Universidade de Aveiro
2023

Diogo Miguel
Pires Ferreira da Silva

Low-cost image communication systems solutions for satellites and unmanned aerial vehicles

Soluções de sistemas de comunicação de imagens de baixo custo para satélites e veículos aéreos não tripulados

Universidade de Aveiro
2023

Diogo Miguel
Pires Ferreira da Silva

Low-cost image communication systems solutions for satellites and unmanned aerial vehicles

Soluções de sistemas de comunicação de imagens de baixo custo para satélites e veículos aéreos não tripulados

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor António Navarro, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor José Miguel da Silva Bergano, co-fundador e CEO da ATLAR Innovation.

o júri / the jury

presidente / president	**Professor Doutor Pedro Renato Tavares de Pinho**
	Professor Associado, Universidade de Aveiro
vogais / examiners committee	**Professor Doutor António Manuel Raminhos Cordeiro Grilo**
	Professor Auxiliar, Instituto Superior Técnico de Lisboa
	Professor Doutor António José Nunes Navarro Rodrigues
	Professor Auxiliar, Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Quero agradecer à minha família e namorada pelo apoio incondicional e paciência durante esta viagem. Aos meus orientadores, Professor Doutor António Navarro e ao Professor Doutor Miguel Bergano pela oportunidade de trabalhar neste projeto e pelo apoio e orientação, que me permitiram crescer e explorar novos horizontes. Ao projeto Eye In the Sky pela bolsa de estudo, por me ter disponibilizado todo o material necessário para o desenvolvimento desta dissertação e por me ter permitido participar na IAC-22 em Paris com o projeto pi National Oceanographic and Atmospheric Administration (piNOAA). Ao projeto Doppler [1] e ao falecido Professor Doutor António Batel Anjo pela oportunidade de participar num evento em Maputo para a promoção da educação e do conhecimento. Ao Instituto de Telecomunicações de Aveiro por todas as condições disponibilizadas. À Universidade de Aveiro por este percurso académico. E aos meus amigos, com quem partilhei esta aventura académica, pelo companheirismo e partilha de experiências.

I want to express my gratitude to my family and girlfriend for their unwavering support and patience during this journey. To my supervisors, Prof. António Navarro, Ph.D. and Prof. Miguel Bergano, Ph.D., for the opportunity to work on this subject and for their support and guidance, which allowed me to grow and explore new horizons. To the Eye In The Sky project for the scholarship, for providing me with all the necessary facilities for the development of this dissertation. To the project piNOAA for allowing me to participate in IAC-22 in Paris. To the Doppler project [1] and the late Professor António Batel Anjo, Ph.D. for the opportunity to participate in an event in Maputo to promote education and knowledge. To Instituto de Telecomunicações de Aveiro for all the provided facilities. To Universidade de Aveiro for this academic journey. And to my friends, with whom I have shared this academic adventure, for their companionship and shared experiences.

Palavras Chave: Low-cost, NOAA, Raspberry Pi, SDR, GNU Radio, Wildfires, Storms

Resumo

Nos últimos anos, a crescente ameaça de tempestades e incêndios florestais, agravada pelas alterações climáticas e pela atividade humana, tem exigido soluções inovadoras e economicamente viáveis para a monitorização em tempo real e resposta a desastres. Esta dissertação explora a integração de tecnologias de receção de imagens de satélite de baixo custo para ajudar as populações em áreas remotas a estarem preparadas a tempo para a chegada de tempestades. Em alternativa, também pode ser usado para prever precipitação e auxiliar na gestão de recursos hídricos para a agricultura, ou como um sistema de "backup" em estações de observação espacial. Tendo sido no final produzido um protótipo de uma estação de receção de imagens meteorológicas de satélite independente que capta e publica as imagens em redes sociais automaticamente. No que diz respeito aos incêndios, é um facto bem conhecido que, nos últimos anos, Portugal tem sofrido com os incêndios florestais extremos no auge do verão, ocorrendo quer de forma natural devido à falta de limpeza florestal, quer por ação humana. De qualquer forma, eles têm um terrível impacto no ambiente, custando a vida de pessoas e dos animais que habitam na área ou nas proximidades, e causando danos irreversíveis. Todos os anos, os bombeiros arriscam as suas vidas combatendo estes incêndios. Portanto, para ajudá-los a planear melhor as suas ações, técnicas de observação da Terra (ou seja, balões de alta altitude e veículos aéreos não tripulados) podem ser usadas para fornecer-lhes um "olho no céu", mostrando-lhes o progresso do incêndio em tempo real. Para isso, foram exploradas diversas soluções de baixo custo para as estações terrestres bem como para o "payload" com capacidade para transmitir imagens em tempo real de incêndios florestais. No fim foi produzido um sistema "payload" - estação terrestre capaz de transmitir imagens na frequência de 2.4 GHz, testado até uma distância de 2.2 km.

Keywords: Low-cost, NOAA, Raspberry Pi, SDR, GNU Radio, Wildfires, Storms

Abstract

In recent years, the escalating threat of storms and wildfires, exacerbated by climate change and human activity, has necessitated innovative and cost-effective solutions for real-time monitoring and disaster response. This dissertation explores the integration of low-cost satellite image reception technologies to help populations in remote areas be ready in time for incoming storms. Alternatively, it can also be used to predict rain and help manage water resources for agriculture as well as a backup system in space observation stations. In the end, a prototype was produced of an independent satellite weather image receiving station that captures and publishes the images on social networks automatically. Regarding the fires, it is a well-known fact that, in the past few years, Portugal has been suffering from extreme wildfires in the peak of summer, either naturally occurring due to the lack of forest cleaning or by human hand. Either way, they have a terrible impact on the environment, costing the lives of people and the animals that inhabit there or nearby and causing irreversible damage. Every year, firefighters risk their lives fighting these fires. So, in order to help them better plan their actions cost-effective earth observation techniques (i.e., high-altitude balloons and unmanned aerial vehicles) could be used to provide them with an "eye in the sky", showing them the progress of the fire. For that, several low-cost solutions for the ground stations were explored as well as for the payload capable of transmitting real-time images of forest fires. Finally, a payload-ground station system was produced capable of transmitting images on the 2.4 GHz frequency, tested up to a distance of 2.2 km.

Contents

List of Figures

List of Tables

Glossary

3D	3 Dimentions
ADC	Analog-to-Digital Converter
AOS	Acquisition Of Signal
API	Application Programming Interface
APRS	Automatic Packet Reporting System
APT	Automatic Picture Transmission
AX.25	Amateur X.25
BPSK	Binary Phase-Shift Keying
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DC	Direct Current
DVB-T	Digital Video Broadcasting - Terrestrial
EIS	EyeInTheSky
FCT	Fundação para a Ciência e a Tecnologia
FFT	Fast Fourier Transform
FM	Frequency Modulation
FTP	File Transfer Protocol
GNSS	Global Navigation Satellite Systems
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GUI	Graphical User Interface
HAB	High Altitude Balloon
HAT	Hardware Attached on Top
HDTV	High-Definition Television
HQ	High Quality
HRPT	High Resolution Picture Transmission
I2C	Inter-Integrated Circuit
IO	Input/Output
IoT	Internet of Things
IP	Internet Protocol
IR	Infrared
IT	Instituto de Telecomunicações
KISS	Keep It Simple Stupid
LED	Light Emitting Diode
LNA	Low Noise Amplifier
LoRa	Long Range
LOS	Loss Of Signal
MP	Mega Pixel
NASA	National Aeronautics and Space Administration
NGO	Non Governmental Organization
NMEA	National Marine Electronics Association
NOAA	National Oceanographic and Atmospheric Administration
NRA	Núcleo Radioamadores
OS	Operating System
PASO	Pampilhosa da Serra Space Observatory
PA	Power Amplifier
PC	Personal Computer
PCB	Printed Circuit Board
PIL	Python Imaging Library
piNOAA	pi National Oceanographic and Atmospheric Administration
PD	Power Delivery
POES	Polar Orbiting Environmental Satellites
PTA	Pilotless Target Aircraft
PVC	Polyvinyl Chloride
QFH	Quadrifilar Helical
RAM	Random Access Memory

RF	Radio Frequencies
RGB	Red Green Blue
RSSI	Received Signal Strength Indicator
RTC	Real-Time Clock
SDR	Software Defined Radio
SIRESP	Sistema Integrado de Redes de Emergência e Segurança de Portugal
SMB	Server Message Block
SSH	Secure Shell
TCP	Transmission Control Protocol
TDOA	Time Difference Of Arrival
TETRA	Terrestrial Trunked Radio
TLE	Two-Line Element
TNC	Terminal Node Controller
UART	Universal asynchronous receiver/transmitter
UAV	Unmanned Aerial Vehicle
UHF	Ultra High Frequency
USB	Universal Serial Bus
VHF	Very High Frequency
WAV	WAVEform Audio Format
Wi-Fi	Wireless Fidelity

Chapter 1

Introduction

1.1 Context and Motivation

With the most recent trend of Internet of Things (IoT), where everything is connected, it becomes fundamental to guarantee communications with all the devices. On the other hand in civil emergency scenarios, where telecommunication infrastructures are destroyed, it is of the utmost importance to guarantee a redundant service that fulfills the minimum requirements of information transfer for the most affected zones or the national security entities (Proteção Civil). This happens in forest wildfires, earthquakes, storms, and other kinds of scenarios where there are usually electrical and telecommunication failures.
For this reason, this work focused on the development of two autonomous low-cost Earth Observation systems. One is piNOAA and the other is Eye In the Sky (EIS).

piNOAA's goal is to be a low-cost independent daily meteorological weather service using a Raspberry Pi as a processing platform for the acquired data and to be deployed in remote areas so that the local population can have weather forecasts and be warned of incoming storms. In addition, it can be used as a backup service for PASO. The National Oceanographic and Atmospheric Administration (NOAA) satellites transmit low-resolution images in Very High Frequency (VHF) and high-resolution images in the L band. To acquire the images transmitted in that band, the satellite must be tracked with a directional antenna.

EIS's goal is to design and develop an aerial platform, such as an Unmanned Aerial Vehicle (UAV) or a High Altitude Balloon (HAB), to support communications in emergencies, while at the same time providing real-time images of the fire progress to the responsible entities so that they can better deal with the emergency. In the recent national incidents during wildfire fights, it has been identified that an improvement could be made to the current emergency communications and that there is a real difficulty in mapping and predicting the fire propagation fronts.

1.2 Goals

The purpose of this dissertation is to develop two low-cost systems for Earth observation. The first one is for meteorological analysis to provide weather forecasts in remote areas to warn local populations about incoming storms and to aid their agriculture.

The other system provides aerial images of ongoing wildfires to assist firefighters during the fire progress to better plan and be more efficient in their mission. Thus, it is necessary to design a payload capable of taking pictures, determining its position using GPS, and sending data to a ground station. The ground station should be able to track the payload with a directional antenna by using a pointing mechanism capable of a movement from 0° to 360° in the Azimuth and from 0° to 90° in Elevation. It should be simple to set it up some kilometers away from the fire or to be mounted on top of a vehicle.

1.3 Document Structure

This dissertation is divided into four more chapters and is organized as follows:

Chapter 2 shows some work in literature, on Earth Observation, and some Tracking Techniques are described.

Chapter 3 introduces the project piNOAA and describes the development of the hardware and software and the obtained results.

Chapter 4 introduces the project EIS and describes the development of the two iterations of hardware and software and the obtained results.

Chapter 5 concludes the dissertation and details the future work of both projects.

1.4 Main Contributions

The work developed during this master's dissertation allowed to achieve the proposed objectives, and with that, to contribute to future works on image communication systems for satellites and unmanned aerial vehicles, especially with:

- The development of a low-cost, independent, automatic weather station to receive weather data from NOAA satellites, which can publish the decoded images on social networks.
- The development of a low-cost communication system to transmit images acquired from a payload carried by a HAB or a UAV to a ground station, using SDRs, to assist the emergency response entities during wildfires.

Chapter 2

Background

2.1 Earth Observation

In the following subsections, some history of the development of observation balloons, satellites, and UAVs will be described.

2.1.1 Observation Balloons

The history of the balloon started in 1783 in France with Etienne and Joseph Montgolfier [2] who started experimenting with paper bags filled with hydrogen which led them to the conclusion that a buoyant force would cause the ascent of the bag if the gas inside was lighter than the air outside. Later they tried to create a lighter gas with the combustion of a mixture of moistened straw and wool and by accident and, unaware of it, they created the first hot air balloon. This eventually led to the first man flight in the same year. Fig. 2.1 shows a drawing of the first manned flight in a hot air balloon.

Figure 2.1: First manned flight in a hot air balloon (Montgolfière) on the 21st of November, 1793 [2].

During the French Revolutionary Wars, it had its first military usage in the battle of Fleurus, in 1794. It was used to observe the position of the enemy army and report their location to their troops, as shown in Fig. 2.2. Furthermore, this kind of usage of observation balloons was also done, later, during the American Civil War (1861–65) [3] and World War I (1914–18) [4], for example.

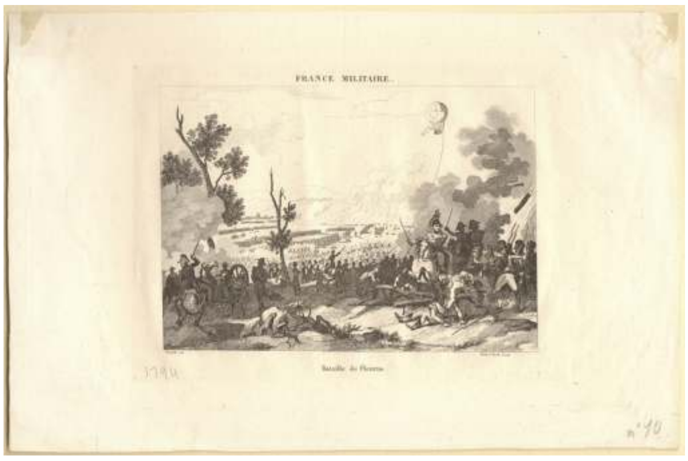


Figure 2.2: Battle of Fleurus, 1794 [5].

More importantly, soon after it was invented, it had its first usage in investigation, opening the doors of atmospheric research and later offering the possibility to study radiations from outer space that could not pass the lower layers of the atmosphere. After an absence of scientific flights between 1805 and 1850, meteorologists from different nations started to make extensive use of balloons to measure meteorological parameters. This manned flight technique determines wind direction at different altitudes. Later, at the end of the century, it was performed the first unmanned flight, a sounding balloon, with a meteorograph on board. By using this technique the discovery of the stratosphere was done by Teissertenc de Bort in France and by Assmann in Germany almost simultaneously [2]. In the 1910s some studies on air ionization were conducted which led to the discovery of cosmic radiation, originally called ultra-radiation, which was explained as penetrating radiation entering the atmosphere from the outside, but not from the sun. These studies were interrupted by the First World War and later resumed in the 1920s. Nowadays, twice a day, another kind of observation balloon, the weather balloon, is launched from almost 900 locations worldwide equipped with a radiosonde which records the atmosphere pressure, the temperature, and the relative humidity as it ascends. By tracking its position it is possible to determine the wind direction and velocity, and the data collected is transmitted via radio and analyzed using forecast models. With the output data, meteorologists can make forecasts and predict storms all over the world [6].

2.1.2 Satellites

It has been around 77 years since the vision of having communication devices, and satellites, orbiting our planet. They provide communication services almost everywhere, especially in remote areas, where it is difficult to reach or expensive to produce the terrestrial means to do so. This vision became a reality when the intercontinental ballistic missile was developed, creating the power to launch a satellite. Therefore, in 1957, the Soviet Union launched the first satellite, Sputnik 1 [7], as seen in Fig. 2.3.

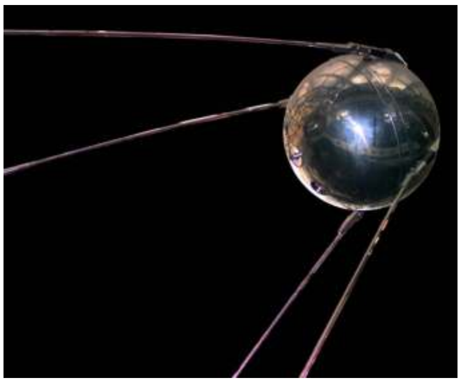


Figure 2.3: Sputnik 1, 1957 [8].

This forced the United States of America to launch their first satellite at the beginning of 1958, Explorer 1, which was "countered" by the Soviets with the launch of a 3-ton "flying laboratory". Thus, the space race began. Later, in 1963, the satellite TIROS VIII with an experimental version of Automatic Picture Transmission (APT), developed by National Aeronautics and Space Administration (NASA), was launched. TIROS VIII followed by two Nimbus satellites successfully demonstrated the APT system by transmitting thousands of pictures of Earth's atmosphere to ground stations all over the world [9]. This allowed the scientific study of Earth's atmosphere and provided valuable data for weather forecasting, thus allowing, for example, more efficient farming by saving water when rain is expected.

In the 70s, satellites started to deliver content to the general public in the form of television broadcasts. In the same decade, satellites allowed for two-way voice and data communication anywhere in the world [10]. Nowadays satellites, namely SpaceX's Starlink which has a constellation of satellites, can even provide low latency internet access almost anywhere in the world.

2.1.3 UAV

Some years after the Wright brothers successfully built and flew the first airplane in 1903, the idea of having an unmanned flying machine was born. The prototype of a UAV was the implementation of autopilot on an aircraft where the pilot was only responsible for the takeoff and landings [11]. This technology was later implemented in early versions of flying bombs. Then after the First World War, because of the improvement and development of fighter aircraft, the training of the pilots had to be adapted. To simulate these new and improved aircraft the Pilotless Target Aircraft (PTA) was created. This new type of UAV was a big innovation because it was the first type of UAV that could return at the end of its mission if it was not destroyed during the pilot training. After the Second World War, the research in UAV continued and was improved with the development of automatic systems. During the Cold War UAVs were also used for strategic reconnaissance. However, the usage of UAVs is not limited to the military. Nowadays we can find UAVs being used in several fields like UAV Monitoring, IoT Networks, and Disaster Management. In the case of UAV Monitoring it can facilitate the tasks of emergency services, monitor traffic in areas of frequent accidents, and security organizations on the ground by using cameras for visual assistance or other sensors to measure air quality in the area where the UAV is flying. In addition, it can be used to extend the coverage of terrestrial cellular networks and improve mobile communications [12]. All of these applications and more can be seen in Fig. 2.4 and Fig. 2.5.

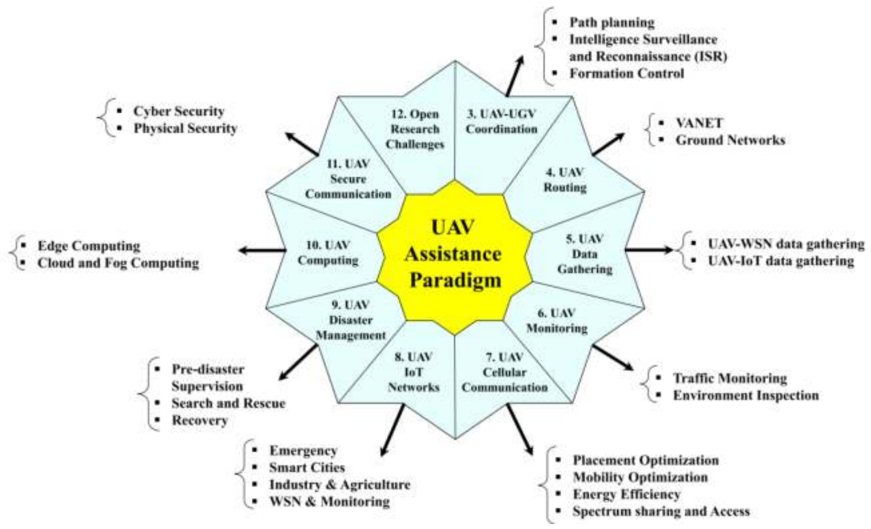


Figure 2.4: UAV Assistance [12].

