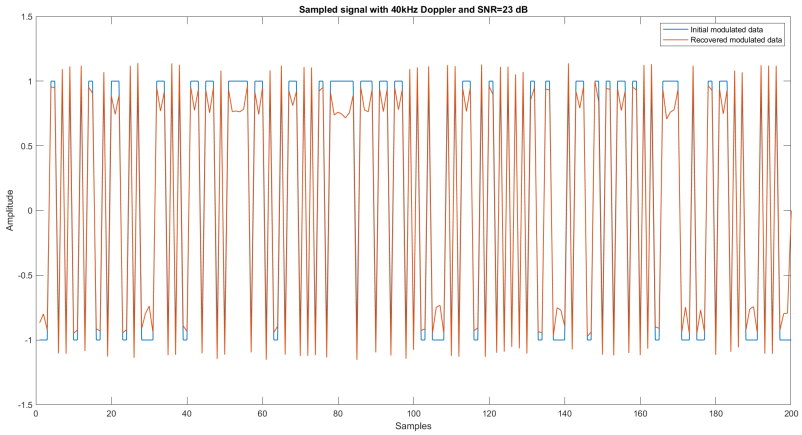


(b) SNR=16 dB

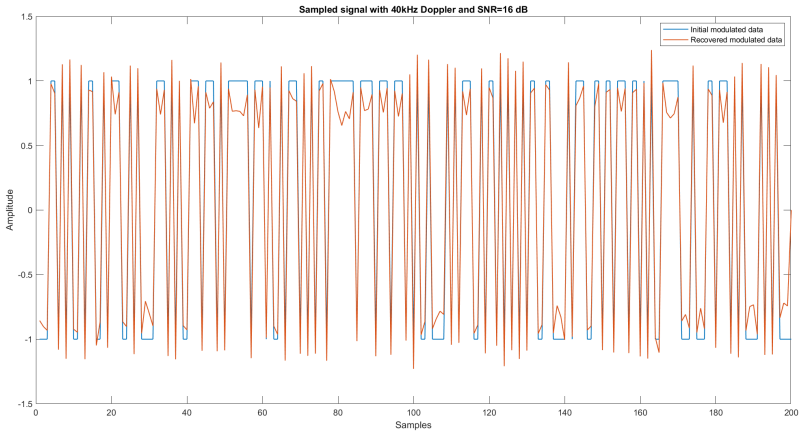


(c) SNR=9 dB

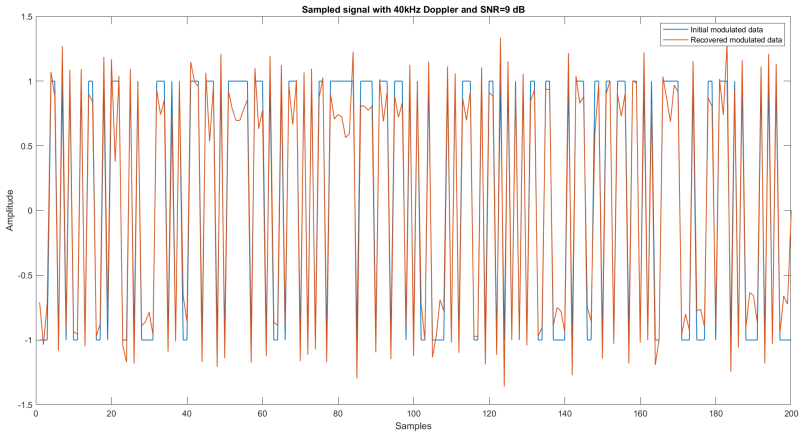
Figure F.7: Costas Loop recovered signal for a Doppler frequency shift of 40 kHz



(a) SNR=23 dB

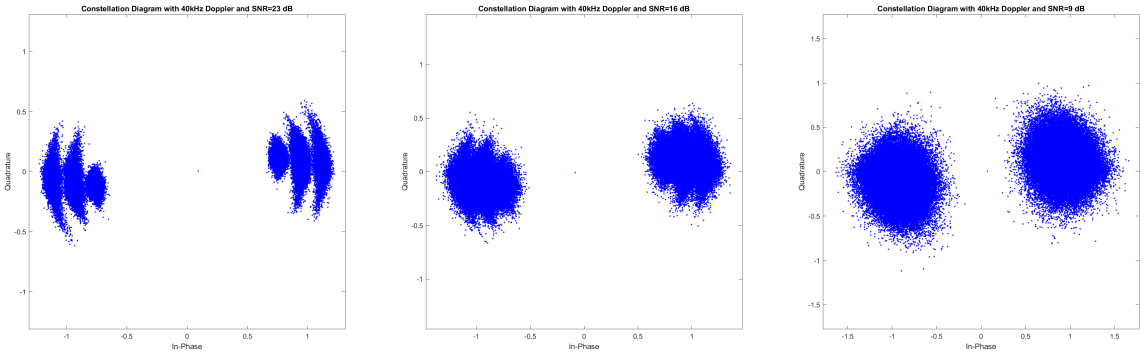


(b) SNR=16 dB



(c) SNR=9 dB

Figure F.8: Costas Loop sampled signal for a Doppler frequency shift of 40 kHz

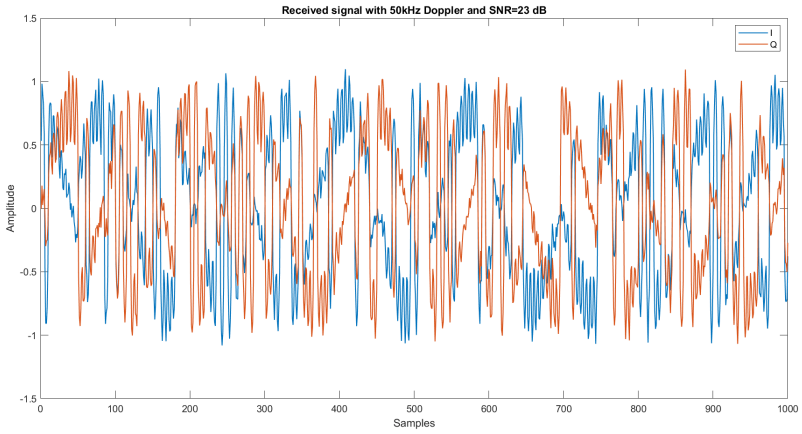


(a) SNR=23 dB (b) SNR=16 dB (c) SNR=9 dB

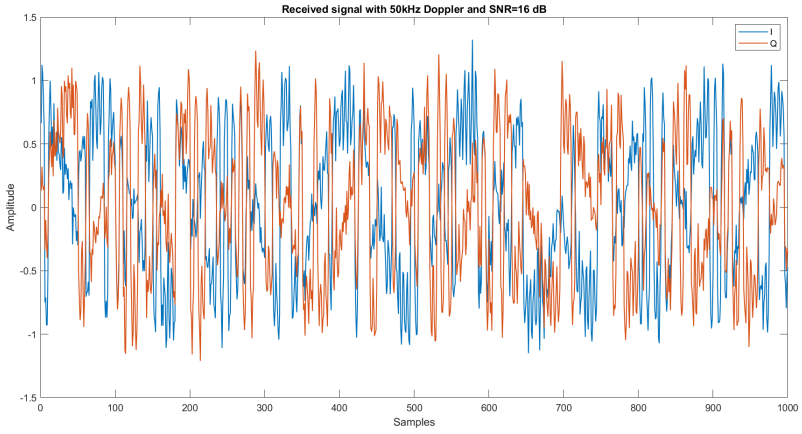
Figure F.9: Costas Loop constellation for a Doppler frequency shift of 40 kHz

F.3 Test for a Doppler frequency of 50 kHz

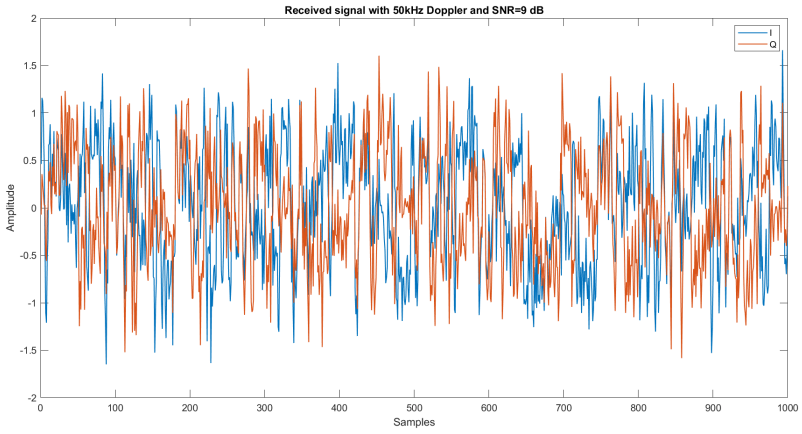
This test presents the same results as the previous, but for the maximum Doppler frequency shift that the system expects. These results then show how the Loop is functional for the range of shift frequencies that the communication link will have to deal with.



(a) SNR=23 dB

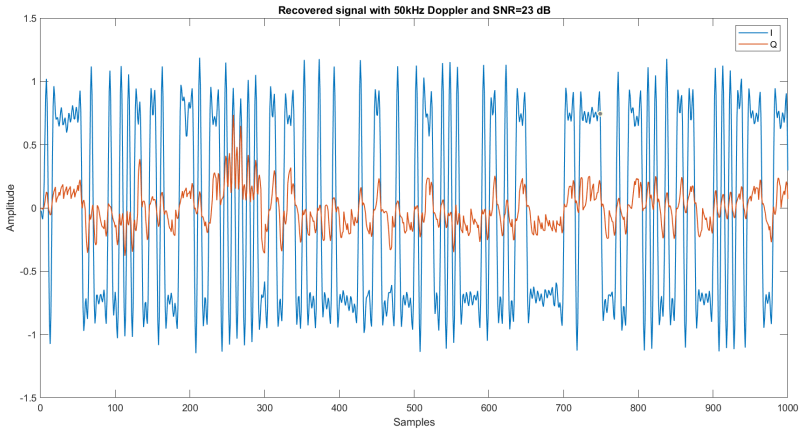


(b) SNR=16 dB

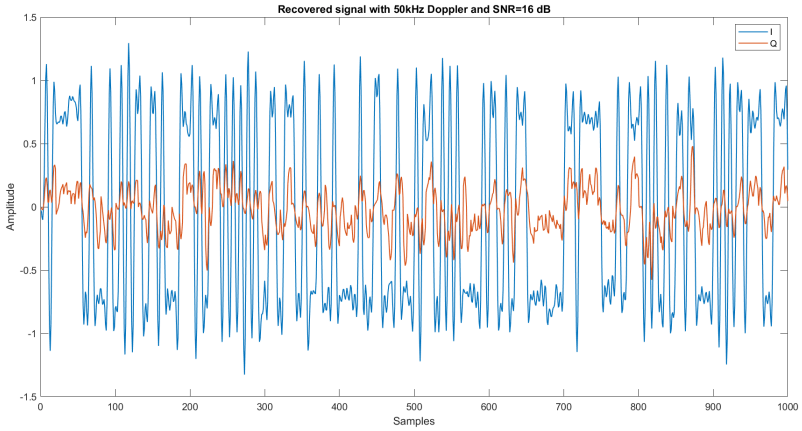


(c) SNR=9 dB

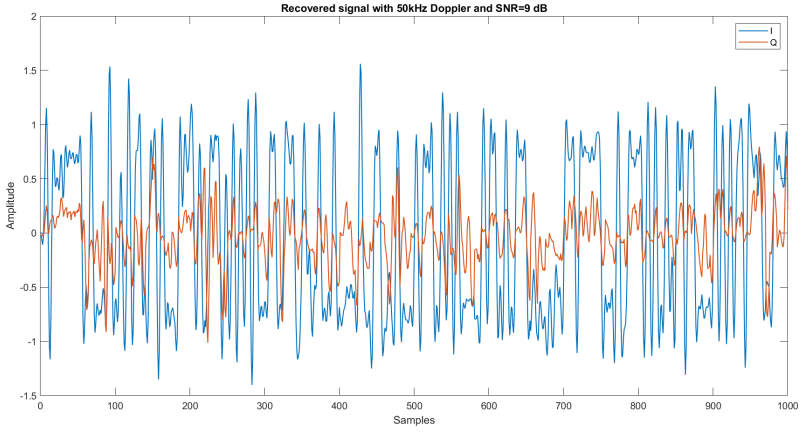
Figure F.10: Costas Loop received signal for a Doppler frequency shift of 50 kHz