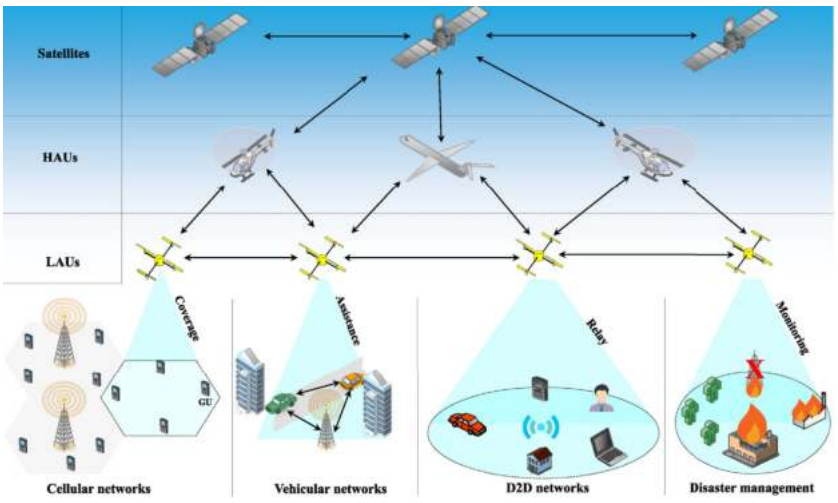


Figure 2.5: UAV Potential usages [12].

In order to receive high-resolution images from a UAV from a long distance, a directional antenna with a tracking mechanism has to be used, and some tracking techniques will be presented in the next section.

2.2 Tracking Techniques

In the following subsections, some tracking techniques will be described.

2.2.1 RSSI - Return Signal Strength Indicator

Received Signal Strength Indicator (RSSI) uses the measured power of the received signal to calculate the distance between the target and the ground station. This method is low cost, energy efficient, does not require much memory, and does not require much processing capabilities. However, this has a low positioning accuracy because it cannot calculate the azimuth and elevation position and it cannot calculate the polarization of the signal. Also, the presence of white and colored noise can reduce even more its low accuracy. Although this technique has low accuracy it can be used as an auxiliary measuring component for other tracking techniques [13]. In the master thesis of Gustafsson and Henriksson from the Luleå University of Technology, Sweden, [14] it was designed a tracking antenna composed of 4 Yagi antennas and depending on the RSSI on each antenna it was possible to track the UAV. Although the experiment was successful it showed that it was not that accurate.

2.2.2 TDOA - Time Difference Of Arrival

In contrast to the previous technique, Time Difference Of Arrival (TDOA) measures the time difference of arrival of the same signal in different nodes to calculate the cross-correlation of the received signal to pinpoint its location. This method is low cost and is of low complexity of implementation. However, this technique requires that the multiple nodes are perfectly synchronized for the best possible accuracy [13].

2.2.3 Use of GPS - Global Positioning System

The significant errors of the previous techniques limit the versatility of their implementations, therefore, it is required to make use of a global positioning system module. The GPS tracking system is the most commonly used approach in getting the location in real-time. This method utilizes GPS technology to obtain the latitude, longitude, and altitude coordinates of both the UAV and tracking device. These coordinates are then used to determine the azimuth and elevation angles, as well as the distance between the UAV and tracking system, by applying formulas such as the haversine distance, bearing formula, and spherical trigonometry. The latter offers high accuracy and precision compared with the previous two techniques. This technique suffers if there is bad GPS reception on either end which results in a lower accuracy. Some factors that can contribute to the appearance of errors are the presence of tall buildings or mountains that can reflect the GPS signal [13].

This technique is the one that will be used for the Eye In The Sky project since it appears to be the most precise and simple to implement from the previously presented.

2.2.4 Use of TLE - Two-Line Element

The Two-Line Element (TLE) set is the mean Keplerian orbital element at a certain point in time for each reported space object. It is generated using the simplified general perturbations theory and is reasonably accurate for long periods of time [15]. It is publicly available in domains such as, for example, celestrak.org with the following format:

```
AAAAAAAAAAAAAAAAAAAAAAAA
1 NNNNNU NNNNNAAA NNNNN.NNNNNNNN +.NNNNNNNN +NNNNN-N +NNNNN-N N NNNNN
2 NNNNN NNN.NNNN NNN.NNNN NNNNNNN NNN.NNNN NNN.NNNN NN.NNNNNNNNNNNNNN
```

where Line 0 is the name of the space object and Lines 1 and 2 are the Two-Line Element set format. Tables 2.1 and 2.2 describe with better details each line.

Table 2.1: TLE Line 1 [16].

Line 1	
Column	**Description**
01	Line Number of Element Data
03-07	Satellite Number
08	Classification (U=Unclassified)
10-11	International Designator (Last two digits of launch year)
12-14	International Designator (Launch number of the year)
15-17	International Designator (Piece of the launch)
19-20	Epoch Year (Last two digits of year)
21-32	Epoch (Day of the year and fractional portion of the day)
34-43	First Time Derivative of the Mean Motion
45-52	Second Time Derivative of Mean Motion (Leading decimal point assumed)
54-61	BSTAR drag term (Leading decimal point assumed)
63	Ephemeris type
65-68	Element number
69	Checksum (Modulo 10) (Letters, blanks, periods, plus signs = 0; minus signs = 1)

Table 2.2: TLE Line 2 [16].

Line 2	
Column	**Description**
01	Line Number of Element Data
03-07	Satellite Number
09-16	Inclination [Degrees]
18-25	Right Ascension of the Ascending Node [Degrees]
27-33	Eccentricity (Leading decimal point assumed)
35-42	Argument of Perigee [Degrees]
44-51	Mean Anomaly [Degrees]
53-63	Mean Motion [Revs per day]
64-68	Revolution number at epoch [Revs]
69	Checksum (Modulo 10)

The TLE can later be used in a propagator model to propagate the orbit of the space object in order to be tracked.

As for project piNOAA, since the NOAA satellites' orbits can be propagated from the publicly available TLE data, this was the preferred method for the data acquisition with an omnidirectional or a directional antenna.

In the following Section, we describe our solution for earth observation based on a Raspberry Pi.

Chapter 3

piNOAA: an independent daily Earth Observation service using a Raspberry Pi data processing platform

3.1 Project Requirements

For this project, it was necessary to develop an independent, low-power, and cheap weather station to receive weather images of the atmospheric activity in remote areas where the local population does not have access to meteorological data so that they can be warned about incoming storms. It also can be used as a backup to assist operations at PASO. Their antenna and observatory can be seen in Fig. 3.1. These weather images [17] are captured by the Polar Orbiting Environmental Satellites (POES) NOAA satellites which can cover, daily, the whole surface of the planet thanks to their polar orbits. It has several imaging sensors named Channel 1, 2, 3A, 3B, 4, and 5 and a satellite operator chooses two of them called Channel A and B. During the day the images on channel A are taken using a near-visible sensor and during the night using an Infrared (IR) sensor. Channel B always has an image from an IR sensor.

Figure 3.1: PASO – In central Portugal.

3.2 Hardware - Receiver Setup

The system is made of a Quadrifilar Helix (QFH) antenna and the following components, as seen in Fig. 3.2 and Fig. 3.3, in the same numeric order:

1. Software Defined Radio (SDR)
2. Direct Current (DC) block
3. Low Noise Amplifier (LNA)
4. Raspberry Pi 3 Model B+

All the previous items are encased in a box to isolate them from the outside environment.

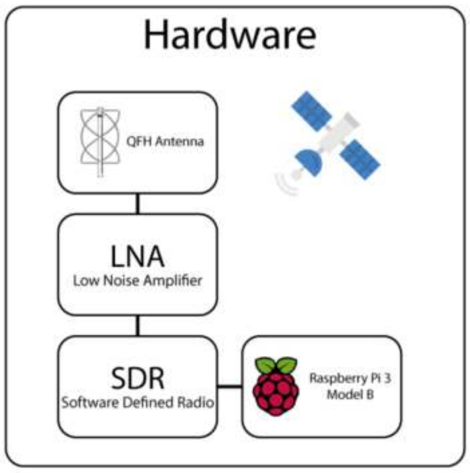


Figure 3.2: Block diagram for the hardware setup.

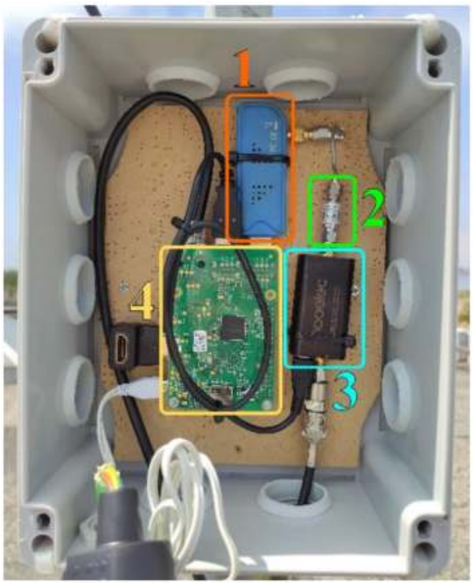


Figure 3.3: Electronics setup.

3.2.1 QFH Antenna

Since the goal is to build an autonomous weather station, the chosen antenna was of the QFH type because of its omnidirectional radiation pattern in a single hemisphere, as shown by Fig. 3.4 and because of its' circular polarity, matching the one used to transmit data on the NOAA satellites. They are constantly moving, which causes linearly polarized signals to continuously change their polarization. This can weaken the signal when it is received by a linearly polarized antenna on the ground, so to improve signal stability, both satellites and ground stations use circularly polarized antennas. Since it is possible to use a linearly polarized antenna on the ground to receive signals from POES satellites, this results in significant signal loss due to cross-polarization between the transmitting and receiving antennas [18]. Also, the station was mounted on the rooftop of Instituto de Telecomunicações in Aveiro while testing (currently mounted on the rooftop of Oswela in Maputo, Mozambique). In that way, we did not have to change the orientation of the antenna manually according to the satellite passage. The antenna dimensions were calculated with an online calculator [19]. It was designed to be more sensitive at the frequency of 137.5 MHz because NOAA 15, 18, and 19 operate on the frequencies shown in Table 3.1.

Table 3.1: Characteristics of NOAA 15, 18 and 19 satellites [9].

Characteristics/Satellites	NOAA 15	NOAA 18	NOAA 19
Automatic Picture Transmission(APT) (MHz)	137.6200	137.9125	137.1000
High Resolution Picture Transmission (HRPT) (MHz)	1702.5	1707.0	1698.0
Altitude (Km)	807	854	870
Period (minutes)	101.10	102.12	102.14

It is built with Polyvinyl Chloride (PVC) pipes, coaxial cables, and hot glue to isolate the open areas. Those are all cheap and easy materials to find. The same website that calculates the dimensions also has guides on how to solder, represented in Fig. 3.6, and assemble everything. Our antenna has a height of 70 cm and a width of 25 cm. This part costs around 30€ with some leftover materials. Fig. 3.5 shows the system with the antenna tuned to 137.5 MHz. This antenna was built by students from the Núcleo Radioamadores (NRA), which was no longer operating at the time.

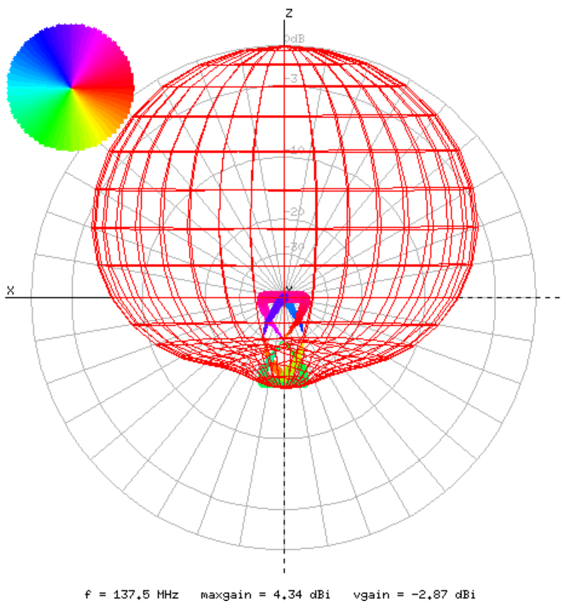


Figure 3.4: QFH Antenna radiation pattern on the vertical plane [20].

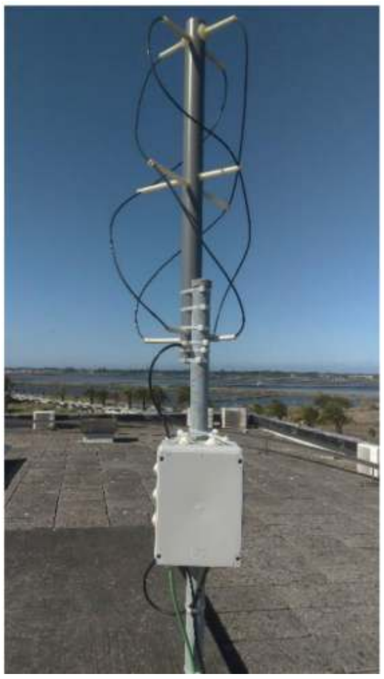


Figure 3.5: Projected NOAA satellites receiver setup [9].

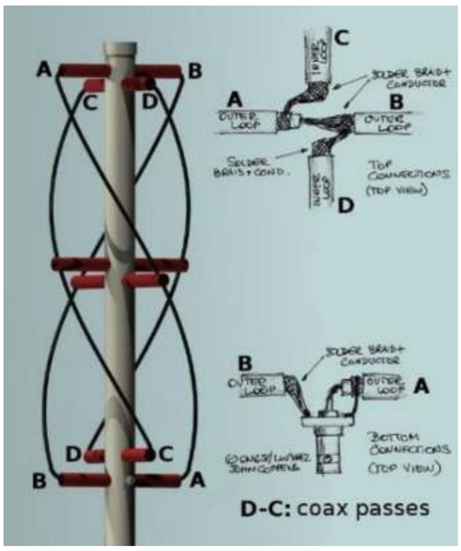


Figure 3.6: QFH Antenna wiring diagram [21].

3.2.2 SDR

Figure 3.7: Nooelec NESDR Mini 2 USB RTL-SDR [22].

We decided to use Nooelec NESDR Mini 2 USB RTL-SDR, shown in Fig. 3.7, because of its reduced size and low power consumption as well as it uses the RTL2832U, from Realtek, which is a high-performance Digital Video Broadcasting - Terrestrial (DVB-T) demodulator that supports USB 2.0. It has an embedded 8-bit Analog-to-Digital Converter (ADC). This is typically used for High-Definition Television (HDTV) reception, but hackers and modders found that it could be used as a cheap and general purpose SDR by pairing it with a R820T2 Tuner. This allows the reception of Radio Frequencies (RF) signals from 500 kHz – 1766 MHz, which is more than what is needed to receive APT signals. This device only draws 270 - 280 mA from the Raspberry Pi 3 Model B+ USB hub, which has a limit of 500 mA per USB port and a total limit of 1.2 A on all ports combined. This part costs around 29€.

An alternative for this SDR would be an Airspy R2 SDR Rx which has a higher dynamic range compared to the one we chose, but it costs 232€. There is also the Airspy Mini SDR Rx with a similar performance to the Airspy R2 SDR Rx and a similar form factor compared to the chosen SDR, but it costs 144.90€. These two are compatible with the same software used in this project but because of their prices, it would increase substantially the final price of the whole system.

3.2.3 DC Block

Figure 3.8: BLK-18-S+.

To avoid DC signals from entering the system, a BLK-18-S+ [22], as shown in Fig. 3.8, was used between the LNA and the SDR. This specific component was chosen because it has an impedance of 50 Ω and it has a low insertion loss, between 0.03 dB and 0.05 dB, for the projected frequency (137.5 MHz), which means the received signal is not significantly attenuated. This component costs around 5.50€.

3.2.4 LNA

Figure 3.9: Nooelec SAWbird+ NOAA LNA.

To amplify the received signal a Nooelec SAWbird+ NOAA LNA [23] was used, as shown in 3.9. This is a low-noise amplifier with a band-pass filter with a center frequency of 137.5 MHz, a bandwidth of 2.6 MHz, and a gain of 41dB at that center frequency. This can be seen in Fig. 3.10 and in Table 3.2. It is specifically used to amplify the received signals from the NOAA satellites, or any other satellite that operates with signals within 136 - 138 MHz.

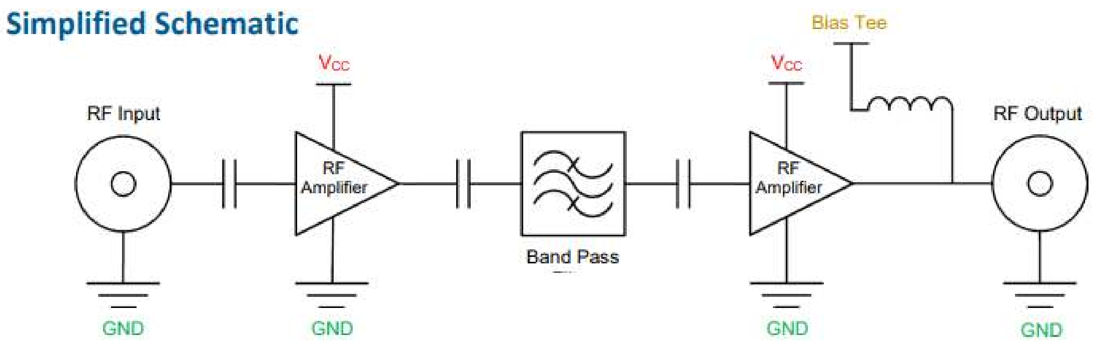


Figure 3.10: Nooelec SAWbird+ NOAA simplified schematic [24].

It can be powered in 3 ways (USB, pin header, or Bias-Tee). It was decided to power it via USB because the Raspberry Pi still has 3 ports available and it still can provide up to 920mA, and it only consumes 175mA, as seen in Table 3.2. This part costs 45€.

Table 3.2: Nooelec SAWbird+ NOAA Electrical Specifications [24].

Parameter	Symbol Min	Typical Max	Unit
3dB Bandwidth	BW	2.6	MHz
10dB Bandwidth	BW	3	MHz
Center Frequency	fo	137.5	MHz
Gain @137.5MHz	S21	41	dB
Input Return Loss @137.5 MHz	S11	-9	dB
Output Return Loss @137.5 MHz	S22	-9.5	dB
Output P1dB	OP1dB	20	dBm
Noise Figure total	NF	1.1	dB
Noise Temperature total	TN	84	K
Supply Current (USB)	l(USB)	175	mA
Supply Current (Bias -T)	I(Bias -T)	170	mA

3.2.5 Raspberry Pi 3 Model B+

Figure 3.11: Raspberry Pi 3 Model B+.

To process the received data, upload it, and power the LNA and the SDR, it was decided to use a Raspberry Pi 3 Model B+, as shown in Fig. 3.11, running the latest version of Raspberry OS. This small form computer has a 1.4GHz 64-bit quad-core ARM Central Processing Unit (CPU) and 1GB of LDDR2 Random Access Memory (RAM) [25]. It has enough processing power for this application. It is powered by a 5V power supply that can provide up to 2.5A, but it only draws around 500mA when no USB device is plugged in. With the LNA and SDR the total current drawn is at max is 955mA which means the total power draw is approximately 5W while it is idle. At 100% CPU utilization and with both USB devices

plugged in, the total power draw is approximately 7W, which shows that it is a power-efficient system.
In order to keep the CPU cooler a heat sink was placed on the CPU. Even though it is being only passively cooled it is enough to keep the temperature below 60°C, which is way below the maximum supported temperature of 85°C, despite being inside a box with no ventilation. This box is necessary to isolate it from the environment to prevent water from damaging the system. This computer cost 45€ before the chip shortage.

3.3 Software - Hardware Tests and Script

The tests and script were done using the following software modules, all running on a Raspberry Pi. All of them will be described in the following subsections. We had to write a script to call and integrate all those modules.

1. *Gpredict*
2. *GQRX*
3. *noaa-apt decoder*
4. *PREDICT*
5. *rtl_fm*
6. *Sound eXchange (SoX)*
7. *Twitter API*
8. *WXtoImg*

The script, written in Python, was inspired by a code from a GitHub repository of Dr. Paul Brewer: rtlsdr-automated-wxsat-capture [26], which is not maintained for 9 years and it did not work when tested with this system. It is a compilation of software working autonomously. It predicts when the satellite passage will begin and end, using data from the file where the TLE data are stored, it records the received data, demodulates it, and decodes it into an image. To make sure we can collect the most data possible to analyze it later with different software, all the RAW and decoded data is sent to a database. It makes the images public by sending them to Twitter, now known as X. The TLEs are updated every day after a scheduled reboot. A watchdog was also set up to reboot the system if it freezes for some reason.