(a) SNR=23 dB

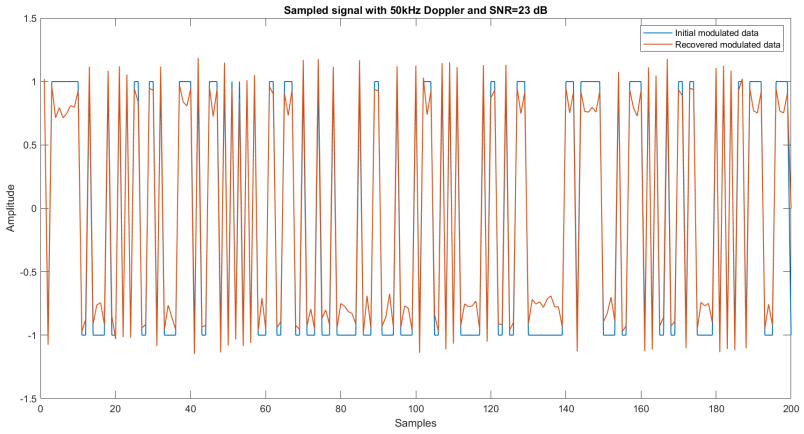


(b) SNR=16 dB

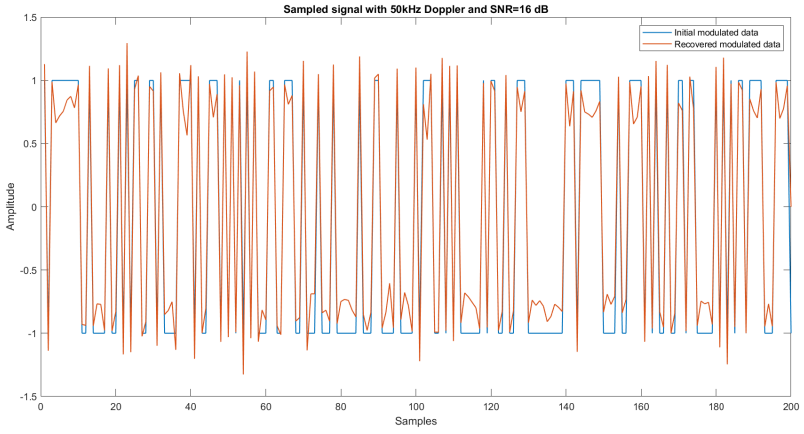


(c) SNR=9 dB

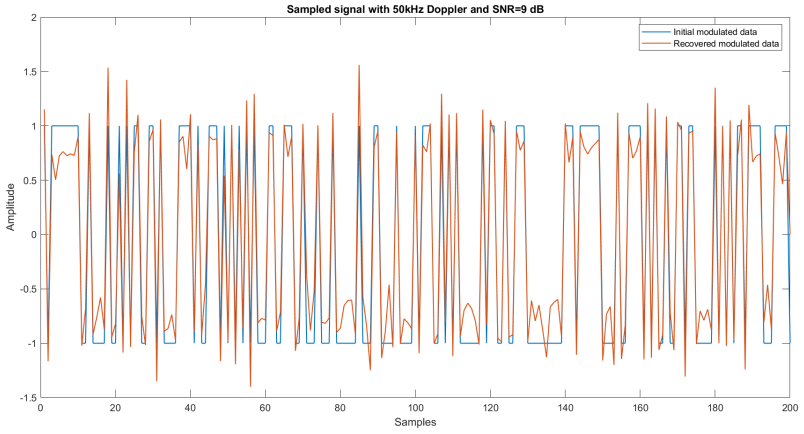
Figure F.11: Costas Loop recovered signal for a Doppler frequency shift of 50 kHz



(a) SNR=23 dB

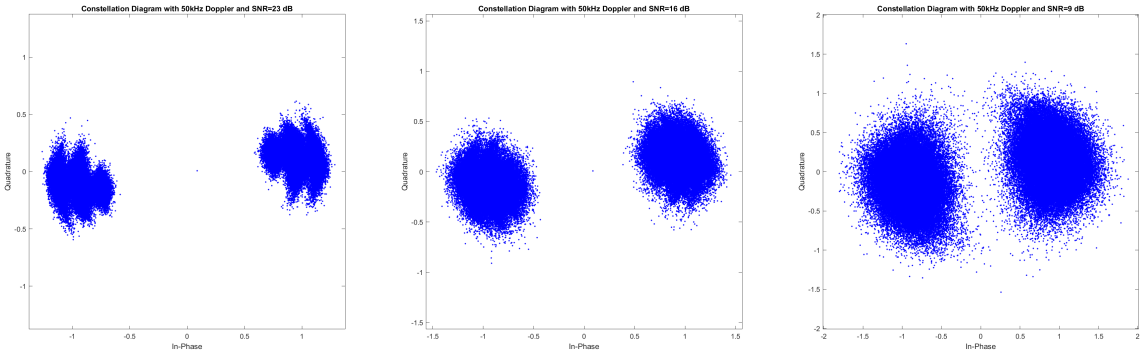


(b) SNR=16 dB



(c) SNR=9 dB

Figure F.12: Costas Loop sampled signal for a Doppler frequency shift of 40 kHz



(a) SNR=23 dB (b) SNR=16 dB (c) SNR=9 dB

Figure F.13: Costas Loop constellation for a Doppler frequency shift of 50 kHz

APPENDIX G. CHANNEL ENCODING AND DECODING TESTS

This appendix provides the channel encoding test results.

G.1 Encoding and decoding of a message

This section contains the tests results of the encoding and decoding of a message:

”0 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 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199”

G.1.1 Periodical errors every 20 bytes

This test has been carried out changing a byte every 20 bytes for a 33.

The console of the test is shown below:

```
{
[2022-06-19 12:32:07.113] [info]  ------- CODIFICATION TEST -------
 Filename of the packet: 020220609083055.tar.gz.partaaaa
DISPLAY ORIGINAL MESSAGE:
0 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 41 42 43 44 ...
    45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
    65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
    85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
    104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
    119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
    134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
    149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
    164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
    179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
    194 195 196 197 198 199
[2022-06-19 12:32:07.120] [info] INFO: Padding added
[2022-06-19 12:32:07.122] [info] INFO: Header added
[2022-06-19 12:32:07.127] [info] INFO: Reed Solomon codification ...
    done
[2022-06-19 12:32:07.127] [info] INFO: Interleaving done
[2022-06-19 12:32:07.128] [info] INFO: Convolutional ...
    codification done
DISPLAY ENCODED MESSAGE:
53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
    188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
```

```
      223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
      170 225 94 238 238 213 41 74 139 218 242 194 106 172 74 65 ...
      220 20 242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 ...
      232 6 34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 ...
      155 234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 ...
      156 226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 ...
      137 1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 ...
      116 109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 ...
      85 170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
      166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
      145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
      174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
      3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
      104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
      128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
      184 235 96 145 94 15 238 179 0 153 205 77 157 160 11 245 43 ...
      45 179 67 147 174 209 23 241 27 133 117 165 152 51 205 19 194 ...
      238 57 133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 ...
      100 162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 ...
      230 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
      125 0 93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 ...
      200 181 232 185 169 236 137 1 42 111 10 183 115 14 83 227 240 ...
      141 235 167 54 94 213 243 182 89 89 248 118 168 155 234 34 ...
      202 250 82 148 114 121 228 44 196 32 176 78 144 160 129 145 ...
      212 101 238 55 152 250 149 5 251 37 247 81 153 113 116 231 33 ...
      199 249 95 151 127 122 61 195 19 210 26 242 194 106 120 158 ...
      219 2 81 94 216 100 239 132 2 39 108 7 180 170 225 138 12 41 ...
      98 9 109 207 105 161 73 76 223 25 43 46 224 197 246 107 227 176
 SIMULATE TRANSMISSION AND RECEPTION WITH ERROR ADDITION
DISPLAY ENCODED MESSAGE WITH ADDED ERRORS:
33 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
      188 33 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
      223 25 255 33 103 175 71 65 86 164 239 132 2 39 108 7 180 170 ...
      225 94 238 238 213 33 74 139 218 242 194 106 172 74 65 220 20 ...
      242 5 251 37 247 81 153 113 33 231 33 199 249 95 175 232 6 34 ...
      19 143 244 81 212 177 57 18 87 50 33 24 136 118 168 155 234 ...
      34 202 250 82 148 114 121 228 44 29 109 129 250 33 37 156 226 ...
      135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 137 33 ...
      42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 116 109 ...
      69 56 33 213 198 187 230 99 159 218 191 55 28 89 60 85 170 94 ...
      224 31 211 35 33 176 0 239 199 62 206 30 22 134 120 166 149 ...
      48 192 16 204 100 162 68 33 6 246 38 249 238 227 66 145 64 ...
      221 159 201 35 189 77 157 160 11 245 33 45 179 67 147 174 209 ...
      23 241 27 133 117 165 152 228 158 2 152 62 187 33 176 3 80 ...
      238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 104 33 ...
      203 184 235 183 46 2 149 69 117 127 179 80 238 243 128 211 88 ...
      126 13 33 224 253 142 221 130 72 59 104 214 203 184 235 96 ...
      145 94 15 238 179 0 33 205 77 157 160 11 245 43 45 179 67 147 ...
      174 209 23 241 27 133 117 165 33 51 205 19 194 238 57 133 137 ...
      254 192 14 30 22 134 120 166 149 48 192 33 204 100 162 68 79 ...
      6 246 38 46 190 64 158 173 223 158 52 39 229 62 33 69 56 101 ...
      213 198 187 230 99 159 218 191 55 28 89 60 85 125 0 93 33 254 ...
      131 222 140 46 12 92 226 178 216 134 107 219 200 181 232 185 ...
      169 236 33 1 42 111 10 183 115 14 83 227 240 141 235 167 54 ...
      94 213 243 182 89 33 248 118 168 155 234 34 202 250 82 148 ...
      114 121 228 44 196 32 176 78 144 33 129 145 212 101 238 55 ...
      152 250 149 5 251 37 247 81 153 113 116 231 33 33 249 95 151 ...
      127 122 61 195 19 210 26 242 194 106 120 158 219 2 81 94 33 ...
```

```
      100 239 132 2 39 108 7 180 170 225 138 12 41 98 9 109 207 105 ...
      161 33 76 223 25 43 46 224 197 246 107 227 176
16 [2022-06-19 12:32:07.141] [info] INFO: Convolutional ...
      decodification done
17 [2022-06-19 12:32:07.141] [info] INFO: Deinterleaving done
18 [2022-06-19 12:32:07.142] [info] INFO: Reed Solomon ...
      decodification done
19 recovered filename : 020220609083055.tar.gz.partaaaa
20 DISPLAY RECOVERED MESSAGE:
21 0 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 41 42 43 44 ...
      45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
      65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
      85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
      104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
      119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
      134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
      149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
      164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
      179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
      194 195 196 197 198 199
22 Recovered filename of the packet: 020220609083055.tar.gz.partaaaa
23 [2022-06-19 12:32:07.146] [info] END OF TEST
24 }
```

As it can be seen, the channel decoding functions are able to recover the message and the filename despite the introduced errors.

G.1.2 Periodical errors every 10 bytes

This test has been carried out changing a byte every 10 bytes for a 33.

The console of the test is shown below:

```
1 {
2 [2022-06-19 13:22:16.116] [info]  ------- CODIFICATION TEST -------
3  Filename of the packet: 020220609083055.tar.gz.partaaaa
4 DISPLAY ORIGINAL MESSAGE:
5 0 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 41 42 43 44 ...
      45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
      65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
      85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
      104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
      119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
      134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
      149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
      164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
      179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
      194 195 196 197 198 199
6 [2022-06-19 13:22:16.123] [info] INFO: Padding added
7 [2022-06-19 13:22:16.123] [info] INFO: Header added
8 [2022-06-19 13:22:16.124] [info] INFO: Reed Solomon codification ...
      done
9 [2022-06-19 13:22:16.128] [info] INFO: Interleaving done
```

```
[2022-06-19 13:22:16.131] [info] INFO: Convolutional ...
    codification done
DISPLAY ENCODED MESSAGE:
53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
    188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
    223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
    170 225 94 238 238 213 41 74 139 218 242 194 106 172 74 65 ...
    220 20 242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 ...
    232 6 34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 ...
    155 234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 ...
    156 226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 ...
    137 1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 ...
    116 109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 ...
    85 170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
    166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
    145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
    174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
    3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
    104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
    128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
    184 235 96 145 94 15 238 179 0 153 205 77 157 160 11 245 43 ...
    45 179 67 147 174 209 23 241 27 133 117 165 152 51 205 19 194 ...
    238 57 133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 ...
    100 162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 ...
    230 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
    125 0 93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 ...
    200 181 232 185 169 236 137 1 42 111 10 183 115 14 83 227 240 ...
    141 235 167 54 94 213 243 182 89 89 248 118 168 155 234 34 ...
    202 250 82 148 114 121 228 44 196 32 176 78 144 160 129 145 ...
    212 101 238 55 152 250 149 5 251 37 247 81 153 113 116 231 33 ...
    199 249 95 151 127 122 61 195 19 210 26 242 194 106 120 158 ...
    219 2 81 94 216 100 239 132 2 39 108 7 180 170 225 138 12 41 ...
    98 9 109 207 105 161 73 76 223 25 43 46 224 197 246 107 227 176
 SIMULATE TRANSMISSION AND RECEPTION WITH ERROR ADDITION
DISPLAY ENCODED MESSAGE WITH ADDED ERRORS:
33 54 78 23 188 58 31 84 63 140 33 217 178 52 17 90 222 156 215 ...
    188 33 31 84 63 140 146 13 94 219 224 33 134 215 217 161 73 ...
    76 223 25 255 33 103 175 71 65 86 164 239 132 2 33 108 7 180 ...
    170 225 94 238 238 213 33 74 139 218 242 194 106 172 74 65 33 ...
    20 242 5 251 37 247 81 153 113 33 231 33 199 249 95 175 232 6 ...
    34 33 143 244 81 212 177 57 18 87 50 33 24 136 118 168 155 ...
    234 34 202 250 33 148 114 121 228 44 29 109 129 250 33 37 156 ...
    226 135 15 36 97 63 75 33 107 219 200 181 232 185 169 236 137 ...
    33 42 111 10 96 46 237 136 80 213 33 176 0 236 154 124 116 ...
    109 69 56 33 213 198 187 230 99 159 218 191 55 33 89 60 85 ...
    170 94 224 31 211 35 33 176 0 239 199 62 206 30 22 134 33 166 ...
    149 48 192 16 204 100 162 68 33 6 246 38 249 238 227 66 145 ...
    64 33 159 201 35 189 77 157 160 11 245 33 45 179 67 147 174 ...
    209 23 241 27 33 117 165 152 228 158 2 152 62 187 33 176 3 80 ...
    238 243 128 211 88 126 33 94 224 253 142 221 130 72 59 104 33 ...
    203 184 235 183 46 2 149 69 117 33 179 80 238 243 128 211 88 ...
    126 13 33 224 253 142 221 130 72 59 104 214 33 184 235 96 145 ...
    94 15 238 179 0 33 205 77 157 160 11 245 43 45 179 33 147 174 ...
    209 23 241 27 133 117 165 33 51 205 19 194 238 57 133 137 254 ...
    33 14 30 22 134 120 166 149 48 192 33 204 100 162 68 79 6 246 ...
    38 46 33 64 158 173 223 158 52 39 229 62 33 69 56 101 213 198 ...
    187 230 99 159 33 191 55 28 89 60 85 125 0 93 33 254 131 222 ...
    140 46 12 92 226 178 33 134 107 219 200 181 232 185 169 236 ...
```

```
      33 1 42 111 10 183 115 14 83 227 33 141 235 167 54 94 213 243 ...
      182 89 33 248 118 168 155 234 34 202 250 82 33 114 121 228 44 ...
      196 32 176 78 144 33 129 145 212 101 238 55 152 250 149 33 ...
      251 37 247 81 153 113 116 231 33 33 249 95 151 127 122 61 195 ...
      19 210 33 242 194 106 120 158 219 2 81 94 33 100 239 132 2 39 ...
      108 7 180 170 33 138 12 41 98 9 109 207 105 161 33 76 223 25 ...
      43 46 224 197 246 107 33 176
[2022-06-19 13:22:16.142] [info] INFO: Convolutional ...
      decodification done
[2022-06-19 13:22:16.145] [info] INFO: Deinterleaving done
[2022-06-19 13:22:16.145] [info] INFO: Reed Solomon ...
      decodification done
recovered filename : 020220609083055.tar.gz.partaaaa
DISPLAY RECOVERED MESSAGE:
0 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 41 42 43 44 ...
      45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
      65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
      85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
      104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
      119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
      134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
      149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
      164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
      179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
      194 195 196 197 198 199
Recovered filename of the packet: 020220609083055.tar.gz.partaaaa
[2022-06-19 13:22:16.151] [info] END OF TEST
}
```

As it can be seen, the channel decoding functions are able to recover the message and the filename despite the introduced errors.

G.1.3 Periodical errors every 5 bytes

This test has been carried out changing a byte every 5 bytes for a 33.

The console of the test is shown below:

```
{
[2022-06-19 13:24:13.063] [info]  ------- CODIFICATION TEST -------
 Filename of the packet: 020220609083055.tar.gz.partaaaa
DISPLAY ORIGINAL MESSAGE:
0 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 41 42 43 44 ...
      45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
      65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
      85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
      104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
      119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
      134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
      149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
      164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
      179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
      194 195 196 197 198 199
```

```
[2022-06-19 13:24:13.066] [info] INFO: Padding added
[2022-06-19 13:24:13.067] [info] INFO: Header added
[2022-06-19 13:24:13.068] [info] INFO: Reed Solomon codification ... done
[2022-06-19 13:24:13.071] [info] INFO: Interleaving done
[2022-06-19 13:24:13.074] [info] INFO: Convolutional ... codification done
DISPLAY ENCODED MESSAGE:
53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
    188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
    223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
    170 225 94 238 238 213 41 74 139 218 242 194 106 172 74 65 ...
    220 20 242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 ...
    232 6 34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 ...
    155 234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 ...
    156 226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 ...
    137 1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 ...
    116 109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 ...
    85 170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
    166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
    145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
    174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
    3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
    104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
    128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
    184 235 96 145 94 15 238 179 0 153 205 77 157 160 11 245 43 ...
    45 179 67 147 174 209 23 241 27 133 117 165 152 51 205 19 194 ...
    238 57 133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 ...
    100 162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 ...
    230 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
    125 0 93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 ...
    200 181 232 185 169 236 137 1 42 111 10 183 115 14 83 227 240 ...
    141 235 167 54 94 213 243 182 89 89 248 118 168 155 234 34 ...
    202 250 82 148 114 121 228 44 196 32 176 78 144 160 129 145 ...
    212 101 238 55 152 250 149 5 251 37 247 81 153 113 116 231 33 ...
    199 249 95 151 127 122 61 195 19 210 26 242 194 106 120 158 ...
    219 2 81 94 216 100 239 132 2 39 108 7 180 170 225 138 12 41 ...
    98 9 109 207 105 161 73 76 223 25 43 46 224 197 246 107 227 176
 SIMULATE TRANSMISSION AND RECEPTION WITH ERROR ADDITION
DISPLAY ENCODED MESSAGE WITH ADDED ERRORS:
33 54 78 23 188 33 31 84 63 140 33 217 178 52 17 33 222 156 215 ...
    188 33 31 84 63 140 33 13 94 219 224 33 134 215 217 161 33 76 ...
    223 25 255 33 103 175 71 65 33 164 239 132 2 33 108 7 180 170 ...
    33 94 238 238 213 33 74 139 218 242 33 106 172 74 65 33 20 ...
    242 5 251 33 247 81 153 113 33 231 33 199 249 33 175 232 6 34 ...
    33 143 244 81 212 33 57 18 87 50 33 24 136 118 168 33 234 34 ...
    202 250 33 148 114 121 228 33 29 109 129 250 33 37 156 226 ...
    135 33 36 97 63 75 33 107 219 200 181 33 185 169 236 137 33 ...
    42 111 10 96 33 237 136 80 213 33 176 0 236 154 33 116 109 69 ...
    56 33 213 198 187 230 33 159 218 191 55 33 89 60 85 170 33 ...
    224 31 211 35 33 176 0 239 199 33 206 30 22 134 33 166 149 48 ...
    192 33 204 100 162 68 33 6 246 38 249 33 227 66 145 64 33 159 ...
    201 35 189 33 157 160 11 245 33 45 179 67 147 33 209 23 241 ...
    27 33 117 165 152 228 33 2 152 62 187 33 176 3 80 238 33 128 ...
    211 88 126 33 94 224 253 142 33 130 72 59 104 33 203 184 235 ...
    183 33 2 149 69 117 33 179 80 238 243 33 211 88 126 13 33 224 ...
    253 142 221 33 72 59 104 214 33 184 235 96 145 33 15 238 179 ...
    0 33 205 77 157 160 33 245 43 45 179 33 147 174 209 23 33 27 ...
```

```
    133 117 165 33 51 205 19 194 33 57 133 137 254 33 14 30 22 ...
    134 33 166 149 48 192 33 204 100 162 68 33 6 246 38 46 33 64 ...
    158 173 223 33 52 39 229 62 33 69 56 101 213 33 187 230 99 ...
    159 33 191 55 28 89 33 85 125 0 93 33 254 131 222 140 33 12 ...
    92 226 178 33 134 107 219 200 33 232 185 169 236 33 1 42 111 ...
    10 33 115 14 83 227 33 141 235 167 54 33 213 243 182 89 33 ...
    248 118 168 155 33 34 202 250 82 33 114 121 228 44 33 32 176 ...
    78 144 33 129 145 212 101 33 55 152 250 149 33 251 37 247 81 ...
    33 113 116 231 33 33 249 95 151 127 33 61 195 19 210 33 242 ...
    194 106 120 33 219 2 81 94 33 100 239 132 2 33 108 7 180 170 ...
    33 138 12 41 98 33 109 207 105 161 33 76 223 25 43 33 224 197 ...
    246 107 33 176
[2022-06-19 13:24:13.082] [info] INFO: Convolutional ...
    decodification done
[2022-06-19 13:24:13.082] [info] INFO: Denterleaving done
ERROR
[2022-06-19 13:24:13.083] [info] INFO: Reed Solomon ...
    decodification done
recovered filename : 00000000000000.tar.gz.partaaaa
DISPLAY RECOVERED MESSAGE:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Recovered filename of the packet: 00000000000000.tar.gz.partaaaa
[2022-06-19 13:24:13.089] [info] END OF TEST
}
```

As it can be seen, the channel decoding functions are not able to recover the message. When there are two many errors to be corrected, the system returns all 0, which enables it to identify when the packet is not recoverable and discard it.

G.1.4 Deep fading errors along the transmission

This test has been carried out by adding 3 deep fading errors to the encoded message.

The console of the test is shown below:

```
{
[2022-06-19 13:29:26.600] [info] ------- CODIFICATION TEST -------
 Filename of the packet: 020220609083055.tar.gz.partaaaa
DISPLAY ORIGINAL MESSAGE:
0 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 41 42 43 44 ...
    45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
    65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
    85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
    104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
    119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
    134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
    149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
    164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
```

```
      179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
      194 195 196 197 198 199
[2022-06-19 13:29:26.606] [info] INFO: Padding added
[2022-06-19 13:29:26.607] [info] INFO: Header added
[2022-06-19 13:29:26.608] [info] INFO: Reed Solomon codification ...
      done
[2022-06-19 13:29:26.609] [info] INFO: Interleaving done
[2022-06-19 13:29:26.610] [info] INFO: Convolutional ...
      codification done
DISPLAY ENCODED MESSAGE:
53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
      188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
      223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
      170 225 94 238 238 213 41 74 139 218 242 194 106 172 74 65 ...
      220 20 242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 ...
      232 6 34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 ...
      155 234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 ...
      156 226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 ...
      137 1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 ...
      116 109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 ...
      85 170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
      166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
      145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
      174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
      3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
      104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
      128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
      184 235 96 145 94 15 238 179 0 153 205 77 157 160 11 245 43 ...
      45 179 67 147 174 209 23 241 27 133 117 165 152 51 205 19 194 ...
      238 57 133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 ...
      100 162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 ...
      230 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
      125 0 93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 ...
      200 181 232 185 169 236 137 1 42 111 10 183 115 14 83 227 240 ...
      141 235 167 54 94 213 243 182 89 89 248 118 168 155 234 34 ...
      202 250 82 148 114 121 228 44 196 32 176 78 144 160 129 145 ...
      212 101 238 55 152 250 149 5 251 37 247 81 153 113 116 231 33 ...
      199 249 95 151 127 122 61 195 19 210 26 242 194 106 120 158 ...
      219 2 81 94 216 100 239 132 2 39 108 7 180 170 225 138 12 41 ...
      98 9 109 207 105 161 73 76 223 25 43 46 224 197 246 107 227 176
 SIMULATE TRANSMISSION AND RECEPTION WITH ERROR ADDITION
DISPLAY ENCODED MESSAGE WITH ADDED ERRORS:
53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
      188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
      223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
      170 33 33 33 33 33 33 33 33 33 33 194 106 172 74 65 220 20 ...
      242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 232 6 ...
      34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 155 ...
      234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 156 ...
      226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 137 ...
      1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 116 ...
      109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
      170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
      166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
      145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
      174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
      3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
      104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
```

```
      128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
      184 235 96 145 94 15 238 179 0 153 205 77 157 160 33 33 33 33 ...
      33 33 33 33 33 33 241 27 133 117 165 152 51 205 19 194 238 57 ...
      133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 100 ...
      162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 230 ...
      69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 125 0 ...
      93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 200 ...
      181 232 185 169 236 33 33 33 33 33 33 33 33 33 33 33 33 235 ...
      167 54 94 213 243 182 89 89 248 118 168 155 234 34 202 250 82 ...
      148 114 121 228 44 196 32 176 78 144 160 129 145 212 101 238 ...
      55 152 250 149 5 251 37 247 81 153 113 116 231 33 199 249 95 ...
      151 127 122 61 195 19 210 26 242 194 106 120 158 219 2 81 94 ...
      216 100 239 132 2 39 108 7 180 170 225 138 12 41 98 9 109 207 ...
      105 161 73 76 223 25 43 46 224 197 246 107 227 176
[2022-06-19 13:29:26.626] [info] INFO: Convolutional ...
    decodification done
[2022-06-19 13:29:26.627] [info] INFO: Deinterleaving done
[2022-06-19 13:29:26.627] [info] INFO: Reed Solomon ...
    decodification done
recovered filename : 020220609083055.tar.gz.partaaaa
DISPLAY RECOVERED MESSAGE:
0 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 41 42 43 44 ...
    45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
    65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
    85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
    104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
    119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
    134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
    149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
    164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
    179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
    194 195 196 197 198 199
Recovered filename of the packet: 020220609083055.tar.gz.partaaaa
[2022-06-19 13:29:26.630] [info] END OF TEST
}
```

As it can be seen, the channel decoding functions are able to recover the message. This is thanks to the interleaver, that scatters the errors along the whole message allowing the decoding scheme to correct them.