3.3.1 The first test - *Gpredict*, *GQRX*, and *Noaa-apt Decoder*

To test the hardware setup it was necessary to know when a satellite would pass above and its max elevation in degrees. For that, it was used *Gpredict* [27]. This is a real-time satellite tracking and orbit prediction software that graphically displays the location of each satellite of interest, its observable area on the surface of the Earth, and other data such as the current coordinates in azimuth and elevation. It can also predict future passages providing the time for Acquisition Of Signal (AOS) and Loss Of Signal (LOS) as well as its max elevation during the passage above the station. It also has a built-in feature to control an antenna rotator for automatic tracking. An example of the GUI can be seen in Fig. 3.12

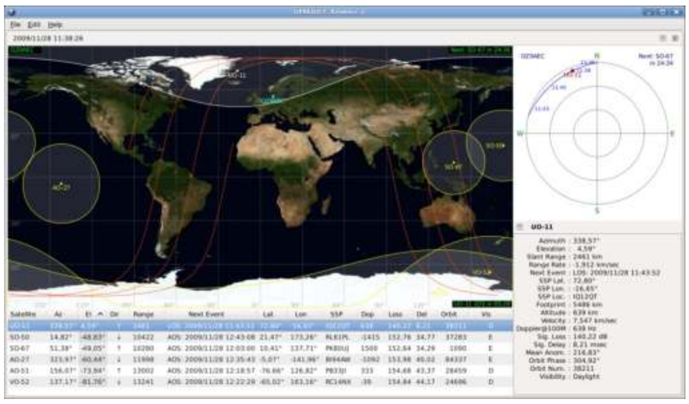


Figure 3.12: *Gpredict* GUI example [27].

Then, after waiting for a passage of a NOAA satellite, *GQRX* was used [28]. This is an open-source software defined radio receiver (SDR) software with a GUI that allows the user to define a frequency, a frequency correction, and a bandwidth to analyze the spectrum, record the received signal, demodulate it, and save it as a WAVEform Audio Format (WAV) file. This can be seen in Fig. 3.13. In our case, it was necessary to adjust the frequency with the frequency correction field. To do so we set the frequency to the transmission frequency of the satellite we were tracking and adjusted that parameter until the peak of the Fast Fourier Transform (FFT) matches the desired frequency.

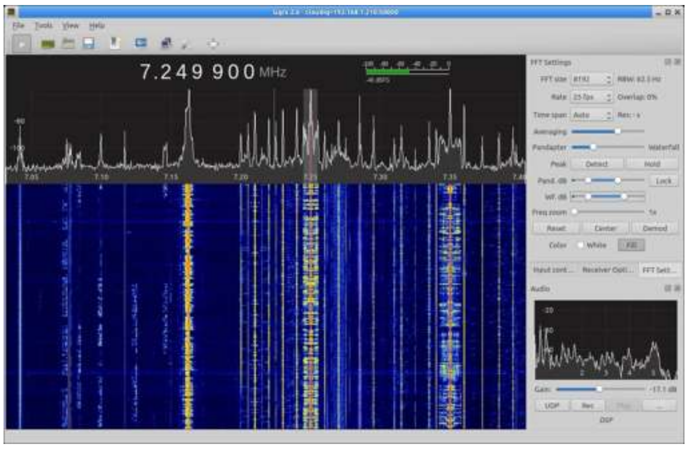


Figure 3.13: *GQRX* GUI example [28].

Having it demodulated, it can now be decoded using *noaa-apt decoder* [29], which is also an open-source software that uses as the input the demodulated WAV file and outputs the image taken by the satellite. It can apply a false color using the brightness of the pixels. However, after testing, it proved it was not accurate. It also can place an overlay of the map on top of the image, but most of the time the map does not match perfectly with the land seen on the image. At the end, a weather image was received and the functionality of the hardware setup was validated. This same image can be seen in Fig. 3.14. This was a process that took some time because we were dependent on the satellite passages at that moment. We noted that most of them did not have enough elevation during the passage and the received signal was too noisy or half of the passages were at night so we only had from 2 to 4 windows of opportunity to test the system during the day.

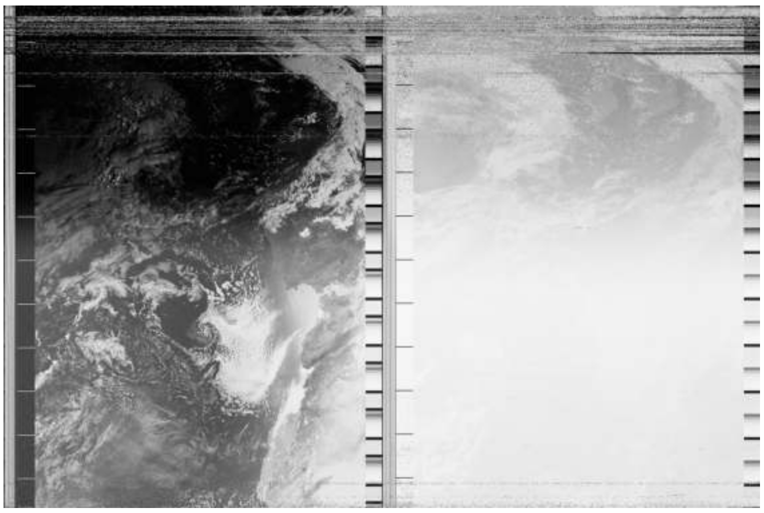


Figure 3.14: First manually acquired image from NOAA 15, 29th of September 2021.

3.3.2 Automation Script

After inspecting the algorithm of the repository of the project rtlsdr-automated-wxsat-capture from Dr. Paul Brewer [26], it was observed that it was using *PREDICT* to anticipate the passage of the NOAA satellites on a secondary script called *pypredict.py*. This is an open-source software for Linux, that tracks satellites and propagates their orbits from a list of TLEs of the desired satellites [30]. Then on the main script, noaacapture.py, he uses *rtl_fm*, from the *rtl_sdr* project, which is a CPU efficient analog demodulator that can record the received signal as a RAW file [31]. It is so light that it can even run on an ARM platform, like the Raspberry Pi. Then it takes that file and inputs it on *Sound eXchange (SoX)*, which is a sound processing software and in this application, it is used to resample, change the bit-rate, and change the file format to WAV [32]. Finally, it uses *aptdec* to decode the image. However, this last software did not work. It is an open-source project to decode apt images that is no longer maintained. The last commit was in 2010. We replaced it with *noaa-apt decoder* and the script worked. It runs the decoder twice to produce one image with a map and another without it. All of these software run on the terminal with no GUI.

The original code did not organize the produced files and it saved them all in the same folder where the script is, so 4 folders were created to organize it by type of file and the output with map and without. Since the microSD card does not have much capacity, a Server Message Block (SMB) server was created in another computer in the same network and the same files were uploaded there with the same organization and then deleted from the Raspberry Pi. In this way, there are no risks of running out of storage.

Later, it was noticed that some images were noisy or with a small portion of an image from the satellite. This happens because the satellite did not pass with enough elevation and the prediction script, *pypredict.py*, did not have this in consideration. Therefore, by analyzing the output of *PREDICT*, we noticed that it outputs several lines with the predicted epoch time, the elevation for that time, and some more information. Therefore, by saving all the predicted elevations for that passage on a list, we could determine the maximum elevation for that satellite. After adjusting the minimum elevation allowed to record, it was decided to set it at 30º. To be sure that the script was functioning while testing a user interface was implemented on the terminal that shows when is the next passage and a countdown for when it will begin recording the signal. Then, to make these images publicly available it was decided to post them on Twitter using the Twitter Application Programming Interface (API) for Python, to make tweets with an image attached. While looking at the images on Twitter we noticed that some were reversed. This happens because the orbits of the NOAA satellites are polar which means that sometimes they will pass above the station from North to South or from South to North. It appears that the images that are reversed, are the ones from the passages from South to North. To solve this issue, we modified *pypredict.py* again to use the output of *PREDICT* to track the changes in the latitude for each passage. If the latitude increases on each line it means that it is going from South to North, otherwise, it is from North to South. Now that it is possible to determine the direction of the passage, it is just a matter of rotating the image 180º, when necessary, using the Python Imaging Library (PIL) library. The total processing time, from acquisition of the signal to posting the image on Twitter is from 15 to 20 minutes. This process is best visualized in Fig. 3.15.

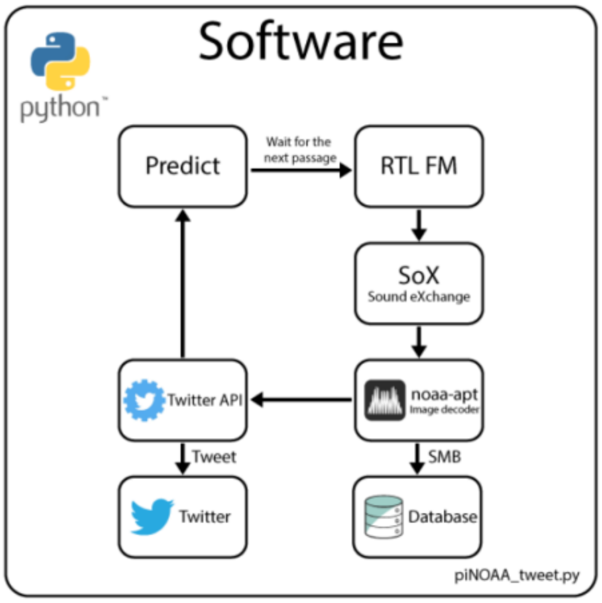


Figure 3.15: Block diagram for the automation script.

Some time later, it was decided to take this weather station to Maputo, Mozambique as an educational project with the prospect of delivering weather information to remote areas with the intent to help, for example, in agriculture to better manage the usage of water. At the moment, it is mounted on the rooftop of Oswela, a Non Governmental Organization (NGO). The location is not ideal because it is surrounded by tall buildings on the horizon and many houses use electrical fences, that can produce RF noises which can interfere with the received signals. It can be seen on the piNOAA's Twitter page that the quality of the images is not the same as when it was mounted on the rooftop of the Instituto de Telecomunicações of Aveiro.

3.3.3 ***WXtoImg***

After the system was tested and mounted in Mozambique, we tried to do some analysis of the images to try to get some more weather information. This proved to be impossible by using *noaa-apt decoder* because it could not differentiate the water from the land or the clouds from the land sometimes, and upon further reading of their website documentation [29], we read that *WXtoImg* provides better results when applying false colors. This software is also an APT decoder, that stopped being maintained in 2018 when the website went offline. It was later restored by the community and made available for everybody, even the professional version [33]. This software proved to be superior to *noaa-apt decoder* used in the script. It can do recording, decoding, live decoding, advanced color enhancements, multi-pass images, and more.
After reading the documentation [34], we decided to try the Multispectral Analysis - Precipitation, which determines which regions are most likely to be clouds, land, or sea based on an analysis of the images of both channels. The high cloud tops are colored to give an approximate indication of the probability and intensity of precipitation. We also tried the IR contrast enhancement which increases the contrast in the darker land/sea regions and colors the cold cloud tops. It provides a very readable indication of cloud top temperatures. The results were satisfactory.
To note that this was done on a laptop with the latest i7 CPU running Ubuntu since we no longer have access to the Raspberry Pi of the weather station.

3.4 Experimental Results

In this section, we show some examples of received images at different times of the day and, in the end, images decoded using *WXtoImg*. In Fig. 3.16 and in Fig. 3.17 it can be seen the used sensors at different times of the day, being possible to see the switch of one sensor from day to night mode, and from night to day mode. It also shows that a minimum of 30° in the maximum elevation is enough for good reception of images. This was only valid when it was mounted on the roof of IT Aveiro. When mounted in Maputo, on the roof of Oswela, even a passage with a maximum elevation of 85° cannot produce an image with the same resolution as the images produced in Aveiro. This can be seen in Fig. 3.18. Finally, Fig. 3.19 and Fig. 3.20 were produced using *WXtoImg*. With this type of analysis from this software, it is possible to determine the temperature of the top of the clouds, to determine the probability and intensity of precipitation.

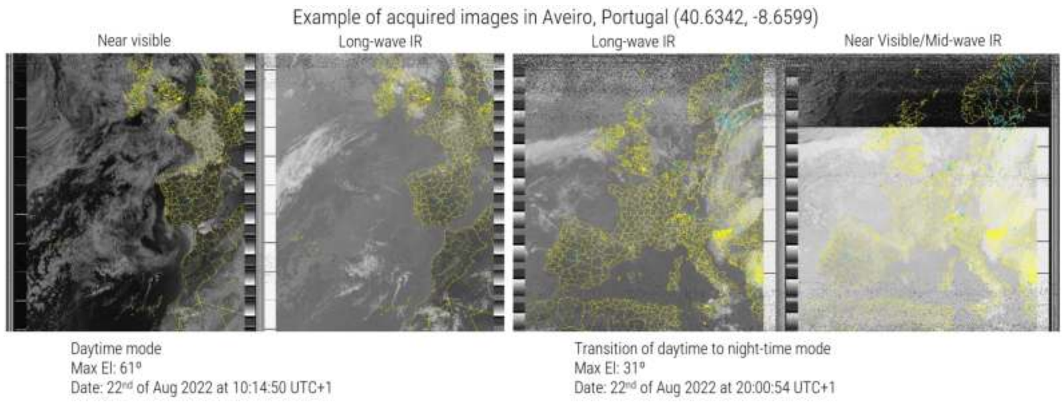

Example of acquired images in Aveiro, Portugal (40.6342, -8.6599)
Near visible
Long-wave IR
Long-wave IR
Near Visible/Mid-wave IR
Daytime mode
Max El: 61°
Date: 22nd of Aug 2022 at 10:14:50 UTC+1
Transition of daytime to night-time mode
Max El: 31°
Date: 22nd of Aug 2022 at 20:00:54 UTC+1

Figure 3.16: Example 1 of acquired images in Aveiro, Portugal (40.6342, -8.6599).

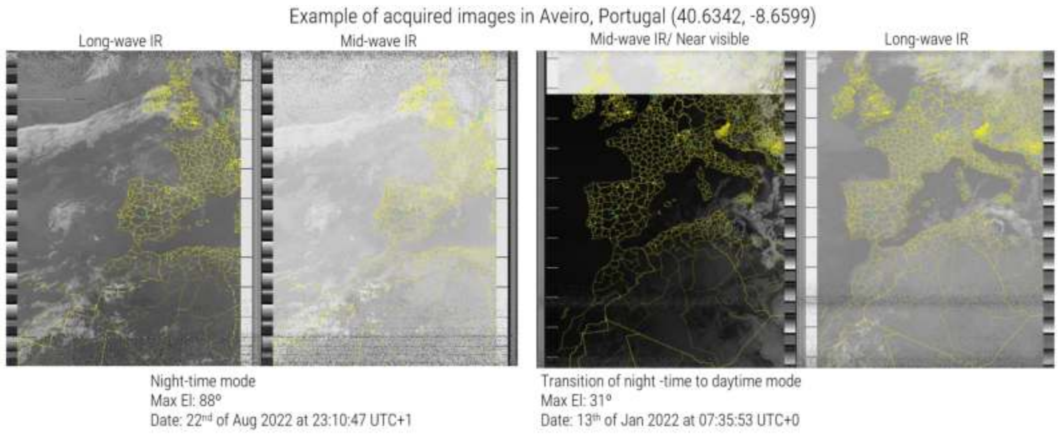

Example of acquired images in Aveiro, Portugal (40.6342, -8.6599)
Long-wave IR
Mid-wave IR
Mid-wave IR/ Near visible
Long-wave IR
Night-time mode
Max El: 88°
Date: 22nd of Aug 2022 at 23:10:47 UTC+1
Transition of night -time to daytime mode
Max El: 31°
Date: 13th of Jan 2022 at 07:35:53 UTC+0

Figure 3.17: Example 2 of acquired images in Aveiro, Portugal (40.6342, -8.6599).

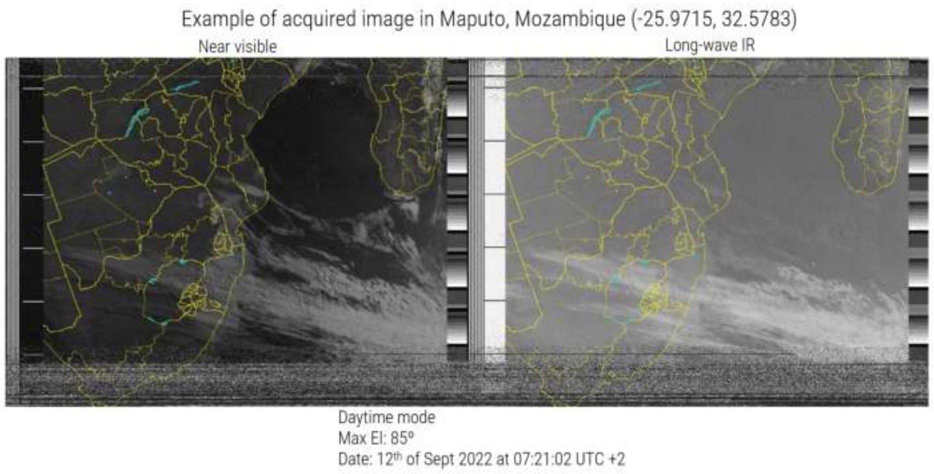

Example of acquired image in Maputo, Mozambique (-25.9715, 32.5783)
Near visible
Long-wave IR
Daytime mode
Max El: 85º
Date: 12th of Sept 2022 at 07:21:02 UTC +2

Figure 3.18: Example of acquired image in Maputo, Mozambique (-25.9715, 32.5783).

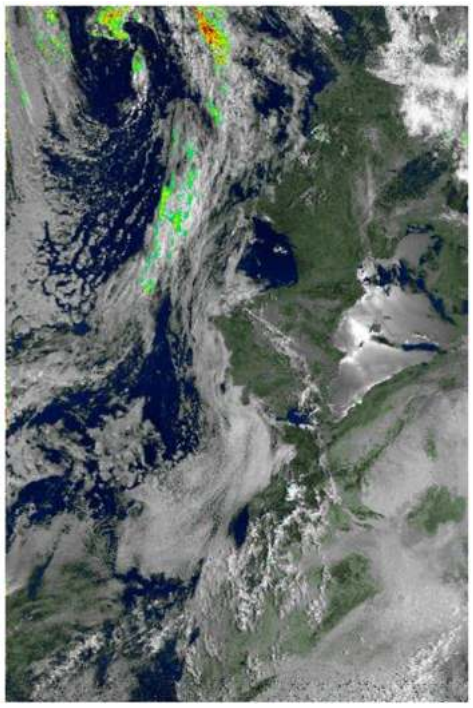


Figure 3.19: Example of an acquired image in Aveiro with a Multispectral Analysis - Precipitation, Portugal (40.6342, -8.6599).

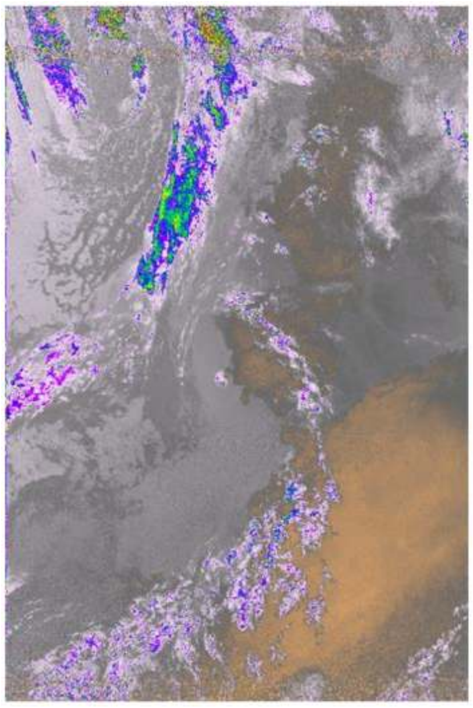


Figure 3.20: Example of an acquired image in Aveiro with IR contrast enhancement, Portugal (40.6342, -8.6599).

3.5 Summary

In the context of this dissertation's proposed work, where the aim is to develop a low-cost earth image acquisition, obtained from satellites of meteorological analysis. A ground station was developed, dedicated to the reception of images provided by the NOAA satellites containing data/information acquired above the Earth. It consisted of the implementation of a handmade antenna, LNA, SDR, a set of free software tools to process the data, storage on a remote server, and usage of a API to publish the images on a social network. The system was installed on the roof of Instituto de Telecomunicações with a dedicated internet connection and power supply.

The work is best described in Fig. 3.2 and Fig. 3.15

On the hardware diagram, it is possible to see how the antenna, LNA, SDR, and the Raspberry Pi are connected and the software diagram shows the flow of the script. It starts by predicting the next visible passage that fulfills the minimum degrees of elevation, processes it, uploads it to a server, and sends the image to Twitter using the Twitter API.

The main challenges were the satellite visibility/availability from where the ground station was installed. To have a better quality image reception from the satellite transmitter, it was defined to only acquire the data from the NOAA satellites at elevations higher or equal to 30°.

Regarding the antenna, it consists of a helicoidal antenna tuned for a central frequency of 137.5 MHz prepared to receive with circular polarization, to be in accordance with the transmitted signal from the NOAA satellites. From the possible options, this was chosen because of its radiation pattern, since the passages do not have the same trajectory for each satellite and each different passage of the same satellite. A LNA was used to amplify the signal received by the antenna by a minimum of 30dB and to guarantee a maximum signal excursion at the SDR.

The signal acquisition is performed by a SDR which is based on the RTL2832U from Realtek. The process consists of saving the raw data received in a file. After this, the file is processed, converted into an audio file, representing an image, and then decoded into an image using a APT decoder. All of it processed by a Raspberry Pi 3 Model B+

The NOAA satellites TLEs are obtained from a public site [35] and propagated by the *PREDICT* tool for 1 day. After this, the software starts the procedure of acquisition based on the elevation of the satellite and at the time previewed by *PREDICT*.

noaa-apt decoder is responsible for the decoding of the audio file into an image and it overlays a map on top of it.

WXtoImg is another decoder, that had a professional license, but it is no longer maintained. However, it proved to be more powerful than *noaa-apt decoder*, which is open source. It can crop automatically the noise from the beginning and end of the capture where the elevation is low and as a consequence, the quality of the signal is poor. Furthermore, it has a better set of tools to analyze the image, for example, Multispectral Analysis - Precipitation. By combining the images from each channel it can colorize the tops of the clouds which indicates the likelihood of rain in that area.

Twitter API allows us to upload the images to a public space where they can be accessed by the community.

After some months passed, by analyzing the images on Twitter, we noticed that the accuracy of the drawn map on the image was reducing over time, and then later, the noise from the AOS time disappeared and the noise of LOS time increased. When looking at the TLE that *PREDICT* was using, we noticed that they were outdated. This happened because we forgot to give executable permission to that script, and the system was running like this for at least 3 or 4 months. It shows that the effects of an outdated TLE are not immediately noticed, but they increase considerably over time.

The piNOAA ground station performs an Earth Observation service dedicated to meteorological imaging reception automatically. It can be easily installed anywhere, requiring only an internet connection and electricity. The entire system is made of Commercial Off-The-Shelf (COTS) materials. The software tools are open source and are available [36].

To provide better-quality images, the system needs to change the antenna to a directional type with higher gain and sensitivity. It has to be tuned for the L band with a tracking system to capture HRPT, instead of APT. Despite this, the system is very robust and fully operational. The prototype price is 154.5€ as shown in Table 3.3.

Table 3.3: Price table of piNOAA.

Component	Price in €
QFH Antenna	30
Nooelec NESDR Mini 2 USB RTL-SDR	29
BLK-18-S+	5,5
Nooelec SAWbird+ NOAA	45
Raspberry Pi 3 Model B+	45
Total	154,5

Chapter 4

EIS: Eye In The Sky

4.1 Project Requirements

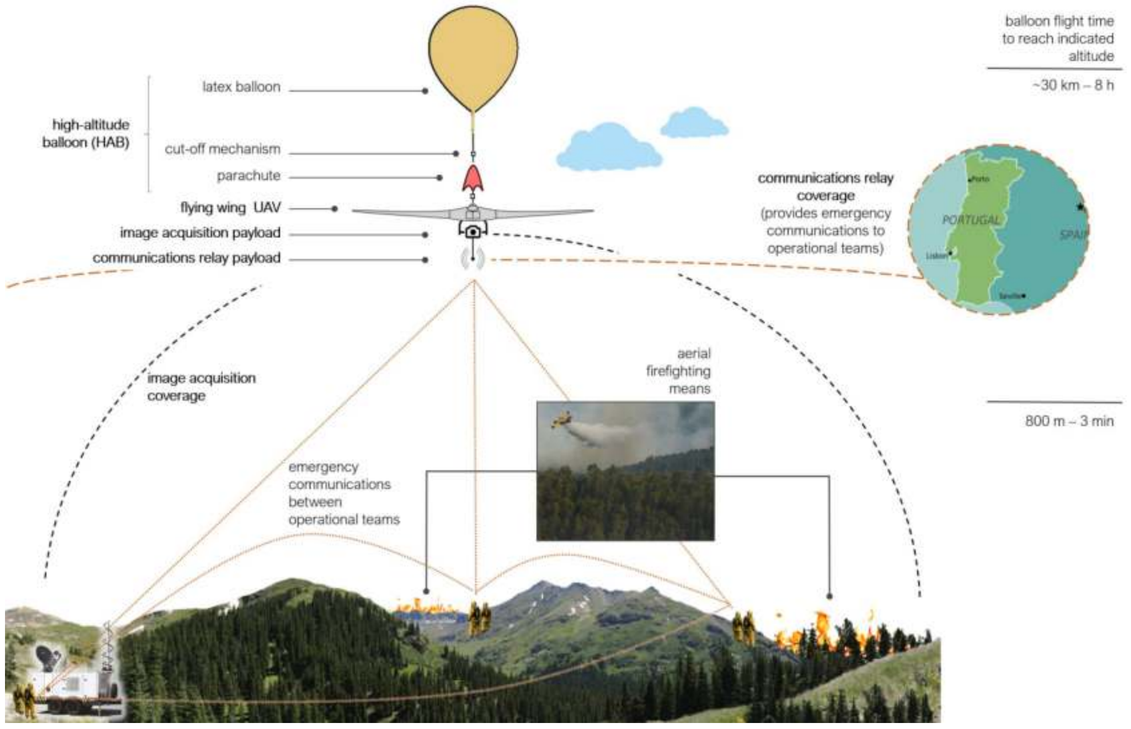

latex balloon
high-altitude balloon (HAB)
cut-off mechanism
parachute
flying wing UAV
image acquisition payload
communications relay payload
balloon flight time to reach indicated altitude
~30 km – 8 h
communications relay coverage
(provides emergency communications to operational teams)
PORTUGAL
image acquisition coverage
aerial firefighting means
800 m – 3 min
emergency communications between operational teams

Figure 4.1: Mock-up of the EIS project.

In the framework of Eye In the Sky [37], a Fundação para a Ciência e a Tecnologia (FCT) sponsored project, it was necessary to develop a low-cost payload that will go on board of a HAB or a UAV. It should be able to take pictures and send them using the Amateur X.25 (AX.25) protocol when requested. It should also send telemetry data, in order to know where the pictures were exactly taken. A low-cost ground station also needs to be developed, to send the requests for a picture and to receive them. Since the payload is on a HAB or a UAV, it should be light and power-efficient, to be able to fulfill its mission when deployed. A mock-up of this project can be seen in Fig. 4.1. Some of the proposed solutions for communication devices were the usage of radios, Terrestrial Trunked Radio (TETRA) (i.e. Kenwood Th-D74E), and SDRs to send and receive the images, LoRa for telemetry or another radio to transmit Automatic Packet Reporting System (APRS) and a Sistema Integrado de Redes de Emergência e Segurança de Portugal (SIRESP) repeater for voice communication.

We designed two solutions. One based on TETRA devices (Kenwood Th-D74E), and another based on SDR for data transmission and LoRa for telemetry.

4.2 Version 1 - Kenwood TH-D74E

4.2.1 Hardware

The setup is made of the following components, powered by a Raspberry Pi. We will describe each of them in the following subsections. Fig. 4.2 represents the payload.

Payload:

1. Radio - Kenwood Th-D74E
2. Quarter Wave Antenna
3. Raspberry Pi 4 Model B 8GB
4. Raspberry Pi High-Quality camera
5. Xiaomi Mi 50W Power Bank 20000mAh
6. USB C Power Delivery Board
7. Buck converter

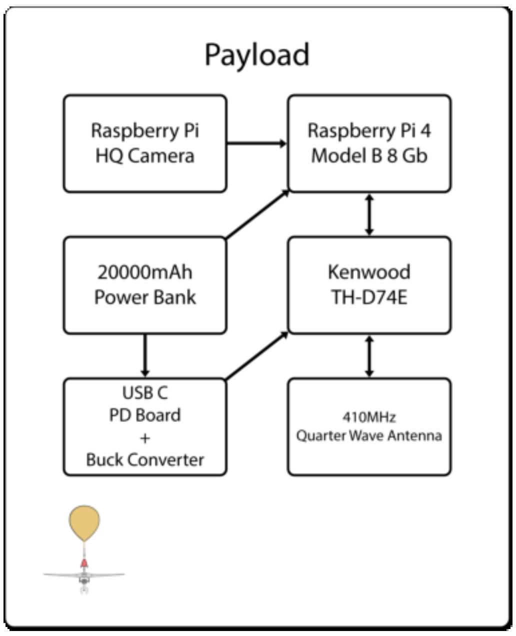

Payload
Raspberry Pi HQ Camera
Raspberry Pi 4 Model B 8 Gb
20000mAh Power Bank
Kenwood TH-D74E
USB C PD Board + Buck Converter
410MHz Quarter Wave Antenna

Figure 4.2: Payload block diagram.

Ground station:

1. Laptop
2. Radio - Kenwood Th-D74E
3. Antenna

Radio - Kenwood TH-D74E

Figure 4.3: Front view of the radio TH-D74E.

For this first version, it was decided to use the Kenwood TH-D74E, as seen in Fig. 4.3. It is a handheld radio that offers a lot of functionalities in a small form factor, such as APRS, D-STAR which is a digital voice and data protocol used in amateur radio, AX.25 via a Terminal Node Controller (TNC) using Keep It Simple Stupid (KISS), which is a protocol derived from X.25 used in packet communication by radio amateurs and GPS. It can operate in the VHF and Ultra High Frequency (UHF) bands and transmit with four power modes: 0.05W (EL - Extremely Low), 0.5W (L - Low), 2W (M - Medium), and 5W(H - High). It has a battery of 1800 mAh [38]. It weighs 345g with the battery and it is connected via USB to a Raspberry Pi 4 Model B 8GB. After some tests it was concluded that the battery did not last enough time for the duration of the flight and a solution was found by making a dummy battery connected, as seen in Fig. 4.4, to a buck converter calibrated for 7.4V. This buck converter needed to have its lm2596 replaced because it was heating up more than it should for the current that was being drawn. Upon further inspection, we noticed that the switching frequency was 50 KHz instead of the supposed 150 KHz proving that it had a fake lm2596. It is then connected to a generic 12V USB C PD module, at the end powered by a 20000 mAh power bank. The buck converter and USB-C PD modules can be seen in Fig. 4.5. All of these components, except for the power bank, are secured in place by a 3D printed support we designed using

FreeCAD and printed using 3D printers provided by IT. It is fixed with Velcro to the inside of the payload box. The 3D model can be seen in Fig. 4.6. This was necessary to organize the placement of the components inside the payload, to prevent damage when the payload falls to the ground at the end of the mission.

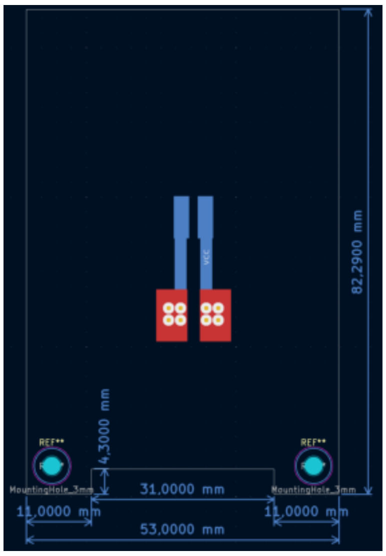


Figure 4.4: Dummy battery layout.

Figure 4.5: LM2596 module and USB C PD module

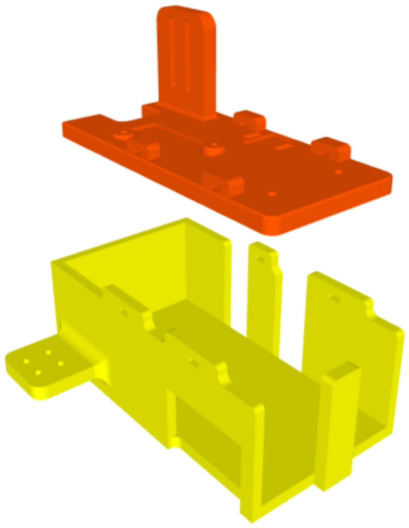


Figure 4.6: 3D model of the radio support.

Quarter Wave Antenna

To avoid damaging the original antenna of the radio, from the fall of the payload during the tests, a quarter wave antenna, which is also an omnidirectional antenna, was constructed. As seen in Fig. 4.7. It was made with thick copper wires from a power cable and designed to operate at 410 MHz. The radiating element length needs to be a quarter of the wavelength of the signal. In this case, at 410 MHz the wavelength is 73.12 cm, so the length of the radiating element needs to be 18.28 cm. The radials should be 12% longer (20.47 cm) and have a 45^{o} angle, so the impedance is around 50Ω. As seen in Fig. 4.8 the gain of this antenna is 10dB.

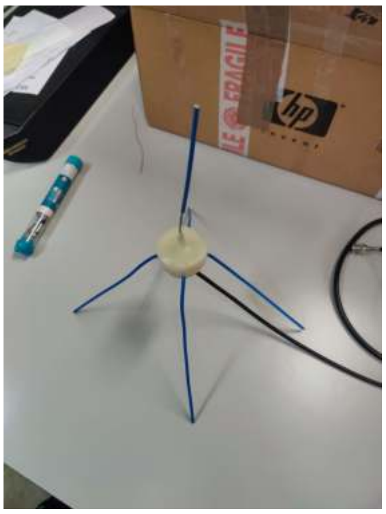


Figure 4.7: Built quarter wave antenna.

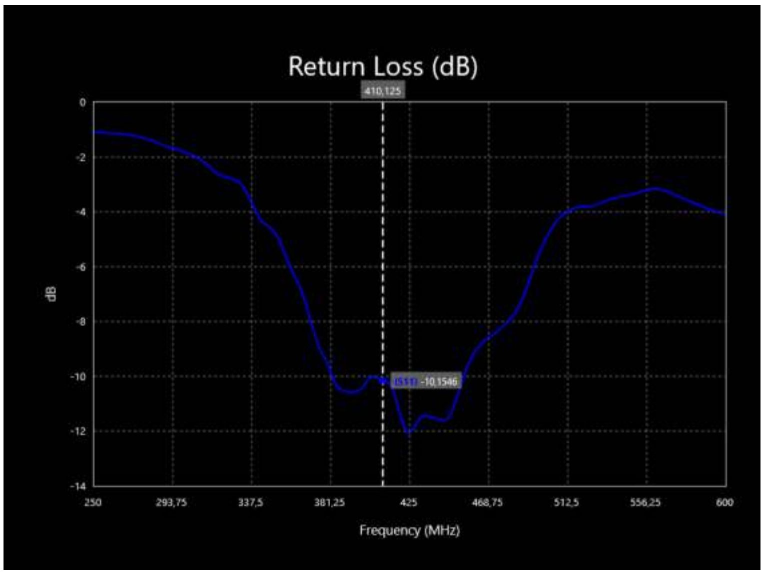


Figure 4.8: S-Parameter, s11, of the built quarter wave antenna.

Raspberry Pi 4 Model B 8GB and Camera

Figure 4.9: Raspberry Pi 4 Model B.

To take pictures, compress them, process the received messages requesting pictures or Input/Output (IO) commands to control the UAV, and send the taken pictures it was decided to use a Raspberry Pi 4 Model B, shown in Fig. 4.9, running the latest version of Raspberry Operating System (OS). This small form computer has a 1.8 GHz 64-bit quad-core ARM CPU, overclocked to 2.0 GHz, and 8GB of LPDDR4-3200 SDRAM [39]. Is powered via USB C by a 20000 mAh power bank that supports USB PD (Power Delivery) and can provide up

to 50W (20V- 2.5A) but in this application it can only provide 15W (5V - 3A), which is the minimum required for normal operation. It only draws around 500mA at idle when no USB device that requires power from it is plugged in. At 100% CPU utilization it consumes 6.4W, which shows that it is a power-efficient system.
In order to keep the CPU cooler a heat sink was placed on the CPU. In this application, there was the concern of the CPU reaching a temperature below $0^{\underline{o}}$ because of the conditions of the atmosphere when conducting the tests. However, it was proved in tests in a Temperature Humidity Chamber that, by having the CPU overclocked and running the script, it would keep itself warm enough to operate in such conditions. The payload box is made of Styrofoam and filled with foam which also contributed to keeping the internal temperature and isolating it from the outside atmospheric conditions. Paired with it is a Raspberry Pi High Quality Camera 12.6 [40] Mega Pixel (MP) sensor with a 6mm lens, mounted on the back of the board. The radio is connected via USB. This part of the setup is mounted on a 3D printed support, designed by us and printed at IT, and fixed to the inside of the payload with Velcro. The 3D model can be seen in Fig. 4.10.

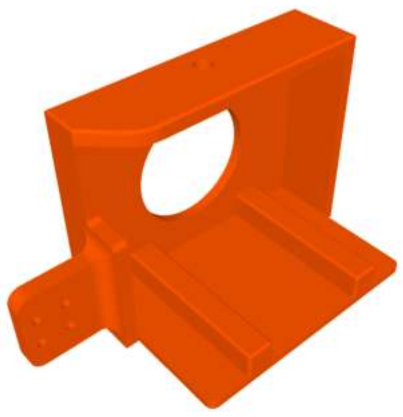


Figure 4.10: 3D model of the Raspberry Pi 4 and camera support.

Xiaomi Mi 50W Power Bank 20000mAh

Figure 4.11: Xiaomi Mi 50W Power Bank 20000mAh.

To power everything the 20000mAh Xiaomi Mi 50W Power Bank, shown in Fig. 4.11, was chosen. This is because it has USB PD on the USB C port, being capable of providing 5V-3A/9V-3A/10V-5A/12V-3A/15V-3A/20V-2A and two other USB A ports capable of 5V-3A and more than enough capacity to power everything inside the payload for the whole flight duration. It was used to power the Raspberry Pi 4 and the Kenwood TH-D74E radio. This part is held by a 3D printed support, designed by us and printed at IT, fixed to the inside of the payload with Velcro. The 3D model can be seen in Fig. 4.12.

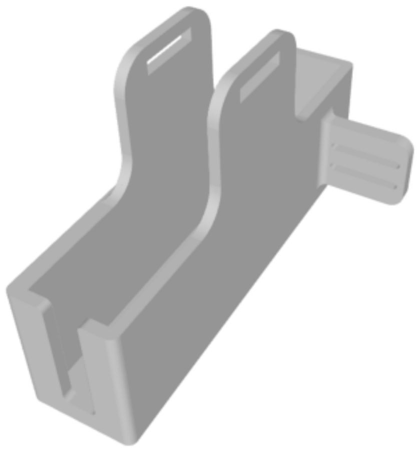


Figure 4.12: 3D model of the power bank support.

Ground Station

A ground station is a terrestrial radio station designed for telecommunication with, typically, a spacecraft or reception of radio waves from astronomical radio sources. In our case, it will be to communicate with a payload being carried by a UAV or a HAB. It consisted of a laptop connected to an unmodded Kenwood TH-D74E with a UHF antenna mounted on the roof of a car with magnetic support. This time the battery of the radio was enough for the duration of the whole operation because it was only receiving data.

4.2.2 Software

The tests and script were done using the following software and protocols, all running on a Raspberry Pi. We will describe them all in the following subsections. We had to write a script to call and integrate all those modules.

1. AX.25
2. libcamera-still
3. ffmpeg
4. PIL python library
5. GPS - National Marine Electronics Association (NMEA)
6. Exif python library
7. Universal asynchronous receiver/transmitter (UART) and IO commands

Software and Field Tests

The first version of AX.25 was a data link layer developed in 1982, based on layer 2 of the old X.25 protocol, to transmit data between two nodes via radio. It can be associated with an Internet Protocol (IP) address. This is widely used by radio amateurs in packet communication. It is used with a TNC, which interfaces between a terminal and a radio transceiver. The data is formatted into AX.25 packets and modulated into audio signals to be transmitted by the radio. KISS is the communication protocol used between the terminal and the TNC. It is designed to easily be implemented in simple embedded devices, capable of asynchronous serial communications. It allows arbitrary data to be transferred, but there is no error handling. While its packets are arbitrary it commonly uses AX.25 or Transmission Control Protocol (TCP) protocols.