APPENDIX H. PHYSICAL CHANNEL TRANSMISSION AND RECEPTION TESTS

This appendix provides the physical channel transmission and reception tests results.

H.1 Transmission and reception of a simple array of bits

This test consists in the transmission of a simple array of bits from an ADALM-PLUTO to another when they are connected through a coaxial cable.

The following code depicts the transmitter console results. As it can be seen it displays the message to be sent, and transmits it.

```
{
[1970-01-01 03:19:16.892] [info] ------------- TRANSMISSION  ...
    -------------
[1970-01-01 03:19:16.892] [info] This test will transmit a given ...
    message through the ADALM-PLUTO.
Message to be transmitted:
0 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 41 42 43 44 ...
    45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
    65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
    85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
    104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
    119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
    134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
    149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
    164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
    179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
    194 195 196 197 198 199
[1970-01-01 03:19:16.893] [info] Transmission code activated
[1970-01-01 03:19:16.893] [info] INFO: Initialising iio-handler
[1970-01-01 03:19:16.893] [info] Buffer size TX in config : 8192
[1970-01-01 03:19:25.414] [info] READY for transmission!
[IIO handler]: being destroyed
[IIO handler]: destroying channels
[IIO handler]: Context destroyed
[1970-01-01 03:19:35.413] [info] Transmission successful
[1970-01-01 03:19:35.413] [info] INFO: END TEST
}
```

The following code depicts the receiver console results. As it can be seen it shows how the synchronization has detected a correlation of 0.967 and activated the Costas Loop. Then it displays the recovered message, that as it can be seen, is coincident with the one that has been sent.

```
{
[1970-01-01 03:19:25.545] [info] ------------- RECEPTION  ...
    -------------
[1970-01-01 03:19:25.545] [info] This test will receive a ...
    message and display it
```

```
[1970-01-01 03:19:25.546] [info] INFO: Initialising iio-handler
[1970-01-01 03:19:25.546] [info] Buffer size RX in config : 30720
[1970-01-01 03:19:28.248] [info] READY for reception!
[1970-01-01 03:19:28.249] [info] start filling buffer
[1970-01-01 03:19:28.249] [info] Fill buffer
[1970-01-01 03:19:28.257] [info] Synch
[1970-01-01 03:19:28.257] [info] Start computation
[1970-01-01 03:19:29.248] [info] Delete vectors of computation
[1970-01-01 03:19:29.249] [info] max_correl_buffIndex found : ...
     19005 with with max correl :0.96743685
[1970-01-01 03:19:29.249] [info] Reception done --> synch word ...
    detected
[1970-01-01 03:19:29.250] [info] Start COSTAS LOOP
[IIO handler]: being destroyed
[IIO handler]: destroying channels
[IIO handler]: Context destroyed
DISPLAY RECOVERED MESSAGE:
0 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 41 42 43 44 ...
    45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
    65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
    85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
    104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
    119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
    134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
    149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
    164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
    179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
    194 195 196 197 198 199
[1970-01-01 03:19:29.525] [info] INFO: DONE :)
}
```

These test demonstrates that the synchronization, Costas loop and demodulation are working correctly, verifying then the receiver signal processing software.

H.2 Transmission and reception of a message

These tests consists on the transmission of an encoded array of bits from an ADALM-PLUTO with several configurations.

H.2.1 Test with coaxial cable between the transmitter and receiver

This test consists in the transmission of an encoded simple array of bits from an ADALM-PLUTO to another when they are connected through a coaxial cable.

The following code depicts the transmitter console results. As it can be seen it displays the original message to be sent and the given filename. After using the channel encoding functions, it displays the encoded message and transmits it.

```
{
[1970-01-01 01:50:00.552] [info] ------------- TRANSMISSION  ...
    -------------
```

```
[1970-01-01 01:50:00.553] [info] This test will encode a given ...
    message and transmit it through the ADALM-PLUTO.
Message to be transmitted:
0 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 41 42 43 44 ...
    45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
    65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
    85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
    104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
    119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
    134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
    149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
    164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
    179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
    194 195 196 197 198 199
Filename of the message to be transmitted: ...
    020220609083055.tar.gz.partaaaa
[1970-01-01 01:50:00.553] [info] INFO: Padding added
[1970-01-01 01:50:00.554] [info] INFO: Header added
[1970-01-01 01:50:00.554] [info] INFO: Reed Solomon codification ...
    done
[1970-01-01 01:50:00.554] [info] INFO: Interleaving done
[1970-01-01 01:50:00.554] [info] INFO: Convolutional ...
    codification done
Encoded message:
53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
    188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
    223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
    170 225 94 238 238 213 41 74 139 218 242 194 106 172 74 65 ...
    220 20 242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 ...
    232 6 34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 ...
    155 234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 ...
    156 226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 ...
    137 1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 ...
    116 109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 ...
    85 170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
    166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
    145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
    174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
    3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
    104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
    128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
    184 235 96 145 94 15 238 179 0 153 205 77 157 160 11 245 43 ...
    45 179 67 147 174 209 23 241 27 133 117 165 152 51 205 19 194 ...
    238 57 133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 ...
    100 162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 ...
    230 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
    125 0 93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 ...
    200 181 232 185 169 236 137 1 42 111 10 183 115 14 83 227 240 ...
    141 235 167 54 94 213 243 182 89 89 248 118 168 155 234 34 ...
    202 250 82 148 114 121 228 44 196 32 176 78 144 160 129 145 ...
    212 101 238 55 152 250 149 5 251 37 247 81 153 113 116 231 33 ...
    199 249 95 151 127 122 61 195 19 210 26 242 194 106 120 158 ...
    219 2 81 94 216 100 239 132 2 39 108 7 180 170 225 138 12 41 ...
    98 9 109 207 105 161 73 76 223 25 43 46 224 197 246 107 227 176
[1970-01-01 01:50:00.555] [info] Transmission code activated
[1970-01-01 01:50:00.556] [info] INFO: Initialising iio-handler
[1970-01-01 01:50:00.556] [info] Buffer size TX in config : 22528
```

```
[1970-01-01 01:50:02.944] [info] READY for transmission!
[IIO handler]: being destroyed
[IIO handler]: destroying channels
[IIO handler]: Context destroyed
[1970-01-01 01:50:30.439] [info] Transmission successful
[1970-01-01 01:50:30.439] [info] INFO: END TEST
}
```

The following code depicts the receiver console results. As it can be seen it shows how the synchronization has detected a correlation of 0.9497 and activated the Costas Loop. Then it displays the recovered message, that as expected, is encoded. After its decoding, the original message is recovered and displayed. As it can be seen, it corresponds to the transmitted message. Furthermore, it can also be seen that the header decoding has also worked properly since the filename has also been recovered correctly.

```
{
[1970-01-01 01:50:09.886] [info] ------------- RECEPTION  ...
    -------------
[1970-01-01 01:50:09.887] [info] This test will receive an ...
    encoded message, decode it and display it
[1970-01-01 01:50:09.887] [info] INFO: Initialising iio-handler
[1970-01-01 01:50:09.887] [info] Buffer size RX in config : 30720
[1970-01-01 01:50:12.956] [info] READY for reception!
[1970-01-01 01:50:12.957] [info] start filling buffer
[1970-01-01 01:50:12.957] [info] Fill buffer
[1970-01-01 01:50:12.961] [info] Synch
[1970-01-01 01:50:12.961] [info] Start computation
[1970-01-01 01:50:13.959] [info] Delete vectors of computation
[1970-01-01 01:50:13.959] [info] max_correl_buffIndex found : ...
     9972 with with max correl :0.94966644
[1970-01-01 01:50:13.960] [info] Reception done --> synch word ...
    detected
[1970-01-01 01:50:13.961] [info] Start COSTAS LOOP
[IIO handler]: being destroyed
[IIO handler]: destroying channels
[IIO handler]: Context destroyed
Recovered encoded message:
53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
    188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
    223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
    170 225 94 238 238 213 41 74 139 218 242 194 106 172 74 65 ...
    220 20 242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 ...
    232 6 34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 ...
    155 234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 ...
    156 226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 ...
    137 1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 ...
    116 109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 ...
    85 170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
    166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
    145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
    174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
    3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
    104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
    128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
    184 235 96 145 94 15 238 179 0 153 205 77 157 160 11 245 43 ...
    45 179 67 147 174 209 23 241 27 133 117 165 152 51 205 19 194 ...
```

```
      238 57 133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 ...
      100 162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 ...
      230 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
      125 0 93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 ...
      200 181 232 185 169 236 137 1 42 111 10 183 115 14 83 227 240 ...
      141 235 167 54 94 213 243 182 89 89 248 118 168 155 234 34 ...
      202 250 82 148 114 121 228 44 196 32 176 78 144 160 129 145 ...
      212 101 238 55 152 250 149 5 251 37 247 81 153 113 116 231 33 ...
      199 249 95 151 127 122 61 195 19 210 26 242 194 106 120 158 ...
      219 2 81 94 216 100 239 132 2 39 108 7 180 170 225 138 12 41 ...
      98 9 109 207 105 161 73 76 223 25 43 46 224 197 246 107 227 176
[1970-01-01 01:50:14.429] [info] Decodification activated
[1970-01-01 01:50:14.433] [info] INFO: Convolutional ...
    decodification done
[1970-01-01 01:50:14.434] [info] INFO: Deinterleaving done
[1970-01-01 01:50:14.434] [info] INFO: Reed Solomon ...
    decodification done
recovered filename : 020220609083055.tar.gz.partaaaa
Recovered decoded message:
0 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 41 42 43 44 ...
    45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
    65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
    85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
    104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
    119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
    134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
    149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
    164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
    179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
    194 195 196 197 198 199
Recovered filename of the packet: 020220609083055.tar.gz.partaaaa
[1970-01-01 01:50:14.435] [info] INFO: DONE :)
}
```

H.2.2 Test with coaxial cable with extra attenuation between the transmitter and receiver

This test consists in the transmission of an encoded simple array of bits from an ADALM-PLUTO to another when they are communicating through a coaxial cable with an added attenuator.

The following code depicts the transmitter console results. As it can be seen it displays the original message to be sent and the given filename. After using the channel encoding functions, it displays the encoded message and transmits it.

```
{
[1970-01-01 02:58:43.531] [info] ------------- TRANSMISSION  ...
    -------------
[1970-01-01 02:58:43.532] [info] This test will encode a given ...
    message and transmit it through the ADALM-PLUTO.
Message to be transmitted:
0 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 41 42 43 44 ...
```

```
      45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
      65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
      85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
      104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
      119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
      134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
      149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
      164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
      179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
      194 195 196 197 198 199
6 Filename of the message to be transmitted: ...
      020220609083055.tar.gz.partaaaa
7 [1970-01-01 02:58:43.533] [info] INFO: Padding added
8 [1970-01-01 02:58:43.533] [info] INFO: Header added
9 [1970-01-01 02:58:43.533] [info] INFO: Reed Solomon codification ...
      done
10 [1970-01-01 02:58:43.534] [info] INFO: Interleaving done
11 [1970-01-01 02:58:43.534] [info] INFO: Convolutional ...
      codification done
12 Encoded message:
13 53 54 78 23 188 58 31 84 63 140 146 217 178 52 17 90 222 156 215 ...
      188 58 31 84 63 140 146 13 94 219 224 2 134 215 217 161 73 76 ...
      223 25 255 193 103 175 71 65 86 164 239 132 2 39 108 7 180 ...
      170 225 94 238 238 213 41 74 139 218 242 194 106 172 74 65 ...
      220 20 242 5 251 37 247 81 153 113 116 231 33 199 249 95 175 ...
      232 6 34 19 143 244 81 212 177 57 18 87 50 140 24 136 118 168 ...
      155 234 34 202 250 82 148 114 121 228 44 29 109 129 250 81 37 ...
      156 226 135 15 36 97 63 75 54 107 219 200 181 232 185 169 236 ...
      137 1 42 111 10 96 46 237 136 80 213 216 176 0 236 154 124 ...
      116 109 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 ...
      85 170 94 224 31 211 35 15 176 0 239 199 62 206 30 22 134 120 ...
      166 149 48 192 16 204 100 162 68 79 6 246 38 249 238 227 66 ...
      145 64 221 159 201 35 189 77 157 160 11 245 43 45 179 67 147 ...
      174 209 23 241 27 133 117 165 152 228 158 2 152 62 187 68 176 ...
      3 80 238 243 128 211 88 126 13 94 224 253 142 221 130 72 59 ...
      104 214 203 184 235 183 46 2 149 69 117 127 179 80 238 243 ...
      128 211 88 126 13 94 224 253 142 221 130 72 59 104 214 203 ...
      184 235 96 145 94 15 238 179 0 153 205 77 157 160 11 245 43 ...
      45 179 67 147 174 209 23 241 27 133 117 165 152 51 205 19 194 ...
      238 57 133 137 254 192 14 30 22 134 120 166 149 48 192 16 204 ...
      100 162 68 79 6 246 38 46 190 64 158 173 223 158 52 39 229 62 ...
      230 69 56 101 213 198 187 230 99 159 218 191 55 28 89 60 85 ...
      125 0 93 237 254 131 222 140 46 12 92 226 178 216 134 107 219 ...
      200 181 232 185 169 236 137 1 42 111 10 183 115 14 83 227 240 ...
      141 235 167 54 94 213 243 182 89 89 248 118 168 155 234 34 ...
      202 250 82 148 114 121 228 44 196 32 176 78 144 160 129 145 ...
      212 101 238 55 152 250 149 5 251 37 247 81 153 113 116 231 33 ...
      199 249 95 151 127 122 61 195 19 210 26 242 194 106 120 158 ...
      219 2 81 94 216 100 239 132 2 39 108 7 180 170 225 138 12 41 ...
      98 9 109 207 105 161 73 76 223 25 43 46 224 197 246 107 227 176
14 [1970-01-01 02:58:43.535] [info] Transmission code activated
15 [1970-01-01 02:58:43.536] [info] INFO: Initialising iio-handler
16 [1970-01-01 02:58:43.536] [info] Buffer size TX in config : 22528
17 [1970-01-01 02:58:45.959] [info] READY for transmission!
18 [IIO handler]: being destroyed
19 [IIO handler]: destroying channels
20 [IIO handler]: Context destroyed
21 [1970-01-01 02:59:13.454] [info] Transmission successful
```

```
22 [1970-01-01 02:59:13.454] [info] INFO: END TEST
23 }
```