In order to use the AX.25 protocol with Python a library was installed from a GitHub repository [41]. However, before being able to use this type of communication some configurations need to be done.

```
$ sudo /usr/bin/killall kissattach
$ sudo /usr/sbin/kissattach /dev/ttyACM0 radio 44.136.8.5 —m 512
$ sudo /sbin/route add —net 44.136.8.0 netmask 255.255.255.0 ax0


$ sudo /sbin/ifconfig ax0 44.136.8.5 netmask 255.255.255.0 up
$ sudo /sbin/ifconfig ax0 broadcast 44.136.8.255 mtu 512
$ sudo /sbin/route add —net 44.136.8.0 netmask 255.255.255.0 ax0
```

This adds a new network interface, to be used by the scripts to communicate using the AX.25 protocol [42]. Sometimes, if the connection is interrupted, when reconnecting the radio it would be associated with/dev/ttyACM1 and the script would fail. In this situation a reboot of the computer and turning the radio off and on would solve the issue.

After this step was done, we could send simple text messages successfully from one computer to another using the radios, with the included examples in the repository. Afterward, those example scripts were modified to open a file, a small grey-scale image of a fire, and send it as text. The transmission of the file was quite slow, but this was because the size of the transmitted packets was small and the file was too big, around 175KB. Thus, FFmpeg was used to compress the image, in Fig. 4.13 and turn it into a H.264 format, which reduced the file size significantly.

```
$ ffmpeg —i IN.tiff —c:v libx264 —crf 25 OUT.h264
```

By adjusting the crf parameter, we can adjust the quality of the output with 0 being lossless and 51 with the most losses. With the previous command, an image with 176KB was reduced to a file with 1.2 KB.

With the following command, the image is converted back to TIFF file format. The output file size is around 62KB because of the losses from the compression. This image can be seen in Fig. 4.14.

```
$ ffmpeg —i OUT.h264 —crf 0 —pix_fmt gray imgOUT.tiff
```

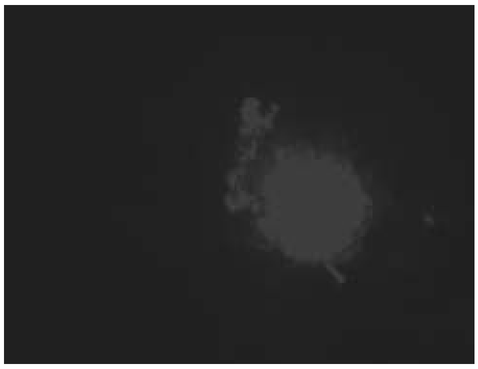


Figure 4.13: Example of an uncompressed image used for testing.

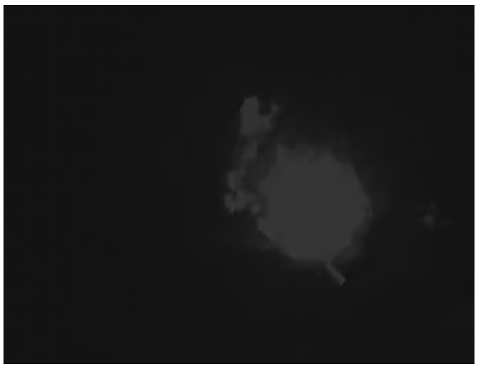


Figure 4.14: Example of a processed image used for testing.

As we increased the size of the packets and by comparing the transmitted and received data, we noticed that if the packet size was too long, the message sometimes arrived incomplete. Thus, we tuned it to a size that did not lose any information. For the next iteration, it would send images periodically. It would list all the files in a directory, order them, put the file names on a list, compress them using FFmpeg, which is a command-line tool that converts audio or video formats, and send them one by one. The receiver was listening and decompressing the receiving files with FFmpeg as well. During this test, some of the images arrived corrupted and it was because of what happened before in the first test. So the packet size was tuned again so the files would be received whole. We settled on a data field size of 250 bytes. By measuring the time of some transfers, we determined that it has an average transfer speed of just 192 bytes per second. Having accomplished the automatic transmission of the files in a folder to prevent the corruption of the AX.25 packets were encapsulated in a TCP packet, but because of the TCP protocol overhead, it increased the already low transfer speed. During our tests, we could also connect to the other computer terminal via Secure Shell (SSH), using the radios as network adapters. It was also slow, but it showed that we could connect to a computer some kilometers away and off the internet. This test prompted us to try an File Transfer Protocol (FTP) connection, however, since this protocol relies on TCP to ensure the files arrive intact it was also really slow. But, again, it showed that is also possible to safely transfer files from one computer to another that are far away from each other and off the internet.

Even though it was slow, for the type of images that were provided to us (Fig. 4.13), it was decided to test the distance the radios could communicate in the various power modes and to test the quarter wave antenna. To do so, one radio was placed on the roof of IT-1 and the other on top of Barra lighthouse, a distance of roughly 7.5 km, and all the tests were conducted successfully except the weakest power option of 0.05W which was to be expected even by having a line of sight between the two points. Fig. 4.15 shows the location of the payload and of the ground station as well as the distance between them.

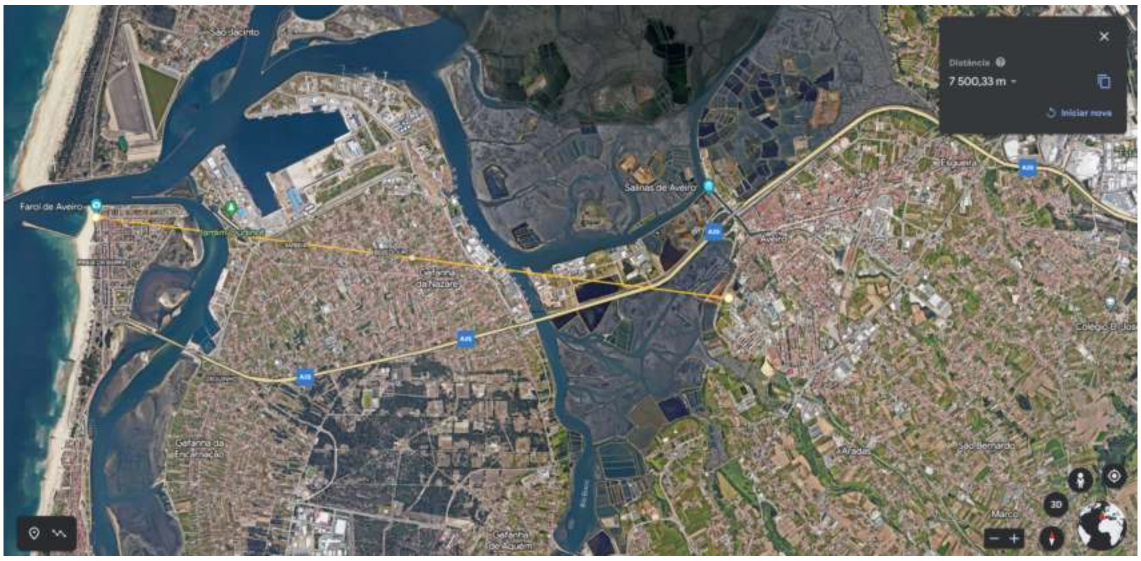
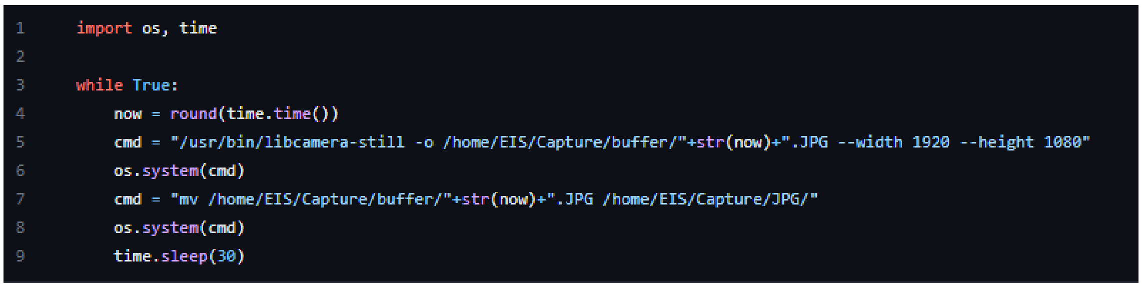


Figure 4.15: Distance between IT-1 and Barra's Lighthouse.

After this test was completed, the next step was to have a camera taking pictures periodically. It was decided to use a Raspberry Pi High-Quality camera because of its compact size, the software support, and it can be mounted behind the computer. To take a picture it is only needed one command with the resolution of the image specified. A simple Python script was developed to take a picture with a resolution of 1920x1080 every 30 seconds.



```python
import os, time

while True:
    now = round(time.time())
    cmd = "/usr/bin/libcamera-still -o /home/EIS/Capture/buffer/"+str(now)+".JPG --width 1920 --height 1080"
    os.system(cmd)
    cmd = "mv /home/EIS/Capture/buffer/"+str(now)+".JPG /home/EIS/Capture/JPG/"
    os.system(cmd)
    time.sleep(30)
```

For this script and radio configuration to run on boot, a cronjob was created. This can be done in the following way:

```
$ crontab -e
@reboot sh /Path/To/Bash/Script.sh
@reboot python3 /Path/To/Python/Script.py
```

Now to automate the Ground Station and the UAV/HAB, two scripts were developed. Because the radios are half-duplex and to prevent two processes from trying to access the same resource at the same time, the scripts have to have some sort of synchronization techniques.

The Ground Station is running both processes permanently and when one is running the other is waiting in a loop for a flag change controlled by the other process, like a mutex, which is a mutual exclusion lock. Only one process can hold the lock, in this case, the radio. In the beginning, it sends a command "TakePhoto" and changes a flag signaling the other process to start running and to wait for a response from the UAV. It stays stuck in a loop waiting for the message to be received or for a timeout to occur. It is also able to send other commands to change the output of some General Purpose Input/Output (GPIO) pins from the Raspberry Pi. To test this feature we turn on/off some Light Emitting Diode (LED)s to simulate the opening or closing of an air valve for the balloon to be used to carry the payload in tests, but it was never used at the end. It also could send some commands to the Raspberry Pi UART, to send commands to a possible micro-controller onboard.

The UAV script is similar to the Ground Station, but since it is running on a Raspberry Pi, in order to save some resources, the control of the processes is different. There is the main process that is always waiting to receive the commands and if it receives a "TakePhoto", it launches another process and waits for the new process to be terminated. This script also runs at boot in a cronjob.

While testing the communication between the two computers, the transmission took longer than before, even if the original file was smaller than the previous .tiff images. This is to be expected because now it is compressing a Red Green Blue (RGB) image instead of a grayscale one. On average the images were four times larger, so it took four times more to transmit them. Some issues were found during testing. Some images were arriving corrupted and if we let the test run long enough, the Kenwood TH-D74E on the payload would freeze and it could only recover by powering it off and on again. This is because the script was sending, via serial communication, the commands too fast and it would fill up the buffer of the radio. To solve this problem a small delay is introduced between packets, to allow the previous packet to be sent and the buffer to be cleared. By doing this it was possible to increase the size of the sent packets.

Being the previous step validated, the next step was to assemble the payload and test if it could withstand the cold temperatures of the troposphere and stratosphere since the maximum altitude a HAB can reach is around 40 km. To do so the payload was inside a Temperature and Humidity Chamber, as seen in Fig. 4.16, set to -40°C while it was taking photos and transmitting them to a Personal Computer (PC) outside the chamber. The internal temperature could not be recorded but the temperature of the CPU was written in the file name of the image. The lowest temperature the chamber could reach was -38°C.

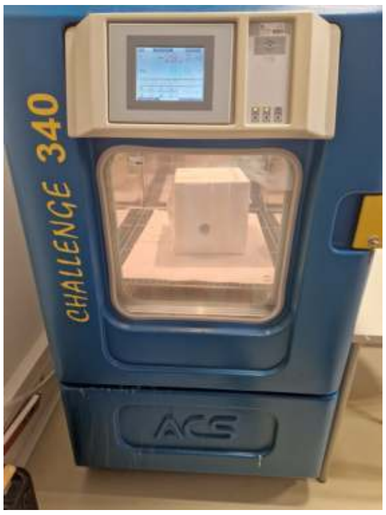


Figure 4.16: Sub zero temperature test.

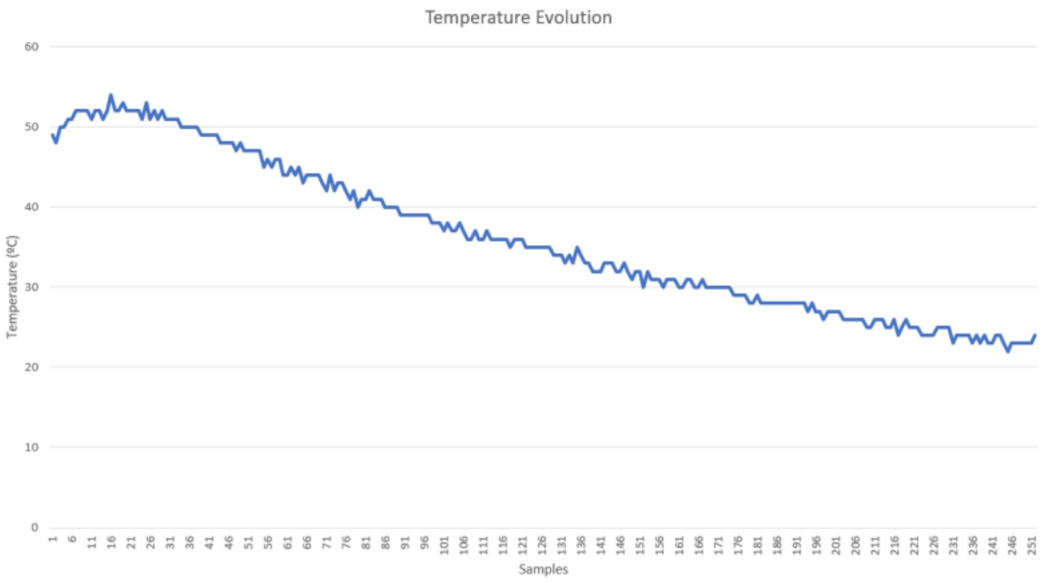


Figure 4.17: CPU temperature evolution.

As we can see from the graph above, in Fig. 4.17, when the temperature outside of the payload was around -38°C the CPU never dropped below 20°C, way above the 0°C limit imposed by the manufacturer. This is possible because the Styrofoam box isolates the inside from the outside conditions and keeps some of the produced heat inside from all the electronics functioning, mainly the CPU of the Raspberry Pi and the radio.

Following this temperature test, we had the green light to do a test with the payload on a HAB. Note that at this point, we were relying on the radio battery to power itself and that it only took pictures when requested, instead of periodically. The launch site was in Nazaré and while we were preparing the payload to make sure it was transmitting images we noticed that it was taking longer than usual to transmit a single image and we took longer to launch the HAB. The preparation for the launch can be seen in Fig. 4.18 and the pre-launch transmission test image can be seen in Fig. 4.19. Because of that, we did not have enough battery for the whole flight and we only managed to get two images in the air. The images can be seen in Fig. 4.20.

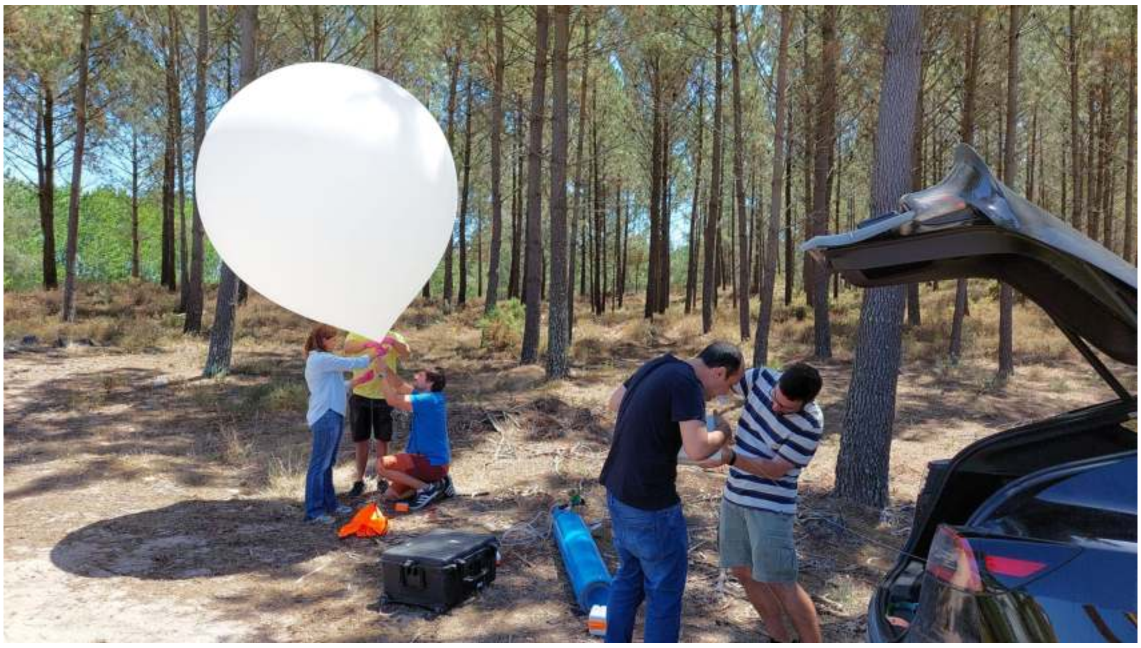


Figure 4.18: Launch preparation in Nazaré.

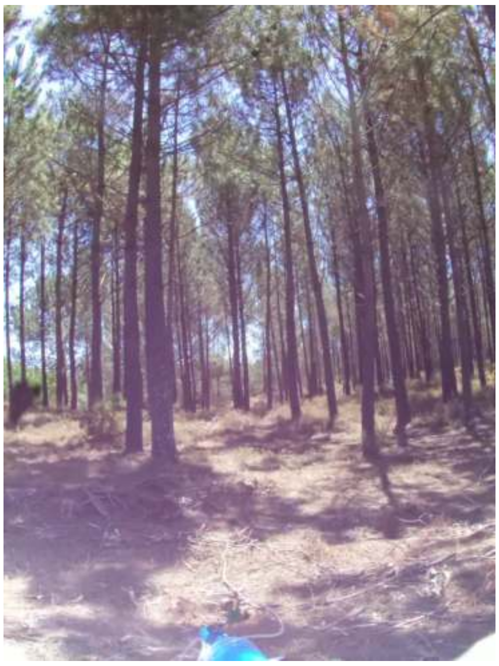


Figure 4.19: Pre-launch transmission test.

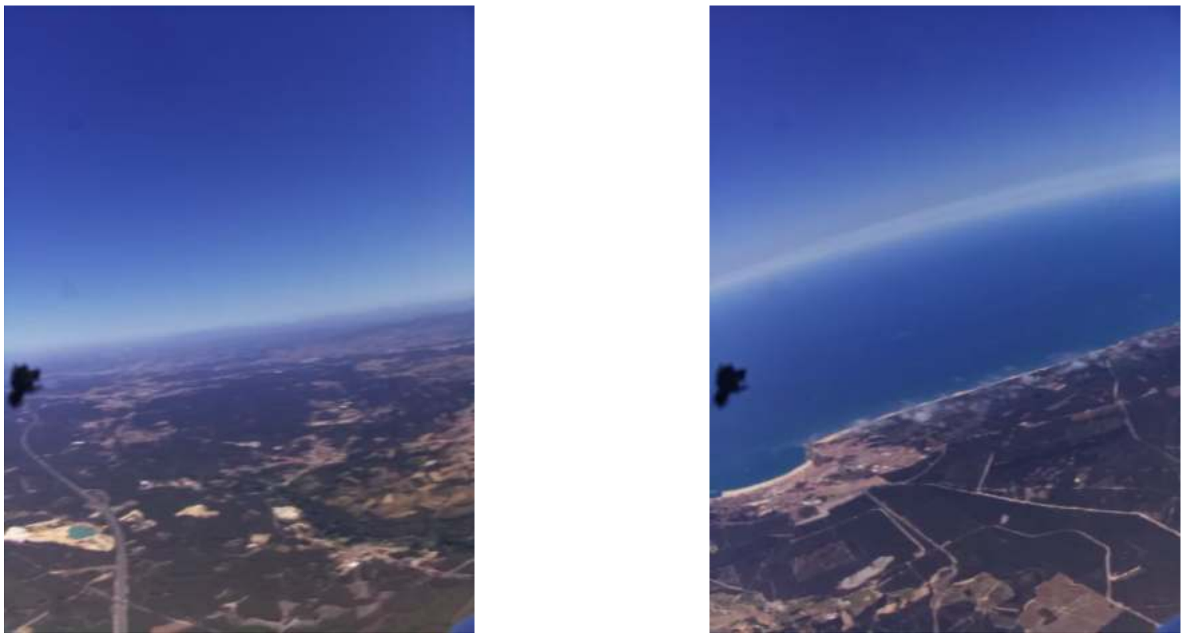


Figure 4.20: Received images from the weather balloon.