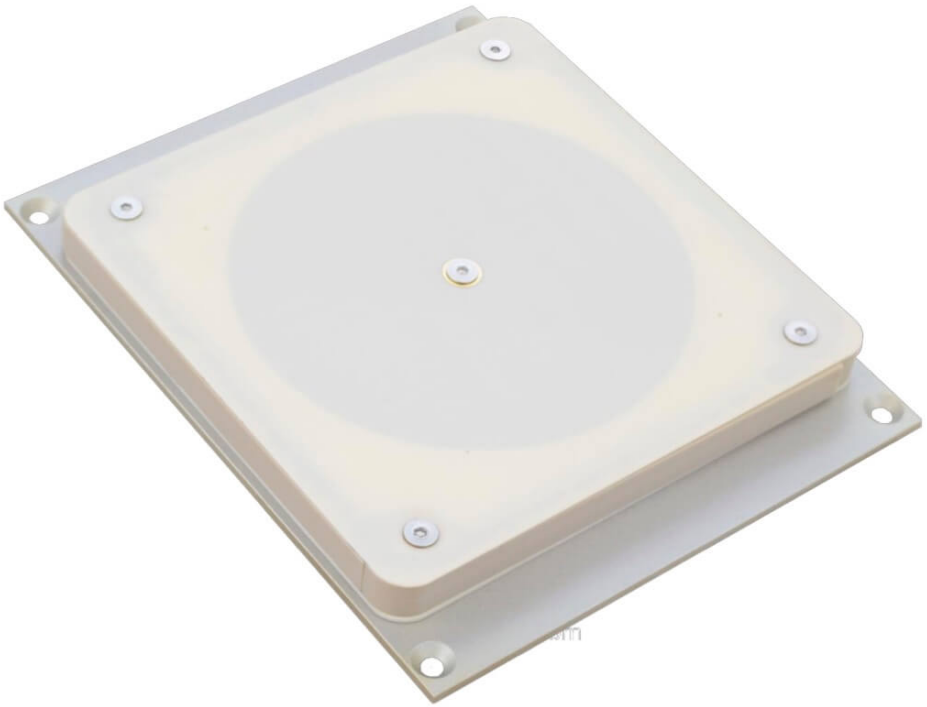
The following code depicts the receiver console results. As it can be seen it shows how the synchronization has detected a correlation of 0.9497 and activated the Costas Loop. Then it displays the recovered message, that as expected, is encoded. After its decoding, the original message is recovered and displayed. As it can be seen, it corresponds to the transmitted message. Furthermore, it can also be seen that the header decoding has also worked properly since the filename has also been recovered correctly.

```
1  {
2  [1970-01-01 02:58:46.056] [info] ------------- RECEPTION  ...
       -------------
3  [1970-01-01 02:58:46.056] [info] This test will receive an ...
       encoded message, decode it and display it
4  [1970-01-01 02:58:46.056] [info] INFO: Initialising iio-handler
5  [1970-01-01 02:58:46.057] [info] Buffer size RX in config : 30720
6  [1970-01-01 02:58:48.939] [info] READY for reception!
7  [1970-01-01 02:58:48.940] [info] start filling buffer
8  [1970-01-01 02:58:48.940] [info] Fill buffer
9  [1970-01-01 02:58:48.944] [info] Synch
10 [1970-01-01 02:58:48.944] [info] Start computation
11 [1970-01-01 02:58:49.947] [info] Delete vectors of computation
12 [1970-01-01 02:58:49.947] [info] max_correl_buffIndex found : ...
        1065 with with max correl :0.9677358
13 [1970-01-01 02:58:49.947] [info] Reception done --> synch word ...
       detected
14 [1970-01-01 02:58:49.949] [info] Start COSTAS LOOP
15 [IIO handler]: being destroyed
16 [IIO handler]: destroying channels
17 [IIO handler]: Context destroyed
18 [1970-01-01 02:58:50.621] [info] INFO: Convolutional ...
       decodification done
19 [1970-01-01 02:58:50.621] [info] INFO: Deinterleaving done
20 [1970-01-01 02:58:50.622] [info] INFO: Reed Solomon ...
       decodification done
21 recovered filename : 020220609083055.tar.gz.partaaaa
22 [1970-01-01 02:58:50.622] [info] Message recovered successfully
23 Recovered encoded message:
24 32 115 6 0 16 32 64 128 95 190 35 70 140 71 142 67 134 83 166 19 ...
       38 76 152 111 222 227 153 109 218 235 137 77 154 107 214 243 ...
       185 45 90 180 55 110 220 231 145 125 250 171 9 18 36 72 144 ...
       127 254 163 25 50 100 200 207 193 221 229 149 117 234 139 73 ...
       146 123 246 179 57 114 228 151 113 226 155 105 210 251 169 13 ...
       26 52 104 208 255 161 29 58 116 232 143 65 130 91 182 51 102 ...
       204 199 209 253 165 21 42 84 168 15 30 60 120 240 191 33 66 ...
       132 87 174 3 6 12 24 48 96 192 223 225 157 101 202 203 201 ...
       205 197 213 245 181 53 106 212 247 177 61 122 244 183 49 98 ...
       196 215 241 189 37 74 148 119 238 131 89 178 59 118 236 135 ...
       81 162 27 54 108 216 239 129 93 186 43 86 172 7 14 28 56 112 ...
       224 159 97 194 219 233 141 69 138 75 150 115 230 147 121 242 ...
       187 41 82 164 23 46 92 184 47 94 188 39 78 156 103 206 195 ...
       217 237 133 85 170 11 22 44 88 176 63 126 252 167 17 34 68 ...
       136 79 158 99 198 211 249 173 5 10 20 40 80 160 31 62 124 248 ...
       175 1 2 4 8 16 32 64 128 95 190 35 70 140 71 142 67 134 83 ...
       166 19 38 76 152 111 222 227 153 109 218 235 137 77 154 107 ...
```

```
      214 243 185 45 90 180 55 110 220 231 145 125 250 171 9 18 36 ...
      72 144 127 254 163 25 50 100 200 207 193 221 229 149 117 234 ...
      139 73 146 123 246 179 57 114 228 151 113 226 155 105 210 251 ...
      169 13 26 52 104 208 255 161 29 58 116 232 143 65 130 91 182 ...
      51 102 204 199 209 253 165 21 42 84 168 15 30 60 120 240 191 ...
      33 66 132 87 174 3 6 12 24 48 96 192 223 225 157 101 202 203 ...
      201 205 197 213 245 181 53 106 212 247 177 61 122 244 183 49 ...
      98 196 215 241 189 37 74 148 119 238 131 89 178 59 118 236 ...
      135 81 162 27 54 108 216 239 129 93 186 43 86 172 7 14 28 56 ...
      112 224 159 97 194 219 233 141 69 138 75 150 115 230 147 121 ...
      242 187 41 82 164 23 46 92 184 47 94 188 39 78 156 103 206 ...
      195 217 237 133 85 170 11 22 44 88 176 63 126 252 167 17 34 ...
      68 136 79 158 99 198 211 249 173 5 10 20 40 80 160 31 62 124 ...
      248 175 1 2
Recovered decoded message:
0 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 41 42 43 44 ...
      45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 ...
      65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 ...
      85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 ...
      104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 ...
      119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 ...
      134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 ...
      149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 ...
      164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 ...
      179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 ...
      194 195 196 197 198 199
Recovered filename of the packet: 020220609083055.tar.gz.partaaaa
[1970-01-01 02:58:50.623] [info] INFO: DONE :)
}
```

APPENDIX I. NANOCOM ANT2090 DUP, ANT2150 DUP AND ANT2150 ISL DATASHEET

GOMSPACE



NanoCom
ANT2090 DUP
ANT2150 DUP
ANT2150 ISL

Datasheet
S-band active antenna

GomSpace A/S | Langagervej 6 | DK-9220 Aalborg East | Denmark
gomspace.com | sales@gomspace.com | T: +45 71 741 741

Product name: NanoCom ANT2000

Document No.: 1010651

Revision: 2.1

Author: ROBB

Approved by: HKK

Approval date: 03-08-2021

Confidentiality Notice

1 Table of Contents

2 Overview

GomSpace NanoCom ANT2090 DUP, ANT2150 DUP and ANT2150 ISL are active antennas specifically designed for interfacing with GomSpace SDR transceivers. All three antennas are referred to under a common name, ANT2000.

Several mounting versions are available depending on placement on a nano-satellite.

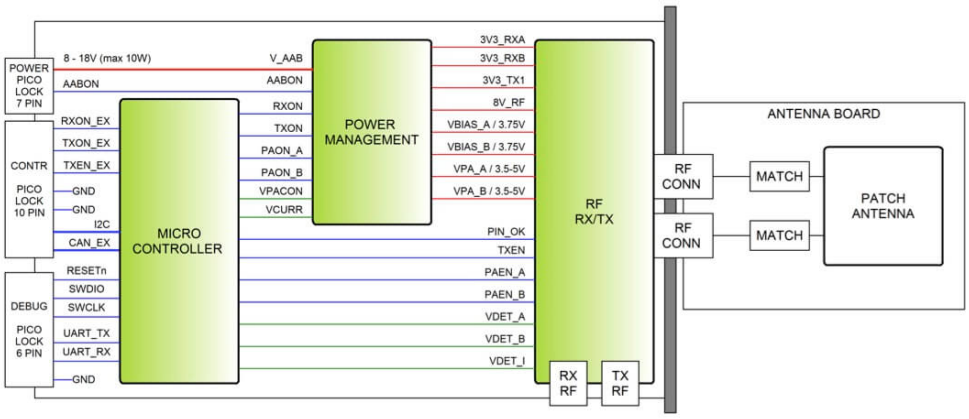
The active antenna is built as a sandwich around a shield/mounting plate. The RF signal connection between the antenna and electronics PCBs are made with RF compression connectors. This construction allows flexible mounting on several different satellite structures – just by changing the shield/mounting plate.

Below is shown two photos of the top and the bottom of the antenna, all three versions are similar in appearance.

2.1 Highlighted Features

- Integrated antenna and PA/LNA results in low loss and optimum RF performance.
- Duplex filter based design results in optimum co-existence with other RF transceivers on-board.
- Flexible sandwich construction allows flexible mounting on different satellite structures – just by changing the shield/mounting plate.
- Shielded electronics.
- Flexible power interface (VIN 8-18 V).
- Default CAN-bus control interface.
- Medium gain (8 dBi) patch antenna with circular polarization.
- ANT2090-DUP version supports full duplex with RX in 1980-2010 MHz and TX in 2170-2200 MHz.
- ANT2150 DUP version supports full duplex with RX in 2025-2110 MHz and TX in 2200-2290 MHz.
- ANT2150 ISL version supports time division duplex (TDD) in 2200–2290 MHz frequency band.
- Temperature sensors (one sensor for PA and one in the microcontroller).
- Input current sensor (for monitoring VIN current above 300mA).
- PCB material:
  - Electronics Board: Glass/Polyimide IPC 6012C cl. 3
  - Antenna Module: Rogers RO4003C
- IPC-A-610 Class 3 assembly

2.2 Block Diagram



2.3 Functional Description

ANT2000 contain: a transmit power amplifier, a receive low noise amplifier, transmit/receive switch (ISL version), and necessary support circuits.

The antenna section includes matching components and the RF compression connectors. Stated antenna gain includes loss in matching circuit and in the RF connectors.

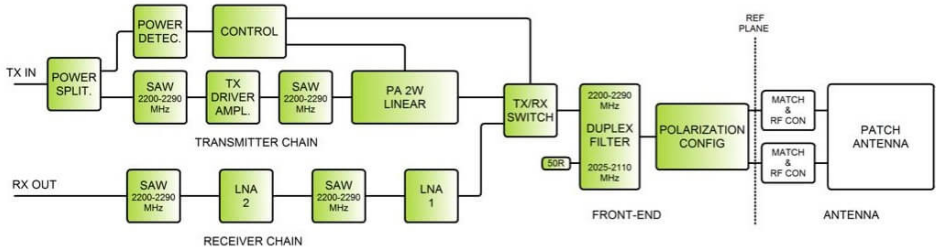
The transmitter chain includes a (pre) driver amplifier stage and a multistage balanced PA. Interstage SAW filters reduce broadband noise and provides stage isolation.

The receiver chain contains two LNAs again with interstage SAW filters to protect following stages from out-of-band interference.

All performance parameters for the electronics board and for the antenna module are given at the reference plane.

2.3.1 ISL Version

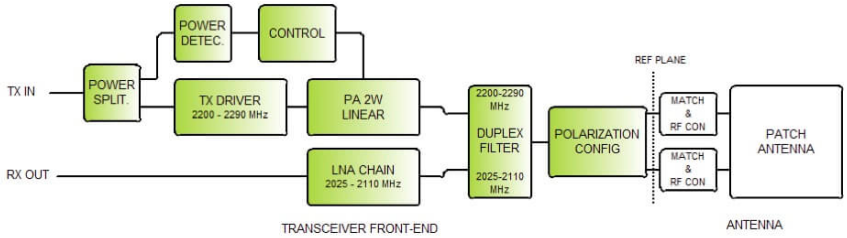
This front-end uses the same frequency band for RX and TX, which allows two ISL radio board to communicate using time division duplex.



The ISL front-end has a TX/RX switch and uses a bandpass filter (half of duplex-filter) that reduces transmitter noise (broadband) significantly. During production/checkout it can be configured if the antenna should use RHCP or LHCP.

2.3.2 DUP Version

The DUP version is the same as the ISL version except for the TX/RX switch and the filters on the Receiver chain. This version is available for two different frequency bands (see section 3.2).



2.4 Calibration and Setup

During production check-out, several calibration values are stored in the board.

For the receiver calibration values describe the RX gain at 7 frequencies across the RX band. These values can be used to implement a compensated RSSI function. (Antenna gain is not included). The RSSI values can also be temperature compensated using stored calibration values for gain temperature dependency.

The transmitter calibration values are primarily used to setup the correct input power to reach a given output power (compensated for frequency and temperature). One calibration value describes the input power detector level for a -10 dBm center frequency carrier, which will allow the SDR platform to set this level independently of cable loss. All other input levels should be set relative to the calibrated -10 dB level.

The PA has three predefined bias levels, which are optimized for 2 W, 1 W, and 0.5 W (se chapter 10.3). All levels are setup for maximum -20 dBc adjacent channel power for a 500 ksymbol/s QPSK (RC 0.35) modulated signal.

3 Versions

When ordering the customer must make a choice of:

- PCB version – ANT2150 ISL, ANT2150-DUP or ANT2090-DUP
- Antenna polarization (ISL default LHCP and DUP default RHCP)
- Mounting plate - depends on where the antenna is to be mounted
- Profile height – a special low profile of the antenna is available for the ISL version
- CAN termination – whether the board shall be equipped with a 120ohm terminated resistor on its CAN interface

3.1 Inter Satellite Link (ISL) and Profile

	ANT2150 ISL	Unit
TX band	2200 - 2290	MHz
RX band	2200 - 2290	MHz

The ISL antenna module version comes with either a standard profile or a low profile. The low profile is intended for type D mounting (see chapter 3.3).

3.2 Duplex (DUP) and Profile

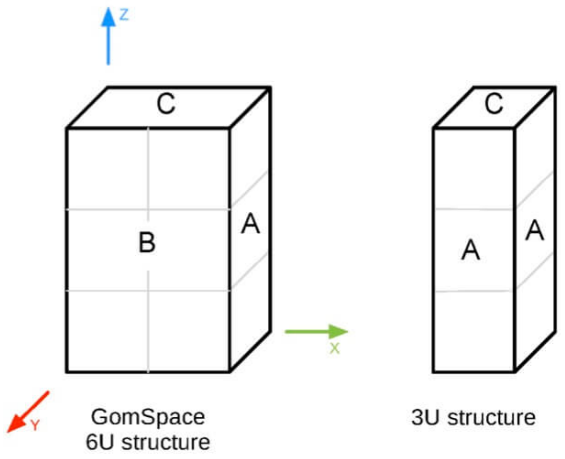
DUP comes in two versions.

	ANT2150 DUP	ANT2090 DUP	Unit
TX band	2200 - 2290	2170 - 2200	MHz
RX band	2025 - 2120	1980 - 2010	MHz

The DUP is only available with the standard profile antenna.

3.3 Mounting Plate

Four different mounting plates are available, depending on where the antenna is placed on a nano-satellite. They are all 1.5 mm aluminum. The 3U is used as an example; the plates can also be mounted on 1U and 2U nano-satellite.



Type A

Used in a 3U structure on the A-sides and on the 6U structure A-sides.

Type B

Please consult GomSpace before choosing mounting plate Type B.Special conditions apply to mounting Plate B.

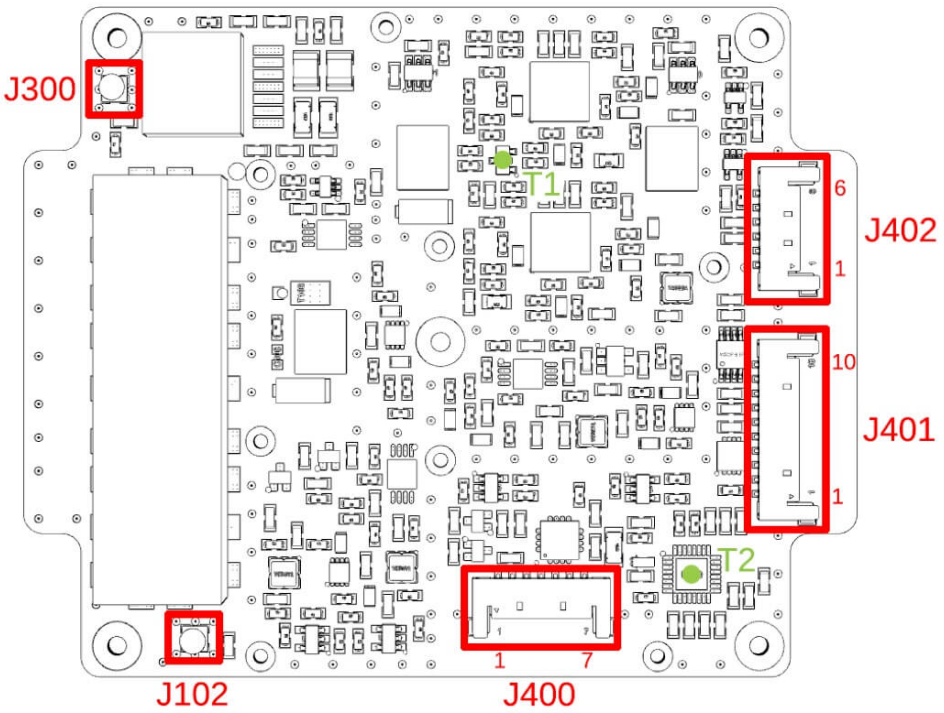
Type C

Used in the top or bottom of a 3U structure and not exceed the height of the structure rails.

Type D

Designed for use with the GomSpace NanoCom ANT-6F. When equipped with a ISL with a low profile antenna, the design is made so that the overall height doesn't exceed the height of the structure rails, when ANT-6F mounted on the top or bottom of a 6U structure. If used with a standard profile antenna, precautions should be taken to ensure the launch-pod can accommodate the extra height.

4 Connector Pinout



4.1 Connector Location Top

● Temperature sensors T1 - PA
T2 - Internal IC

4.1.1 J102 - RX RF COAXIAL CONNECTOR

Molex SSMCX 73413-0040

Pin	Name	Description
1	RX RF	Amplified received signal
2	GND	

4.1.2 J300 - TX RF COAXIAL CONNECTOR

Molex SSMCX 73413-0040

Pin	Name	Description
1	TX RF	Transmitter input signal
2	GND	

4.1.3 J400 - Power Connector

Molex Pico-Lock 1.50 mm pitch 504050-0791

Pin	Name	Description
1	AABON	External power control pin: Low < 0.4 V, High > 2.5 V (max 18V) Low: Antenna is OFF High: Antenna is ON (Antenna enters IDLE mode when AABON pin is asserted)
2	GND	
3	GND	
4	GND	
5	VIN	8 – 18 V supply voltage, externally switchable by AABON
6	VIN	8 – 18 V supply voltage, externally switchable by AABON
7	VIN	8 – 18 V supply voltage, externally switchable by AABON

4.1.4 J401 - Control Connector

Molex Pico-Lock 1.50 mm pitch 504050-1091

Pin	Name	Description
1	GND	
2	GND	
3	I2C_SCL	CSP I^2C bus communication
4	I2C_SDA	CSP I^2C bus communication
5	CANL	CSP CAN bus communication (Linear LTC2875 CAN transceiver used)
6	CANH	CSP CAN bus communication (Linear LTC2875 CAN transceiver used)
7	GND	
8	TXEN_EX	Select TX mode in manual duplex mode
9	TXON_EX	Power on TX circuit
10	RXON_EX	Power on RX circuit

4.1.5 J402 - Debug

Molex Pico-Lock 1.50 mm pitch 504050-0691

Pin	Name	Description
1	GND	
2	UART RX	GOSH serial communication
3	UART TX	GOSH serial communication
4	SWCLK	Firmware upload/debug
5	SWDIO	Firmware upload/debug
6	RESETn	Firmware upload/debug

5 Data Interface

ANT2000 use the CubeSat Space Protocol (CSP) to transfer data to and from CSP nodes on-board the main system bus. CSP is a routed network protocol that can be used to transmit data packets between individual subsystems on the satellite bus and between the satellite and ground station. For more information about CSP please read the documentation on libcsp.org and on Wikipedia:
http://en.wikipedia.org/wiki/Cubesat_Space_Protocol

It's possible to control the board via CAN (default) or I^2C.

6 Debug Interface

The debug interface is a USART that uses the GomSpace Shell (GOSH) to present a console-like interface to the user. GOSH is a general feature present on all GomSpace products.

The console can be used during checkout and satellite integration of the antennas to send commands and inspect/set parameters.

7 Absolute Maximum Ratings

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the antennas. Exposure to absolute maximum rating conditions for extended periods may affect the reliability.

Symbol	Description	Min.	Max.	Unit
VIN	Input Supply voltage	8.0	18.0	V
PIN	Supply power draw	-	11	W
P_{in}	Absolute maximum input power at TX and RX ports		5	dBm
T_{amb}	Operating Temperature	-40	85	°C
T_{stg}	Storage Temperature	-40	85	°C

8 Electrical Characteristics

Symbol	Description	Min.	Max.	Unit
P_{sup_off}	Supply power, OFF	-	1.8	mW
P_{sup_idle}	Supply power, IDLE	-	200	mW
P_{sup_rx}	Supply power, RX mode	-	800	mW
P_{sup_tx}	Supply power, TX STANDBY	-	600	mW
P_{sup_tx}	Supply power, TX ACTIVE	-	11250	mW

9 Physical Characteristics

Description	Value	Unit
Mass (approximate – depends on mounting etc)	~ 110	g
Size (see chapter 12)	98 x 98 x 20.1	mm

10 RF Performance Characteristics

10.1 Receiver ISL

Symbol	Description	Min.	Max.	Unit
Gain,rx	RX Avg. Gain, 25°C	35	39	dB
	Gain ripple 25°C	-3	3	dB
	Gain ripple -40°C to 85°C	-3	-3	dB
NF,rx	RX Noise Figure, 25°C	2.5 (typ)	2.8	dB
	RX Noise Figure, -40°C to 85°C		3.5	dB
Freq, rx	RX frequency band	2200	2290	MHz

10.2 Receiver DUP

Symbol	Description	Min.	Max.	Unit
Gain,rx	RX Avg. Gain, 25°C	39	45	dB
	Gain ripple 25°C	-4	4	dB
	Gain ripple -40°C to 85°C	-5	-5	dB
NF,rx	RX Noise Figure, 25°C	2.0 (typ)	2.5	dB
	RX Noise Figure, -40°C to 85°C		3.0	dB
Freq, rx	RX band (ANT-2150-DUP)	2025	2110	MHz
	RX band (ANT-2090-DUP)	1980	2010	MHz

10.3 Transmitter

Symbol	Description	Min.	Max.	Unit
Pow.level 2	Pout	32.0 (typ)		dBm
	Pout ripple	-2	1	dB
	Pin	-13	-3	dBm
	DC Power (typical) (Vin 10V)	8.9	10.0	W
Pow.level 1	Pout	29.8 (typ)		dBm
	Pout ripple	-2	1	dB
	Pin	-16	-6	dBm
	DC Power (typical) (Vin 10V)	6.1	7.1	W
Pow.level 0	Pout	26.8 (typ)		dBm
	Pout ripple	-2	1	dB
	Pin	-19	-9	dBm
	DC Power (typical) (Vin 10V)	4.0	4.5	W
Pin,thr	Threshold for input detector – rising	-33	-24	dBm
Pin,thr,hys	Hysteresis for input power det.	2		dB
Pout,min	Min. output power for automatic TX	8	16	dBm
Dup,Act	Auto duplex activation time		140	µSec