After analyzing the received images, we noticed that the compressed file was 68.3 KB which is almost 16 times bigger than what was tested in the lab. This makes sense because in the lab all the colors were similar, like the wall and the desk, and some minor details like some black cables from a keyboard. The one in the test pre-launch is full of details of the trees, the ground, and the sky. There are too many changes of colors and because of that it cannot compress it as much as in the lab.

After replacing the battery of the radio for the dummy battery with the USB C PD module connected to a Buck Converter module, powered by the power bank and modifying the script to take photos every 30 seconds, sending the most recent photo, we went to do another field test with a HAB, this time in Coruche. The final payload, of this version, can be seen in Fig. 4.21.

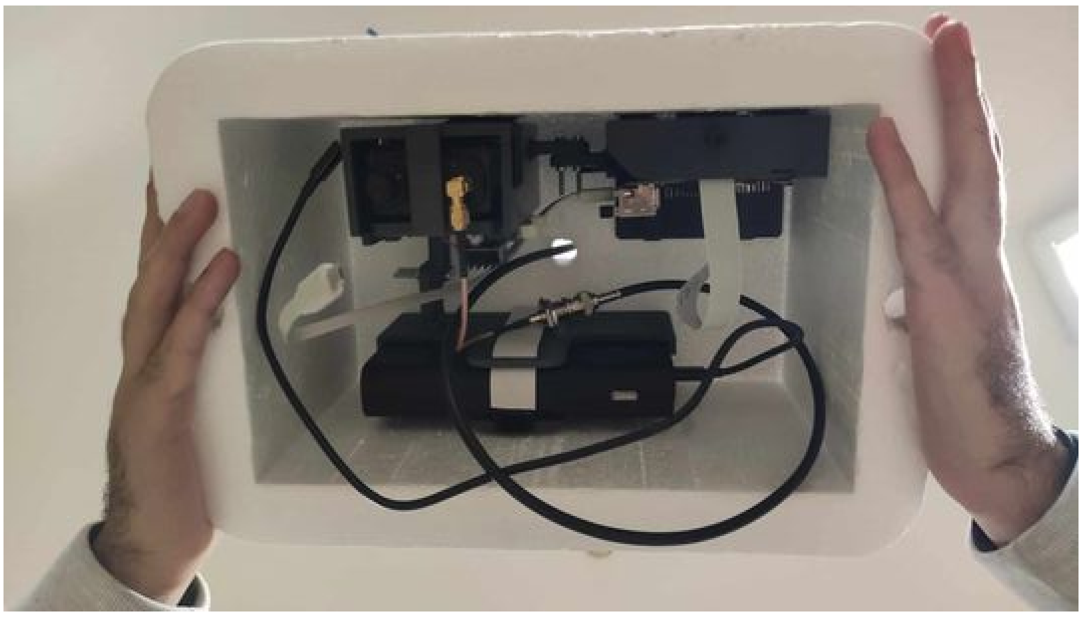


Figure 4.21: Final version of the payload with the Kenwood TH-D74E.

The procedure of the test was the same as the test in Nazaré and after everything was ready the HAB was launched. Fig. 4.22 shows the launch site in Coruche. This time we were receiving a signal for far longer than the first test, but it was losing packets and we did not get a single image while the HAB was airborne. Upon inspection of the hardware, the SMA connector of the antenna mounted on the car was broken. After recovering the payload, we verified that it was still transmitting. The images shown in Fig. 4.23 were retrieved from the microSD card.

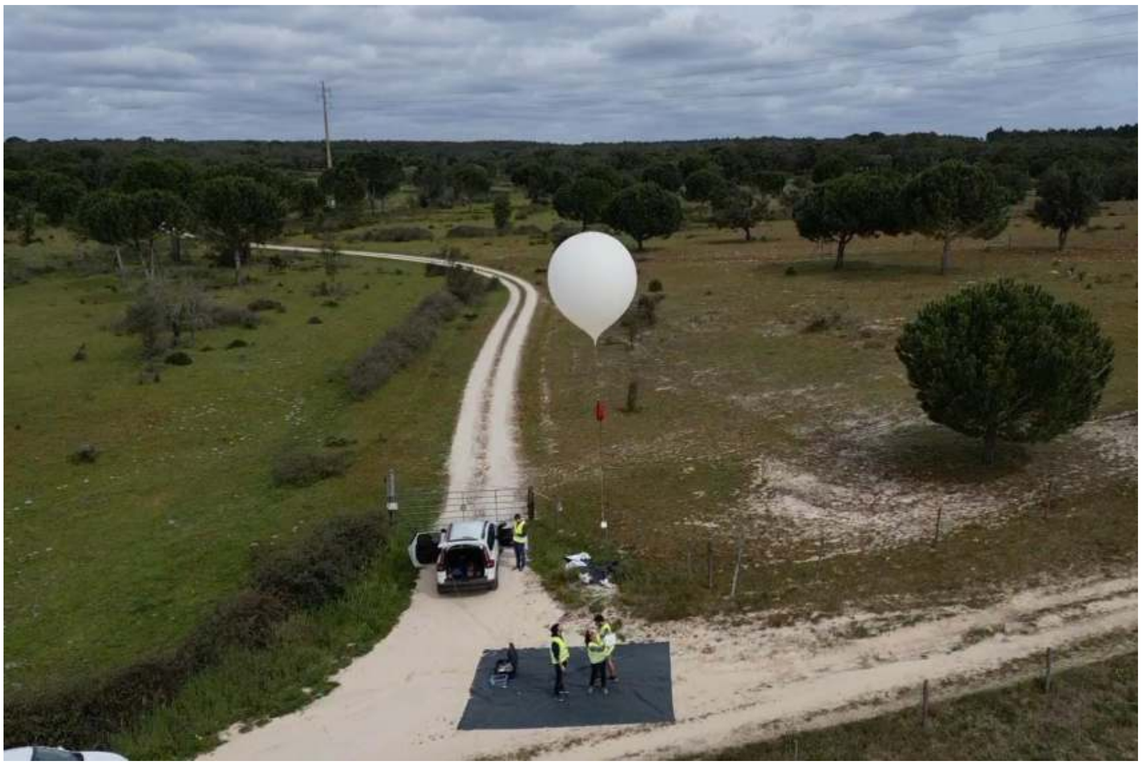


Figure 4.22: Launch preparation in Coruche.

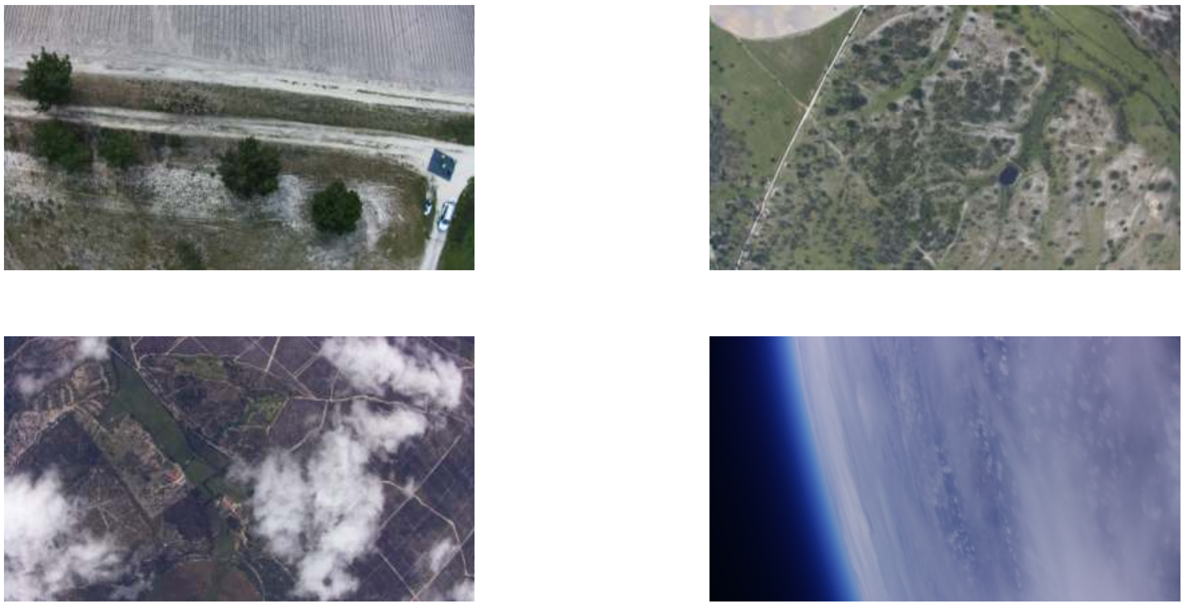


Figure 4.23: Some retrieved images from the microSD card.

Finally, in the lab, the captured images metadata was changed in order to include the geographic coordinates of where the photo was taken. This could be done by simply using the exif python library and updating the latitude, longitude, and altitude fields with some random coordinates. The next step was to get the GPS signal from the TH-D74E radio in the computer. To do so, the radio has to be put in APRS mode and it will start transmitting via the serial USB connection NMEA sentences, which is GPS information.

By using the pynmea2 Python library, the process of extracting the coordinates was simplified. Now everything needs to be put together, however, another issue was found. If the radio is in APRS mode, which is a digital radio system that allows users to share information, such as their location, weather data, and text messages, then this information can be displayed on a map, which can be used to track the movement of objects, identify weather stations, and more. However, it cannot transmit anything from the computer and the GPS data can only be obtained in that mode. After some searching on the documentation and some forums, we found a command that allowed us to switch to APRS mode from the KISS mode, which is the only way to send data to the TNC to be transmitted by the radio. But after exiting that mode we could not switch it back. After this setback, of it being extremely slow and the possibility of the radio freezing mid-flight, it was decided to abandon this radio and try to implement a solution with a SDR.

Price

The price of each component used in this version can be seen in Table 4.1:

Table 4.1: Price table of EIS version 1.

Component	Price in €
Kenwood TH-D74E x2	1370.0
Raspberry Pi 4 Model B 8GB	115.0
LM2596 DC-DC Voltage Regulador	4.0
USB-C Pdc004 PD 12V DC	13.0
Raspberry Pi High Quality (HQ) Camera	60.0
16mm lens for Raspberry Pi HQ camera	68.0
Xiaomi Mi 50W Power Bank 20000mAh	60.0
Total	1690.0

4.3 Version 2 - ADALM-PLUTO SDR

4.3.1 Revision Requirements

In order to improve the previous system, we decided to discard the Kenwood TH-D74E radio and to select individual components for each of the previously used radio functionalities, namely the data transmission and GPS information. The AX.25 protocol does not seem like it is capable of handling the transmission of big files and for simple telemetry data, another protocol should be used.

An issue that the previous version also had, was that the images' time and date information were wrong because the Raspberry Pi cannot update its own clock without access to the internet. This next version should be able to keep track of the time, using an Real-Time Clock (RTC) and it should be sent along with the GPS data.

4.3.2 Hardware

Payload:

1. ADALM-PLUTO SDR
2. 2.4GHz Omnidirectional Antenna
3. Raspberry Pi 4 Model B 8GB
4. Raspberry Pi High-Quality camera
5. Xiaomi Mi 50W Power Bank 20000mAh
6. LoRa Module
7. GPS Module
8. RTC Module
9. Power Amplifier Module

Ground station:

1. Laptop
2. ADALM-PLUTO SDR
3. 2.4GHz Directional Antenna
4. Ground Segment

5. Low Noise Amplifier Module

6. USB C Power Delivery Board

7. Xiaomi Mi 50W Power Bank 20000mAh

8. LoRa Module

9. ESP32

ADALM-PLUTO SDR and Amplifiers

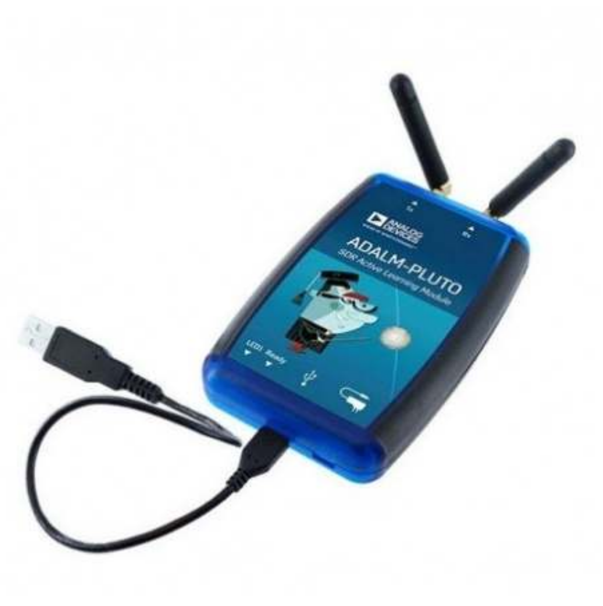


Figure 4.24: ADALM-PLUTO SDR.

To improve the communications capabilities, it was decided to use an ADALM-PLUTO SDR, as seen in Fig. 4.24, instead of a handheld radio. This one in particular is a learning module, but it provides much more than the previous solution. It offers a range between 325 - 3800 MHz with a 20 MHz bandwidth and it is possible to increase it by modifying its configuration files to enable a range of 70 - 6000 MHz and a bandwidth of 52 MHz [43]. It was not necessary to do this modification, because it was decided to use a frequency of 2.4 GHz which is a free band. With this SDR, it is possible to analyze the spectrum and better tune our solution in comparison to doing it blindly with the previous radio. It is self-powered by the USB port that connects to the computer, which means it cannot output as much power as the Kenwood TH-D74E radio, with only a maximum output of 7 dBm or 5 mW. A power amplifier is necessary and the CN0417 [44] was chosen. This module can be seen in Fig. 4.25.

It is recommended to be used with the ADALM-PLUTO SDR, it has a gain of 20dB and 100 mW output power and it is USB-powered. On the ground station side, a Low Noise Amplifier (LNA Wide Band 50-2500 MHz amplifier F1OPA [45]) was used. In Fig. 4.26 this module can be seen. It offers a maximum gain of 20-24 dB but at our selected frequency (2.4 GHz) the gain is only 8.1 dB which is small, but it is better than nothing. It is powered by a USB C PD module of 12 V.

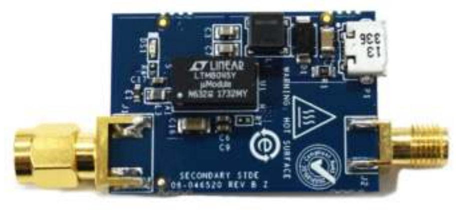


Figure 4.25: Power Amplifier.

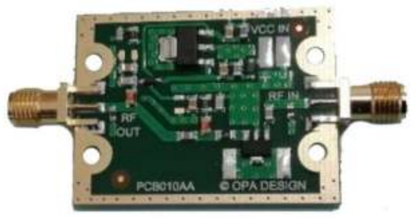


Figure 4.26: Low Noise Amplifier.

This SDR can be used with MATLAB or GNU Radio, however, since this will be running on a Raspberry Pi, it was decided to create the flow graphs using the latter option. Also, it can output a Python script which means the previous scripts can be re-utilized.

The new Antennas and the Ground Segment

It was necessary to replace the included antennas of the ADALM-PLUTO SDR because they were designed to operate from 824 MHz to 894 MHz/1710 MHz to 2170 MHz according to the product highlight from the manufacturer. Therefore, for the payload, a 2.4 GHz omnidirectional antenna with a gain of 6 dBi, as seen in Fig. 4.27, and for the ground station a 2.4 GHz directional antenna with a gain of 24 dBi and a beam of 8° , to improve the reception of the signal, were used. Fig. 4.28 shows the antenna and the radiation pattern on the horizontal plane. It is necessary to be mounted on a mechanical device that can point it to the object to track, but for testing purposes, it was mounted on a mast and manually oriented to the transmission location.

Figure 4.27: 2.4 GHz omnidirectional Antenna.

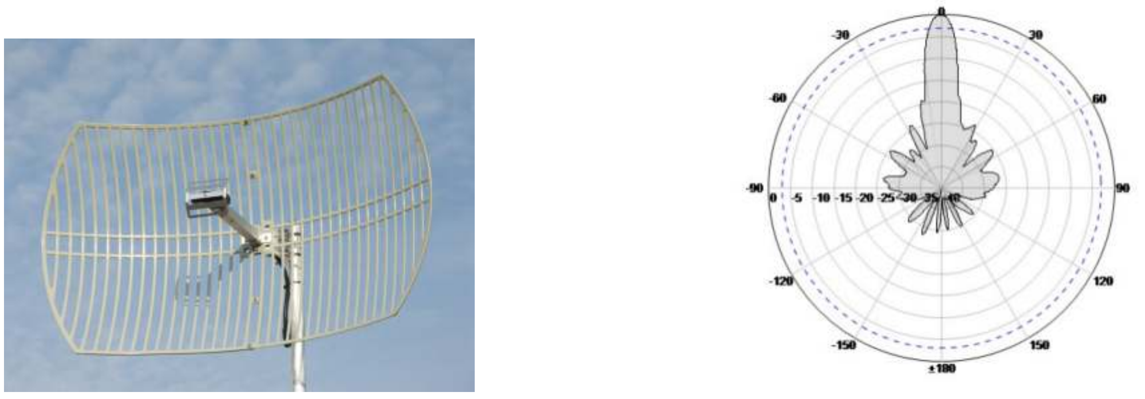


Figure 4.28: 2.4 GHz grid dish Antenna and its radiation pattern on the horizontal plane.

In order to point the antenna automatically, we contacted a third party to build a ground station based on an open-source project, the SatNOGS [46]. It has two rotators controlled by an Arduino. It is connected via USB to a computer and controlled using the software rotctld from Hamlib, which is a library used by amateur radio. We cannot specify more details about this system, because they are confidential. The price of this tracking system is much cheaper than what we can find on the market. The tracking system can be seen in Fig. 4.29.

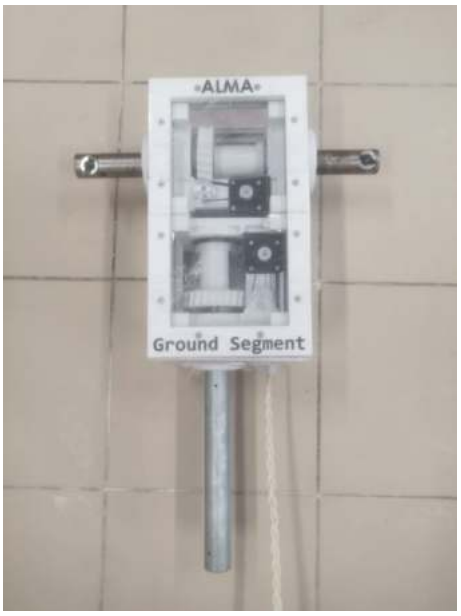


Figure 4.29: Ground segment body.

Due to manufacturing delays, it was never used in the project.

The LoRa, GPS, and RTC Modules

To ensure that the directional antenna can be pointed to the payload, a GPS NEO-7M module, as seen in Fig. 4.30, was used. It comes with a ceramic antenna that connects to the module with an I-PEX connector.

Figure 4.30: GPS NEO-7M Module [47].

This antenna had some problems while testing this module. It would easily disconnect when a little force was applied to it, like a slight bend, and by removing it and inserting it over some time, it broke, proving its fragility. To go around this issue, an I-PEX to SMA adaptor and a different antenna, from a SIRESP radio which had Global Navigation Satellite Systems (GNSS) functionality, were used. This module transmits NMEA GPS commands via its UART pins, at a 9600 baud rate, that connect to one of the UARTs of the Raspberry Pi on the GPIO.

To send and receive the geolocation data, so that the antenna is pointed to the payload, two LoRa modules, represented in Fig. 4.31, were used. The used module was a generic one using the radio modem SX1276 [48]. The data sheet provided by the seller was wrong and some web searching and trial and error were necessary to make it function.

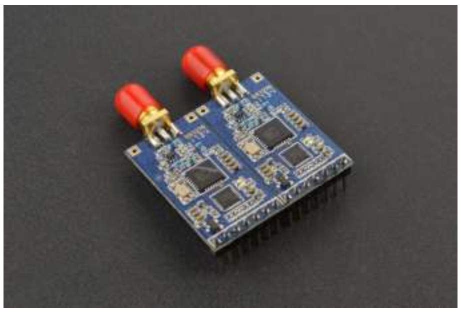
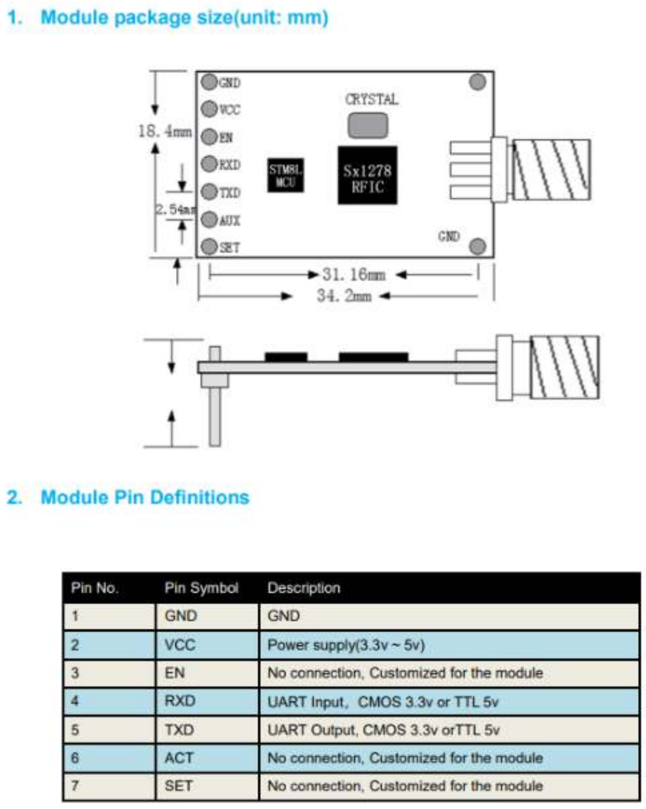


Figure 4.31: Used LoRa module [49].



1. Module package size(unit: mm)

GND
VCC
EN
RXD
TXD
AUX
SET
CRYSTAL
STM8L MCU
Sx1278 RFIC
GND
18. 4mm
2. 54mm
31. 16mm
34. 2mm

2. Module Pin Definitions

Pin No.	Pin Symbol	Description
1	GND	GND
2	VCC	Power supply(3.3v ~ 5v)
3	EN	No connection, Customized for the module
4	RXD	UART Input, CMOS 3.3v or TTL 5v
5	TXD	UART Output, CMOS 3.3v orTTL 5v
6	ACT	No connection, Customized for the module
7	SET	No connection, Customized for the module

Figure 4.32: Information provided by the data sheet [49].

According to a screenshot, shown in Fig. 4.32, from the datasheet the EN, SET, and AUX (ACT in the table on the figure) pins do not need to be connected to anything, however, that is not true. It would only function if EN is connected to GND. It also described some AT commands to configure the module but they never seemed to work. Later it was discovered that the AT commands were being transmitted by the module. Since we managed to make it function we decided to go forward with it. This module is also connected to one of the Raspberry Pi UARTs on other pins of the GPIO. As for the ground station, this module is connected to an ESP32, which displays the received data on a terminal.

To keep track of time and to have the most accurate information of when the photos were taken a RTC, shown in Fig. 4.33, was used. This information is also sent via LoRa with the geolocation data. This module uses the Inter-Integrated Circuit (I2C) protocol to send data to the Raspberry Pi. It is connected to I2C pins of the Raspberry Pi GPIO.

Figure 4.33: RTC module [50].

Everything was connected using jumpers while testing, however, it could not be mounted like this on the payload because of the high risk of a cable disconnecting if there is some turbulence during the flight. Thus, a HAT (Hardware Attached on Top) PCB for the Raspberry Pi was designed, using Altium, to accommodate all the modules and interface them to the GPIO, making everything more compact and secure. This design can be seen in Fig. 4.34. In a second version, some mistakes were fixed, namely connection Tx to Tx and Rx to Rx, because of the Altium netlabels, and an extended header was added so that we could still have access to the GPIO, even after having the HAT mounted. This final version is shown in Fig. 4.35. It can be seen fully assembled in Fig. 4.36

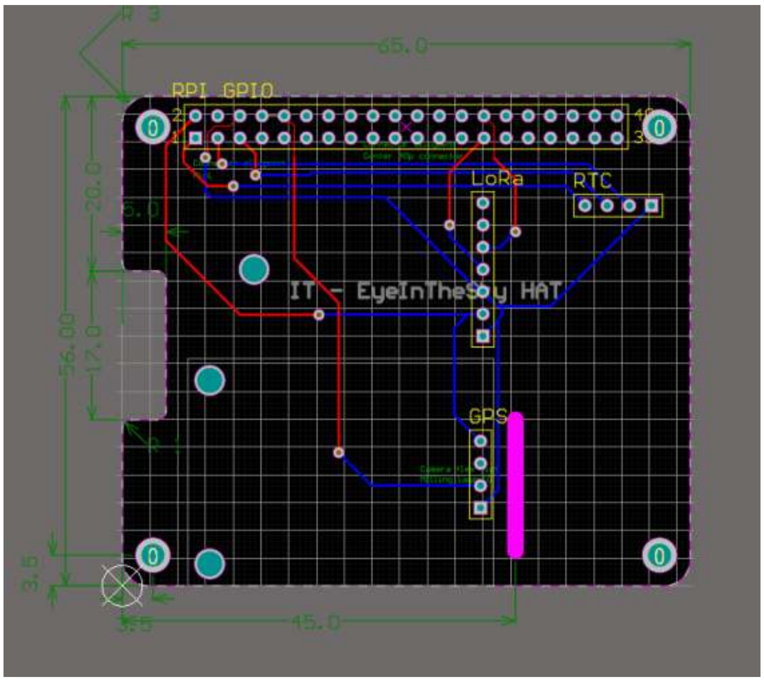


Figure 4.34: HAT PCB Layout.

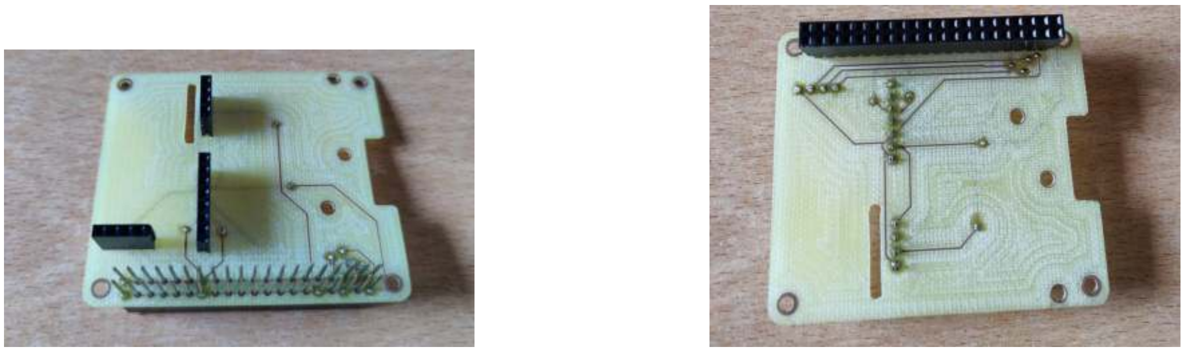


Figure 4.35: HAT top and bottom.

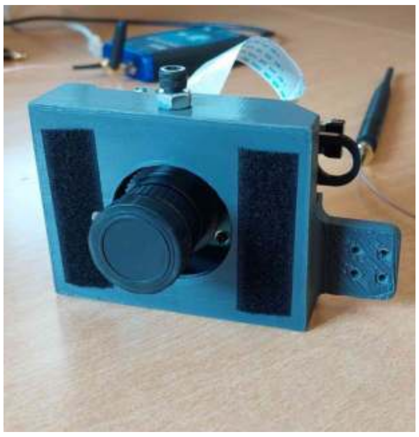
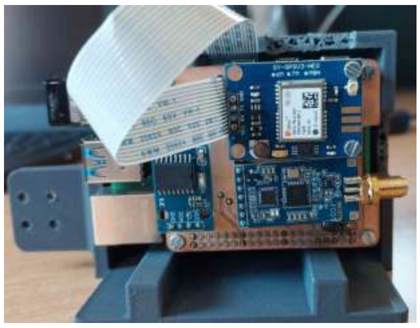



Figure 4.36: Front view and back view of the complete package (Raspberry Pi, Camera, and Modules).

Price

Table 4.2: Price table of EIS version 2.

Component	Price in €
ADALM-PLUTO x2	543.48
Raspberry Pi 4 Model B 8GB	115.0
CN0417 Power Amplifier (PA)	52.0
USB-C PDC004 PD 12V DC	13 .0
LNA Wide Band 50-2500MHz amplifier F1OPA	34.96
16mm lens for Raspberry Pi HQ camera	68.0
Xiaomi Mi 50W Power Bank 20000mAh	60.0
2.4 GHz omnidirectional Antenna	5.5
2.4 GHz directional Antenna	79.9
NEO-7M GPS Module	12.95
LoRa Radio Module - 868MHz	40.0
DS3231 RTC Module	5.54
Total	1030.33

The final price can be seen in Table 4.2. It is only missing the price of the tracking system because, as said before, the price is confidential. Still, it is much cheaper than what can be found on the market, which is in the order of the tens of thousands.

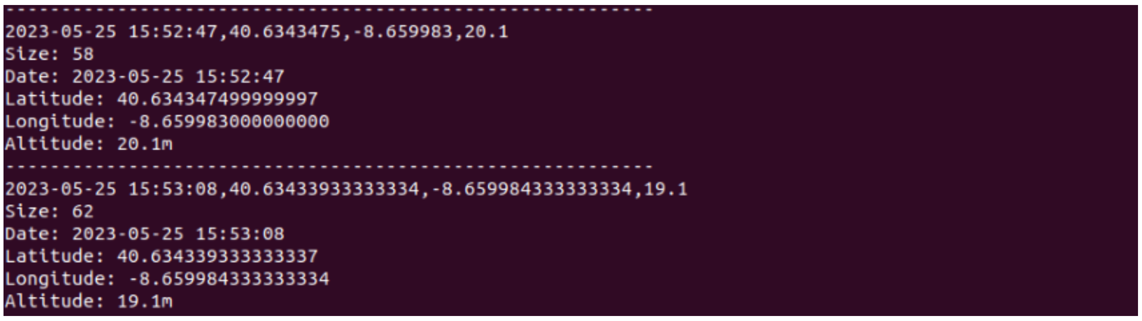
4.3.3 Software

The tests and script were done using the following software and protocols:

1. GNU Radio
2. libcamera-still
3. PIL python library
4. GPS - NMEA
5. Exif python library

Software and Tests

For this version, the script that would take photos automatically was modified. Now, instead of just taking photos every 30 seconds, it takes a photo, reads the serial connection with the GPS module to get the current position, and, using the Python exif library, modifies the new image metadata with the geolocation. In between getting the position and modifying the image's metadata, it sends a message via LoRa containing the geolocation and a timestamp. An example of this can be seen in Fig. 4.37.



```
----------------------------------------------------------
2023-05-25 15:52:47,40.6343475,-8.659983,20.1
Size: 58
Date: 2023-05-25 15:52:47
Latitude: 40.634347499999997
Longitude: -8.659983000000000
Altitude: 20.1m
----------------------------------------------------------
2023-05-25 15:53:08,40.63433933333334,-8.659984333333334,19.1
Size: 62
Date: 2023-05-25 15:53:08
Latitude: 40.634339333333337
Longitude: -8.659984333333334
Altitude: 19.1m
```

Figure 4.37: Received message with LoRa.

The cycle repeats 20 seconds after finishing. While testing, this LoRa module proved that it was not capable of long-distance transmissions as we could only receive data a few meters away. The payload was placed on the edge of the roof of IT-1 and we were with the other LoRa module and with a laptop moving away. The maximum distance we could receive a message was about 25 meters, far less than the advertised distance. This distance can be visualized in Fig. 4.38.

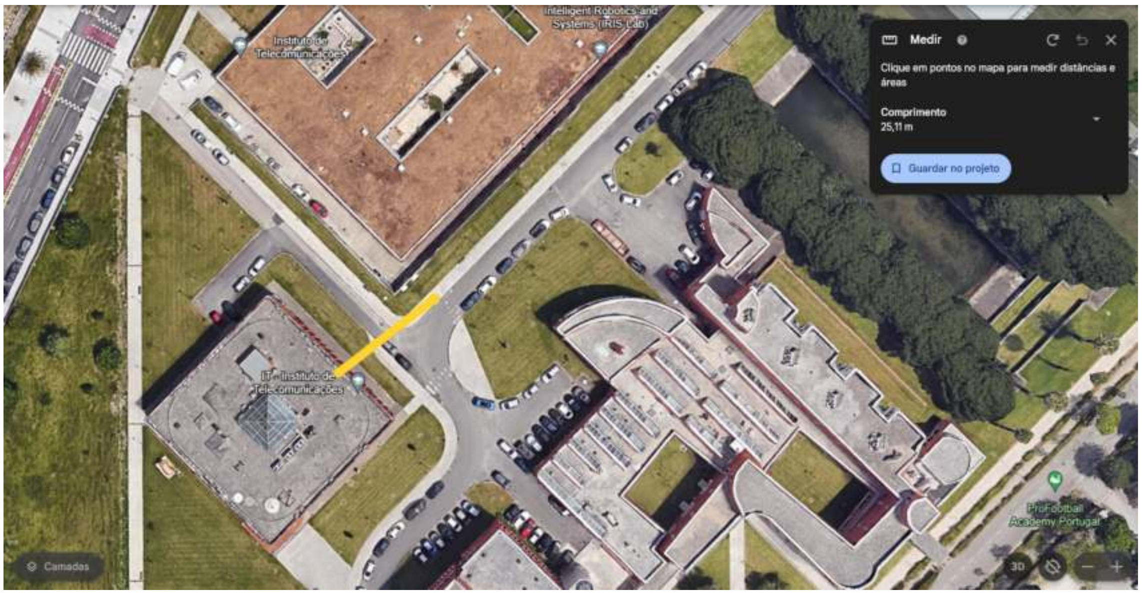
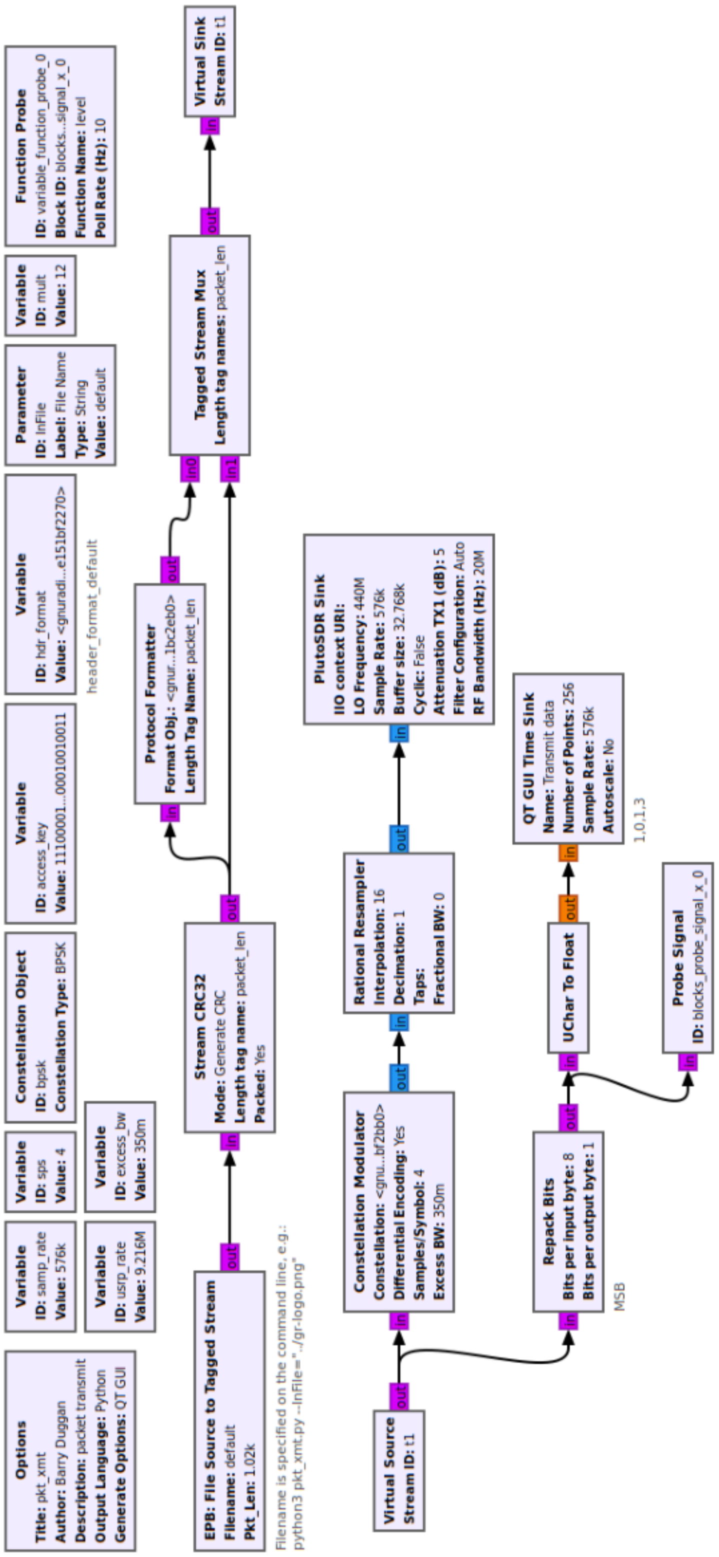

Intelligent Robotics and Systems (IRIS Lab)
Instituto de Telecomunicações
Medir
Clique em pontos no mapa para medir distâncias e áreas
Comprimento
25,11 m
Guardar no projeto
IT - Instituto de Telecomunicações
ProFootball Academy Portugal
Camadas
3D

Figure 4.38: LoRa testing location and distance.

Either we got a bad module or because of the wrong datasheet, something else should have been connected to VCC or GND from the pins that they said had no connection. Because of this, the LoRa was no longer tested, however, the functionality is working.

For the transmission of the images with the ADALM-PLUTO, GNU Radio was used. It was quite challenging to learn how to use this software, paired with this SDR. The first test we did was to do a Frequency Modulation (FM) transmission of music and use the Kenwood TH-D74E radio to listen to it. After successfully transmitting audio with the SDR, we searched about how to send files using packets and found a wiki page, on GNU Radio, with a tutorial [51] about sending files using packets and Binary Phase-Shift Keying (BPSK). This included the GitHub repository with the flow graphs and a Python script ready to test. In the beginning, it did not work, but after communicating and collaborating with the author of that wiki page, Mr. Barry Duggan [52], we managed to make it work, by making some modifications and adjusting the sample rate and bandwidth.