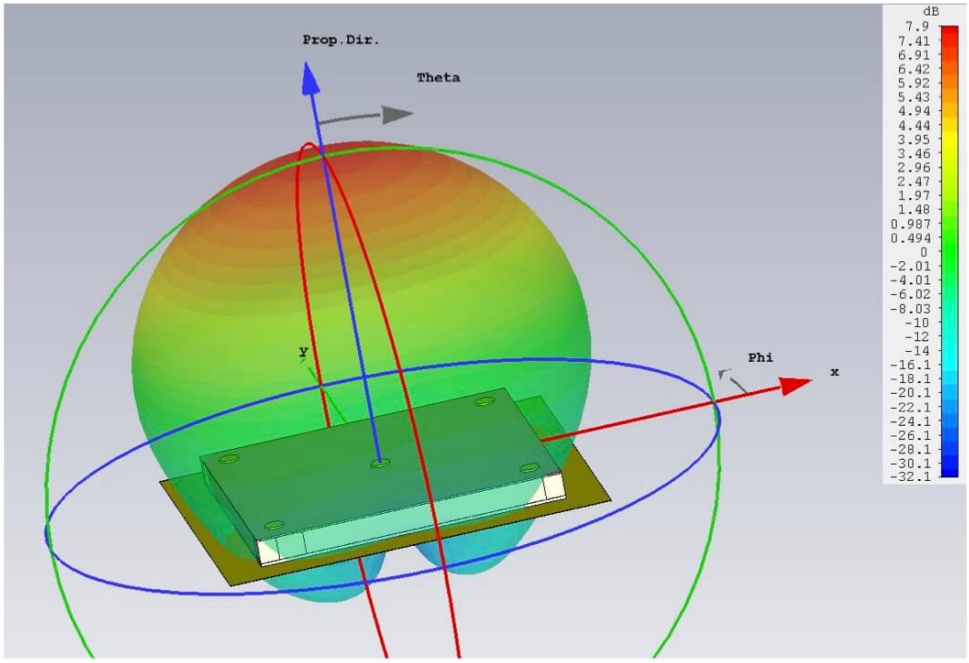
11 Antenna Performance

Figure below is a typical simulated radiation pattern, and a definition of the Phi and Theta angles used to describe the general antenna performance.

The antenna module includes connectors and matching circuit and has two feed ports which must be driven 90° out of phase to generate a circular polarized radiation.

On the electronics board a hybrid coupler generates the necessary quadrature signals. For the antenna measurements an adaptor board with the same hybrid coupler is used as test interface and the adaptor board loss (~0.30 dB) is compensated in the measurements.

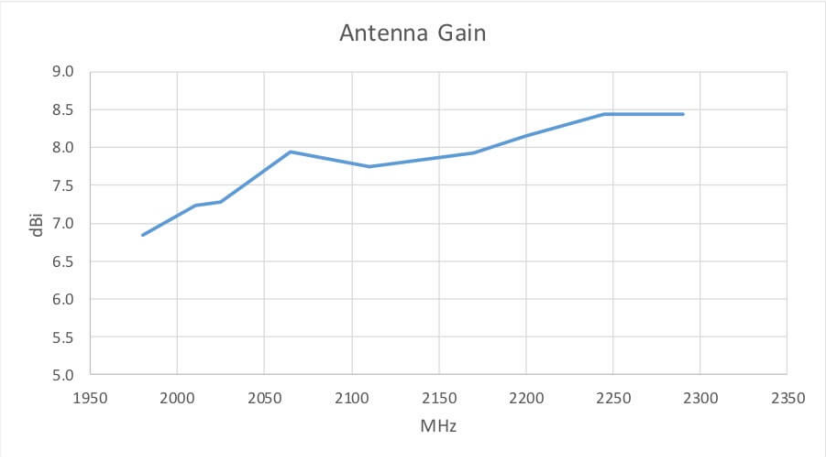
Due to symmetry of the feed ports both circular polarizations will have the same performance and the results shown in this section with be based on RHCP data.

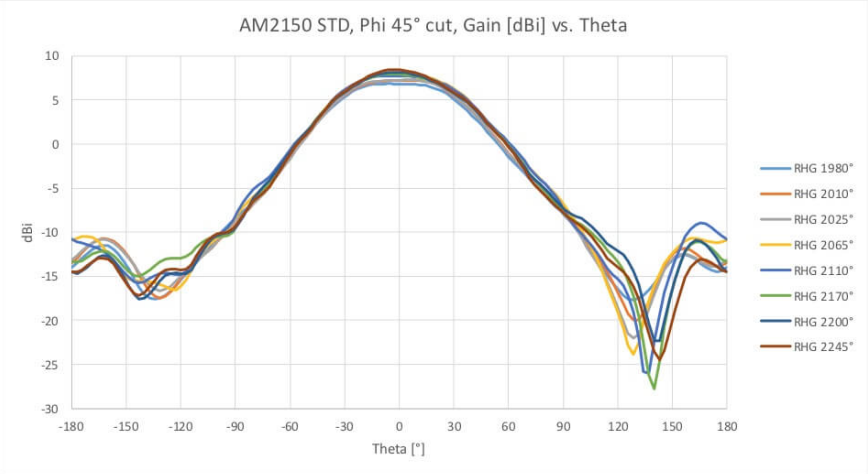


11.1 Standard Profile

11.1.1 Antenna Gain

All antenna radiation measurements are with a Satimo Ring setup using a 98 x 98 mm aluminum ground plane. Below antenna gain (including connector) and radiation pattern for Phi=45° (diagonal across ground plane) is shown.



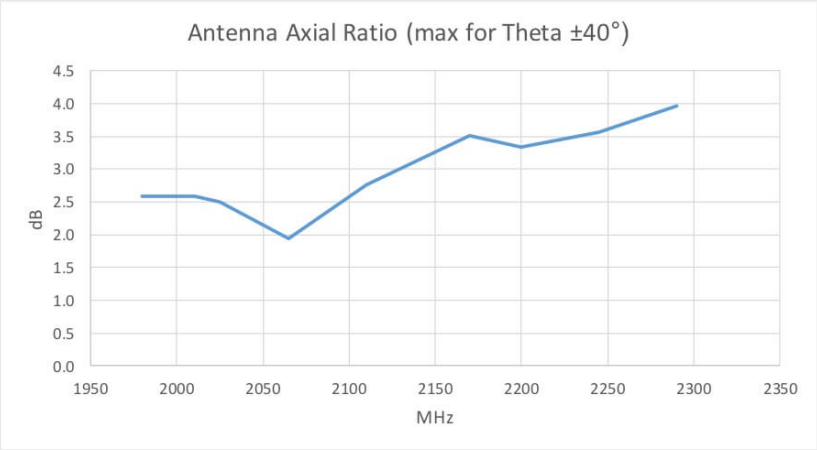


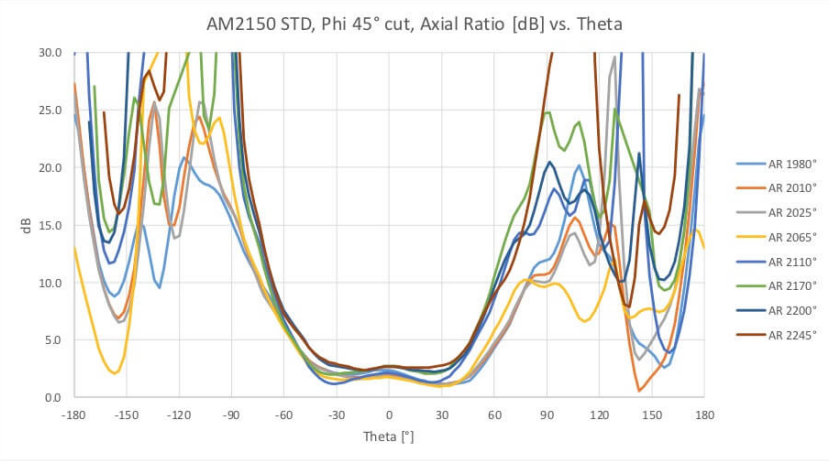
11.1.2 Axial Ratio

All three antennas are circular polarized, and product polarization is determined by the electronics PCB.

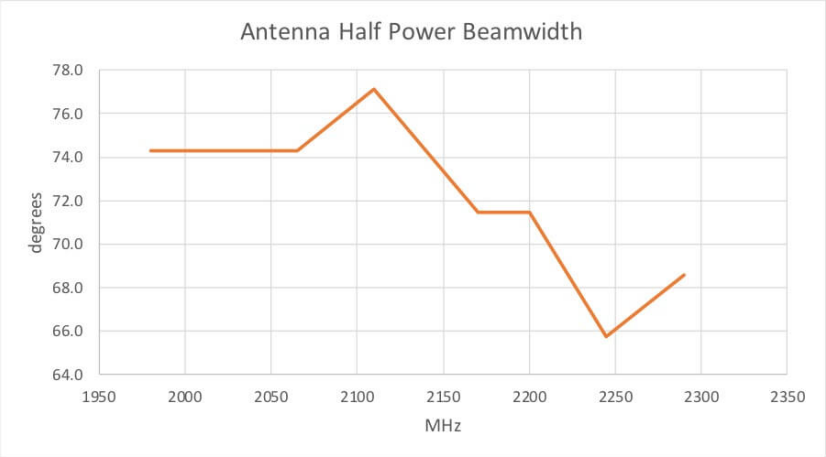
Below the maximum axial ratio for -40...40° elevation is shown as a function of frequency, and the axial ratio as a function of Theta for a Phi angle of 45°.

As seen the axial ratio is quite good within the halfpower beamwidth.

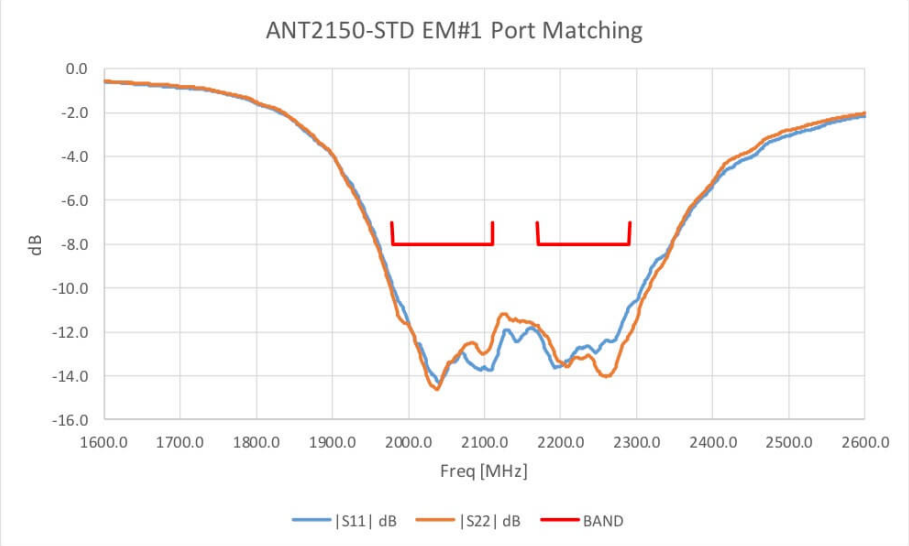




11.1.3 Half Power Beamwidth



11.1.4 Port Matching



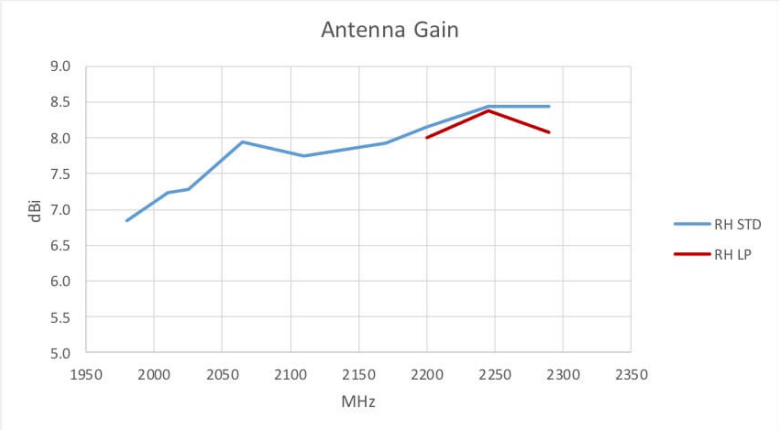
11.2 Low Profile

Only the ANT2150 ISL in the 2200-2290 MHz frequency band can use the low profile antenna.

The Half power beamwidth and the axial ration is virtually the same as for the standard antenna.

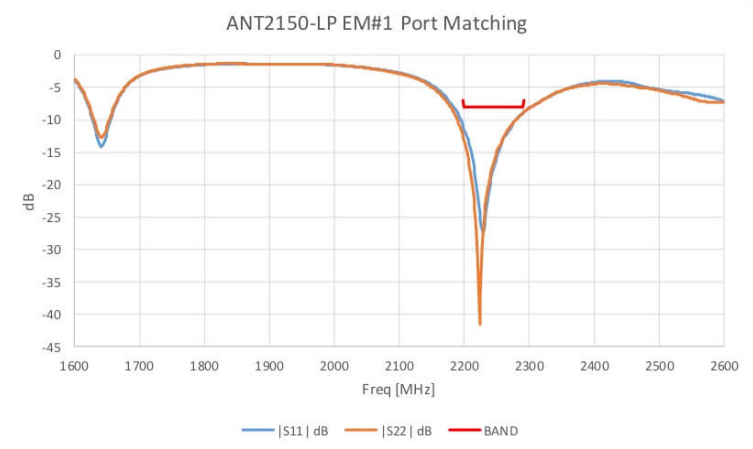
11.2.1 Antenna Gain

Performance within this band is marginally lower than the standard antenna primarily due to matching losses. Below the gain [dBi] is shown in comparison with the standard profile antenna.



11.2.2 Port Matching

The matching of the ports is significantly more narrowband than the standard antenna as shown below.

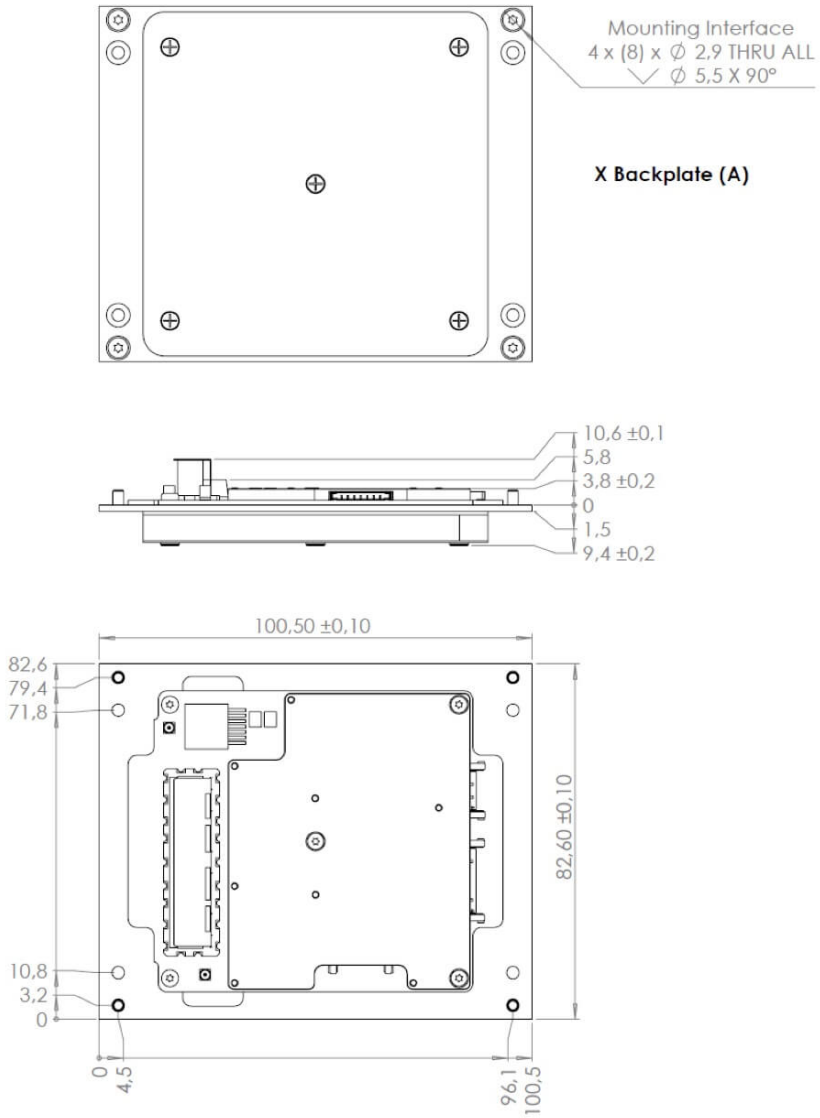


12 Mechanical Drawing

All dimensions in mm.

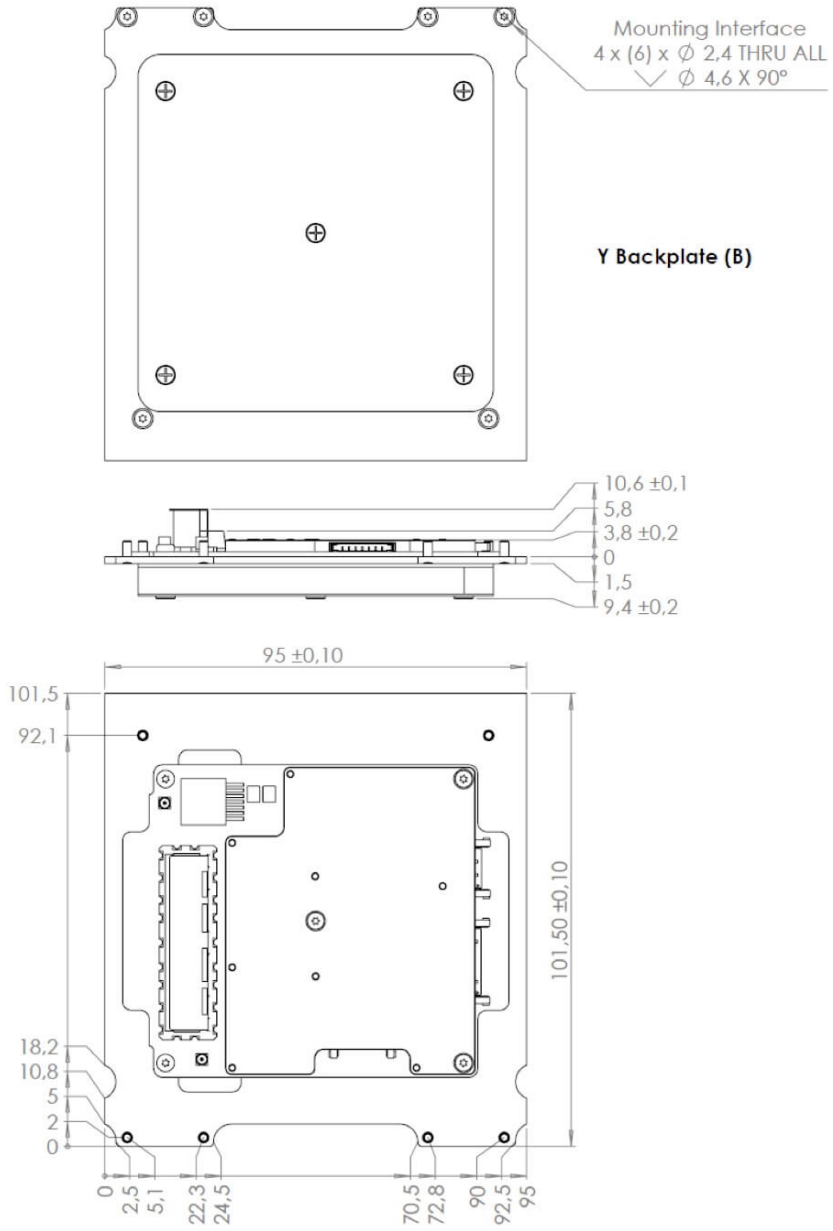
12.1 Type A

Depictured with an ANT2150 with standard profile antenna.



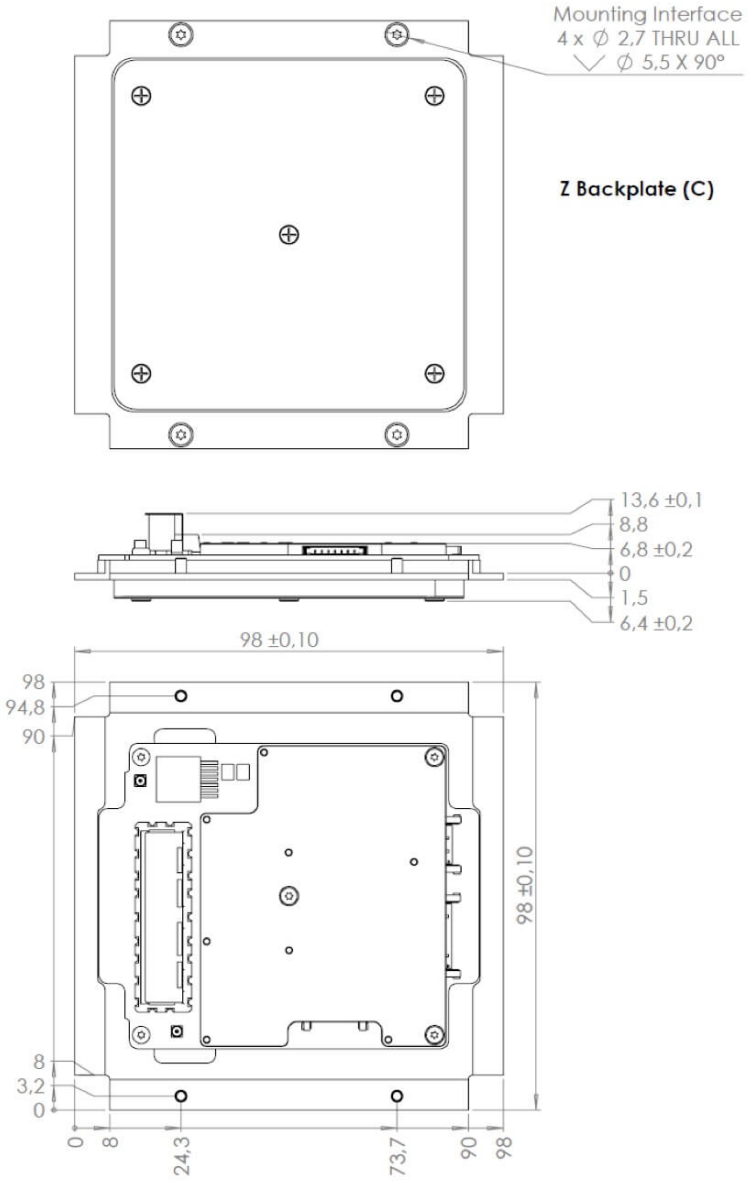
12.2 Type B

Depicture with an ANT-2150 with standard profile antenna.



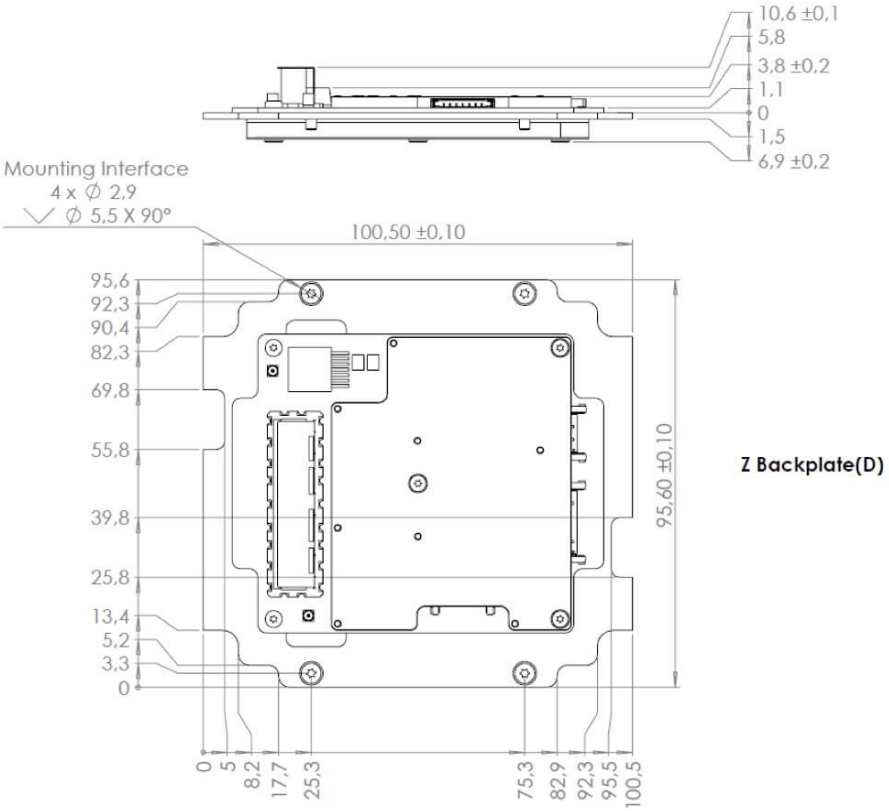
12.3 Type C

Depictured with an ANT-2150 with standard profile antenna.



12.4 Type D

Depictured with an ANT2150 with Low profile antenna.



13 Disclaimer

The information in this document is subject to change without notice and should not be construed as a commitment by GomSpace. GomSpace assumes no responsibility for any errors that may appear in this document.

In no event shall GomSpace be liable for incidental or consequential damages arising from use of this document or the software and hardware described in this document.