Options
Title: pkt_xmt
Author: Barry Duggan
Description: packet transmit
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 576k

Variable
ID: sps
Value: 4

Constellation Object
ID: bpsk
Constellation Type: BPSK

Variable
ID: access_key
Value: 11100001...00010010011

Variable
ID: hdr_format
Value: <gnuradi...e151bf2270>
header_format_default

Parameter
ID: InFile
Label: File Name
Type: String
Value: default

Variable
ID: mult
Value: 12

Function Probe
ID: variable_function_probe_0
Block ID: blocks...signal_x_0
Function Name: level
Poll Rate (Hz): 10

Variable
ID: usrp_rate
Value: 9.216M

Variable
ID: excess_bw
Value: 350m

EPB: File Source to Tagged Stream
Filename: default
Pkt_Len: 1.02k
Filename is specified on the command line, e.g.:
python3 pkt_xmt.py --InFile="../gr-logo.png"

Stream CRC32
Mode: Generate CRC
Length tag name: packet_len
Packed: Yes

Protocol Formatter
Format Obj.: <gnur...1bc2eb0>
Length Tag Name: packet_len

Tagged Stream Mux
Length tag names: packet_len

Virtual Sink
Stream ID: t1

Virtual Source
Stream ID: t1

Constellation Modulator
Constellation: <gnu...bf2bb0>
Differential Encoding: Yes
Samples/Symbol: 4
Excess BW: 350m

Rational Resampler
Interpolation: 16
Decimation: 1
Taps:
Fractional BW: 0

PlutoSDR Sink
IIO context URI:
LO Frequency: 440M
Sample Rate: 576k
Buffer size: 32.768k
Cyclic: False
Attenuation TX1 (dB): 5
Filter Configuration: Auto
RF Bandwidth (Hz): 20M

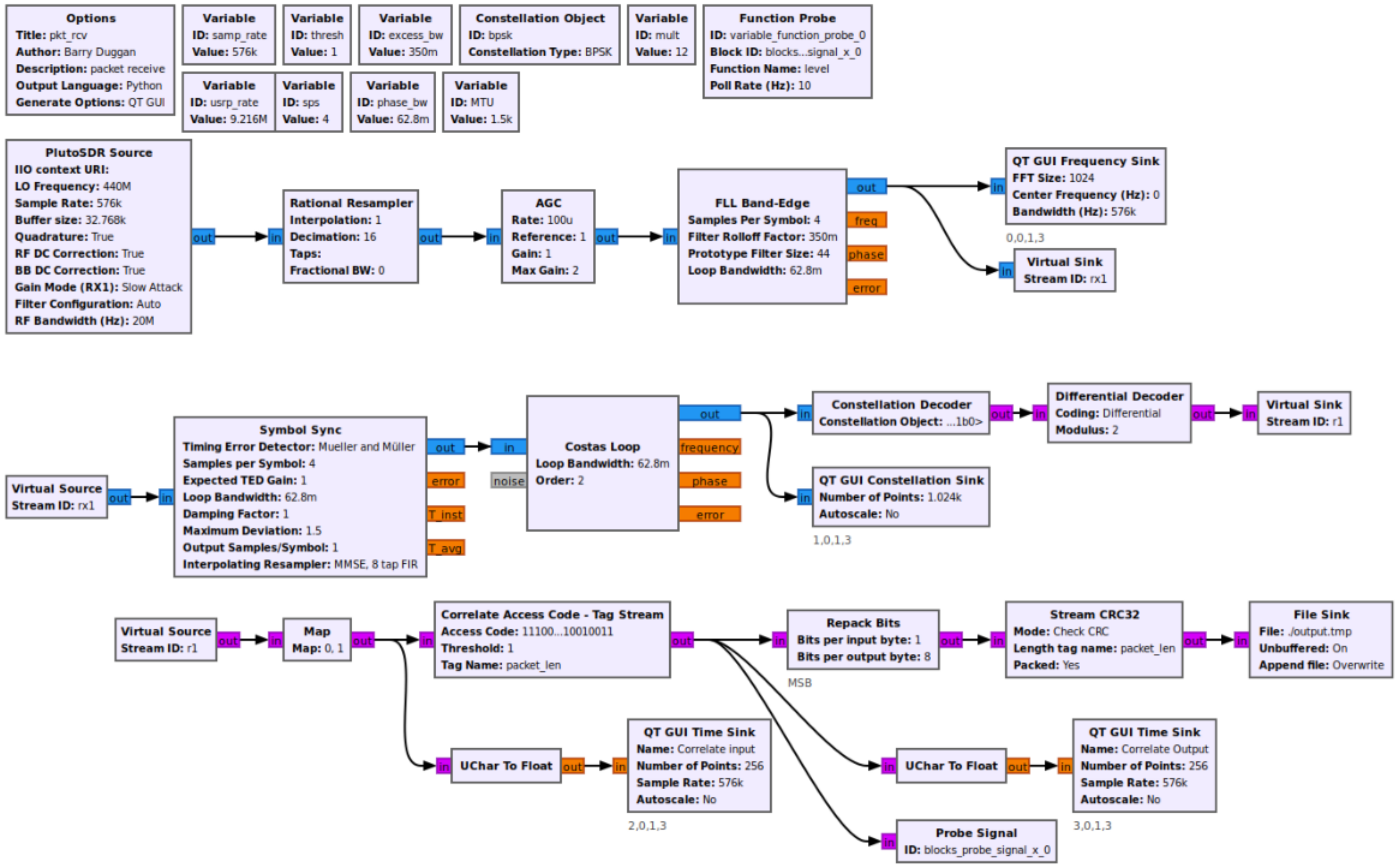
Repack Bits
Bits per input byte: 8
Bits per output byte: 1
MSB

UChar To Float

QT GUI Time Sink
Name: Transmit data
Number of Points: 256
Sample Rate: 576k
Autoscale: No
1,0,1,3

Probe Signal
ID: blocks_probe_signal_x_0

Figure 4.39: Transmitter flow graph.



Options
Title: pkt_rcv
Author: Barry Duggan
Description: packet receive
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 576k

Variable
ID: thresh
Value: 1

Variable
ID: excess_bw
Value: 350m

Constellation Object
ID: bpsk
Constellation Type: BPSK

Variable
ID: mult
Value: 12

Function Probe
ID: variable_function_probe_0
Block ID: blocks...signal_x_0
Function Name: level
Poll Rate (Hz): 10

Variable
ID: usrp_rate
Value: 9.216M

Variable
ID: sps
Value: 4

Variable
ID: phase_bw
Value: 62.8m

Variable
ID: MTU
Value: 1.5k

PlutoSDR Source
IIO context URI:
LO Frequency: 440M
Sample Rate: 576k
Buffer size: 32.768k
Quadrature: True
RF DC Correction: True
BB DC Correction: True
Gain Mode (RX1): Slow Attack
Filter Configuration: Auto
RF Bandwidth (Hz): 20M

Rational Resampler
Interpolation: 1
Decimation: 16
Taps:
Fractional BW: 0

AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 2

FLL Band-Edge
Samples Per Symbol: 4
Filter Rolloff Factor: 350m
Prototype Filter Size: 44
Loop Bandwidth: 62.8m

QT GUI Frequency Sink
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 576k

Virtual Sink
Stream ID: rx1

Virtual Source
Stream ID: rx1

Symbol Sync
Timing Error Detector: Mueller and Müller
Samples per Symbol: 4
Expected TED Gain: 1
Loop Bandwidth: 62.8m
Damping Factor: 1
Maximum Deviation: 1.5
Output Samples/Symbol: 1
Interpolating Resampler: MMSE, 8 tap FIR

Costas Loop
Loop Bandwidth: 62.8m
Order: 2

Constellation Decoder
Constellation Object: ...1b0>

Differential Decoder
Coding: Differential
Modulus: 2

Virtual Sink
Stream ID: r1

QT GUI Constellation Sink
Number of Points: 1.024k
Autoscale: No

Virtual Source
Stream ID: r1

Map
Map: 0, 1

Correlate Access Code - Tag Stream
Access Code: 11100...10010011
Threshold: 1
Tag Name: packet_len

Repack Bits
Bits per input byte: 1
Bits per output byte: 8

Stream CRC32
Mode: Check CRC
Length tag name: packet_len
Packed: Yes

File Sink
File: ./output.tmp
Unbuffered: On
Append file: Overwrite

UChar To Float

QT GUI Time Sink
Name: Correlate input
Number of Points: 256
Sample Rate: 576k
Autoscale: No

UChar To Float

QT GUI Time Sink
Name: Correlate Output
Number of Points: 256
Sample Rate: 576k
Autoscale: No

Probe Signal
ID: blocks_probe_signal_x_0

Figure 4.40: Receiver flow graph.

Essentially, the transmitter flowgraph, as seen in Fig. 4.39, opens a file, divides the image into several packets, and adds a preamble at the beginning and end of the file to synchronize the receiver with the transmitter. After that, it generates a Cyclic Redundancy Check (CRC) of 32 bits and adds it to the packet to be transmitted. Having the packet ready, it is then modulated in a constellation of two symbols and sent.

On the receiver side, as seen in Fig. 4.40, it is a bit more complex. The received signal must be first filtered and then the symbols must be synchronized. After having the packet received, the CRC data field is checked for its validity. If it is valid it is saved on a temporary file, otherwise it is discarded.

After receiving the file, the provided Python script must be executed to remove the preamble from the temporary file and to save the new file as an image. The repetition of symbols at the end and beginning of the file is the preamble.

These flow graphs are designed to be used manually, so it was necessary to automatize them. For that, it was necessary to export the flowgraphs as Python scripts. But only like this, after it finishes transmitting and receiving, the script does not know that it is finished. Thus, to solve that, a probe block was added to both flowgraphs and exported again to a Python script. Then, in each script, there is a counter, counting repetitions of 1s or 0s. If the same symbol is repeated for a long time, it means that the transmission has finished and the script can terminate itself.

Now that the process can be terminated by itself at the end, it can be placed inside a loop and be repeated endlessly during the mission. The receiver just does these two tasks, gets the output file, and saves it in a different location, in a loop with a delay of 10 seconds at the end of each iteration. The transmitter gets the most recent image, makes a copy of its metadata, and compresses the image using the PIL Python library. The new compressed image now has its metadata modified with the data from the original image and it is sent. After finishing transmitting the file, this script sleeps for 1 minute before sending another image. This script runs at boot.

While adjusting the timings of the termination of each script, as well as adjusting the sample rate and bandwidth, some tests were conducted. First inside the lab, with a low-power transmission, we got the following results in Fig. 4.41:

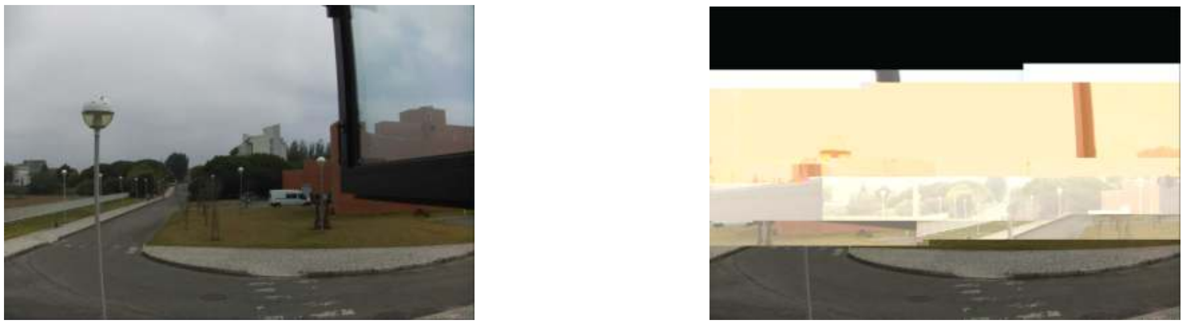


Figure 4.41: Example of received images while doing the first tests.

We noticed in the BPSK constellation graph that when someone or something moved near the radios the symbols would cross and corrupt the image, as seen in the second image, in Fig. 4.41. This could be because new reflections are created as the, already noisy, environment changes. The Wireless Fidelity (Wi-Fi) in the room could also affect the reception of the image. Depending on when the corruption happens it could have wrong colors, parts of the image could be in the wrong place, a combination of both, or a corrupted file if it happens while transmitting the metadata. Fig. 4.42 shows the received symbols with some noise while doing the tests inside the lab.

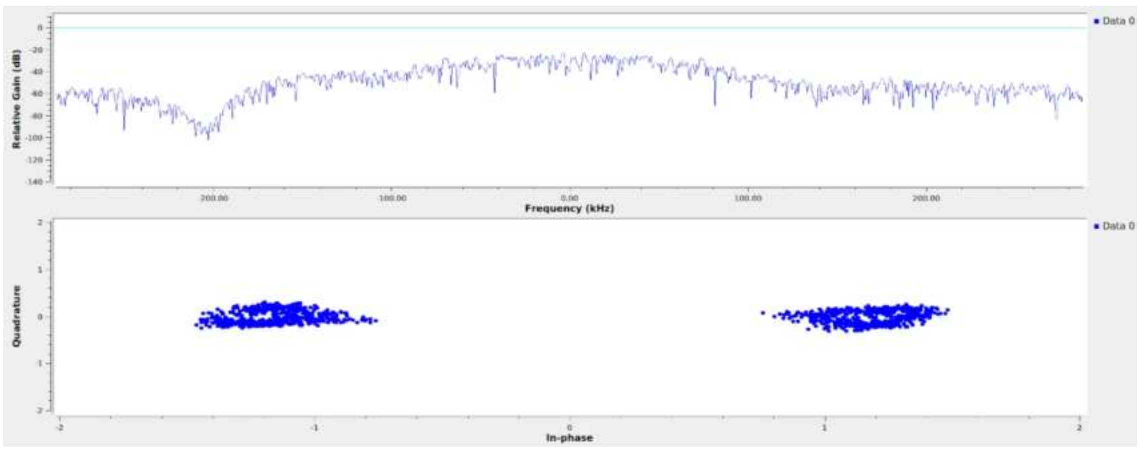


Figure 4.42: Symbols with some noise.

For the next tests, we decided to go outside with the ground station antenna and tested with the transmitter on a window in Instituto de Telecomunicações and the receiver near Departamento de Línguas e Culturas, at the University of Aveiro. This test can be seen in Fig. 4.43.

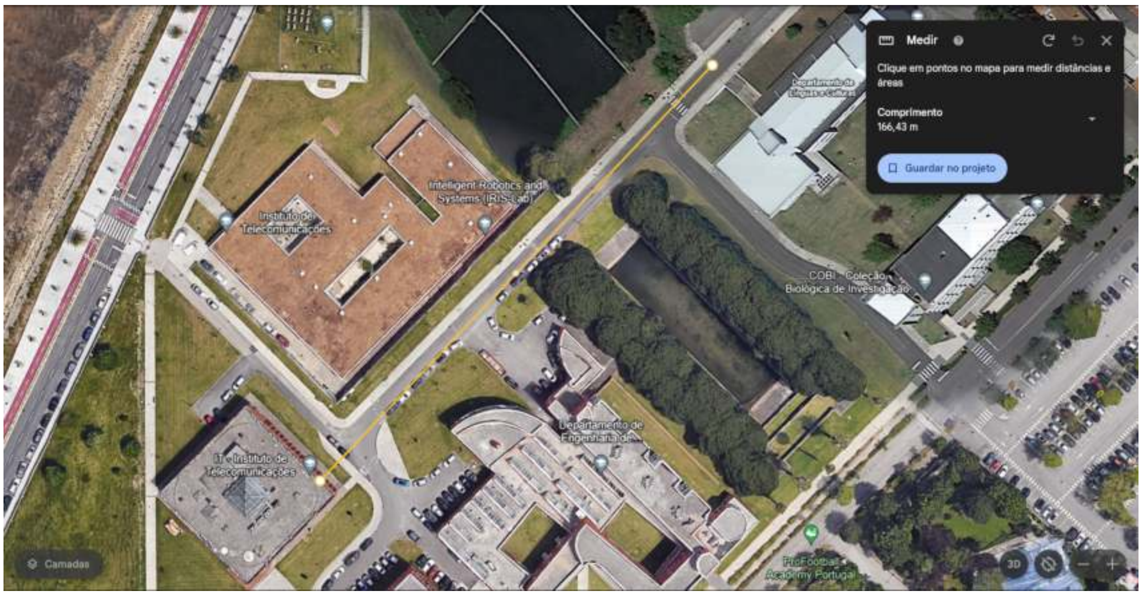

Medir
Clique em pontos no mapa para medir distâncias e áreas
Comprimento
166,43 m
Guardar no projeto
Departamento de Línguas e Culturas
Intelligent Robotics and Systems (IRIS-Lab)
Instituto de Telecomunicações
COBI - Coleção Biológica de Investigação
Departamento de Engenharia de...
IT - Instituto de Telecomunicações
ProFootball Academy Portugal
Camadas
3D

Figure 4.43: Distance between transmitter and receiver on the second test.

Now, the only time the symbols would cross, was when the antenna was moved, because it was being held by hand. Some people passed by and it did not seem to affect the performance, proving the conclusions of the previous test of the interference from reflections and possibly the Wi-Fi network inside the lab.

Next, to make sure that the payload was under the 1 kg restriction each component was weighted. It can be seen in Table 4.3:

Table 4.3: Mass budget.

Parts	Weight (g)	Where to cut	Alternative
GPS Antenna	188.3	X	GPS antenna with shorter cable
2.4GHz Antenna with adapter	32.6		
Power Amplifier	13.6		
PLUTO SDR with one antenna on RX	122.5		
Rpi + HAT + HQ Camera + Plastic supp	266.3		
LoRa Antenna	36.9	X	A better antenna with a shorter cable
20000 mAh Power bank + Plastic supp	531.3	X	A power bank with half the capacity and a smaller plastic supp
USB C male - USB C male	16.7		
USB A male - Micro USB male	51.3	X	Shorter cable (this one has multiple ports: lighting, micro and C)
USB A male - Micro USB male	50.0	X	Shorter cable
Stirofoam box	116.6		
Total	1426.1		

It passes the weight limit by 426 grams. It is possible to cut some weight by having shorter cables inside and maybe by reducing the filling of the 3D printed structures. The heaviest component can also be replaced by a power bank with a lower capacity and with the same functionalities, mainly the USB PD and one extra USB port, at least.

To verify that the current power bank can be replaced by a smaller one, a one-hour test was conducted using an USB voltmeter/ammeter, as seen in Fig. 4.44. This test shows that it is consuming, on average, 1070mAh.

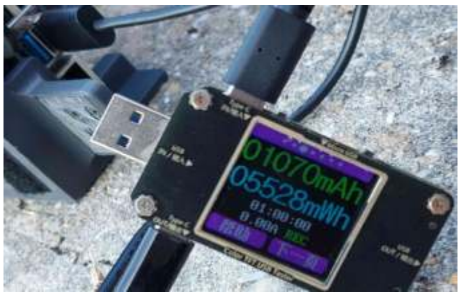


Figure 4.44: USB voltmeter/ammeter with a result of 1070maAh after one hour.

According to the previous figure, we can safely reduce the capacity of the power bank by four to five times, since in one hour of operation (GPS, LoRa and SDR transmission) it only consumed around one-twentieth of its capacity and a normal mission should last two to three hours.

After that, we tested the transmission time with different types of compression quality. This was done inside the lab, but this time the test was conducted with a low power and the radios close to each other. The results can be seen in Table 4.4.

Table 4.4: Transmission time for each quality of compression.

Image quality (%)	Transmission time (s)	File size (bytes)
50	121	102628
75	190	159055
100	327	317435

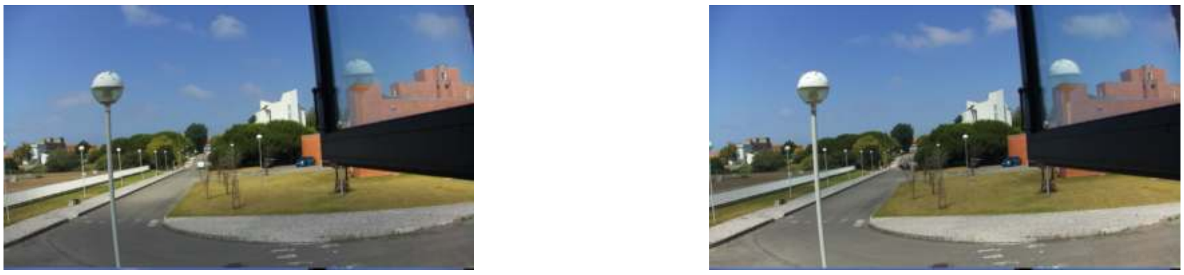


Figure 4.45: Example of received images with 50% and 75% image quality.

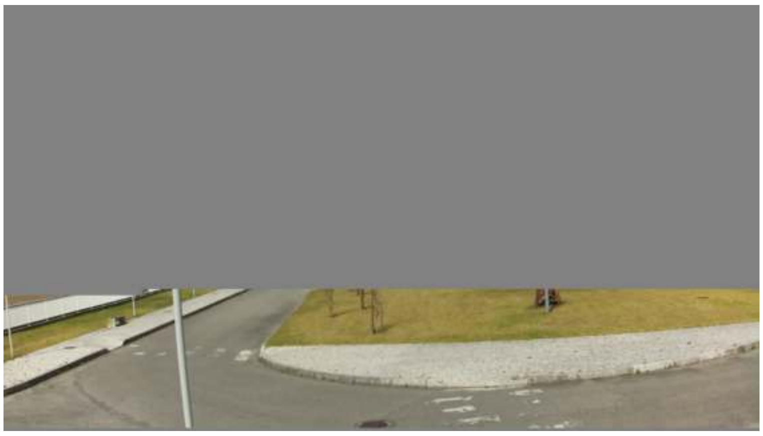


Figure 4.46: Example of the received image with 100% image quality.

The difference between the images is not distinguishable by the human eye unless we zoom in on each image and look for it, as seen in Fig. 4.45. And if a file is bigger than 250KB it is not completely transmitted, as shown by Fig. 4.46. This last test was repeated several times and the result was always the same. Probably it is a limitation of the ADALM-PLUTO or a software limit with GNUradio.

Finally, a long-distance test was conducted. Given our available conditions, the first test was done from the roof of IT-1 where the ground station was installed with the antenna mounted on a mast, pointed to the transmitter, and the payload was placed above a bridge, around 2.2 km away from the receiver. It was the furthest we could go with a line of sight. The two locations and the distance between them can be seen in Fig. 4.47. The transmitter and receiver setup and the final version of the payload can be seen in Fig. 4.48 and 4.49.

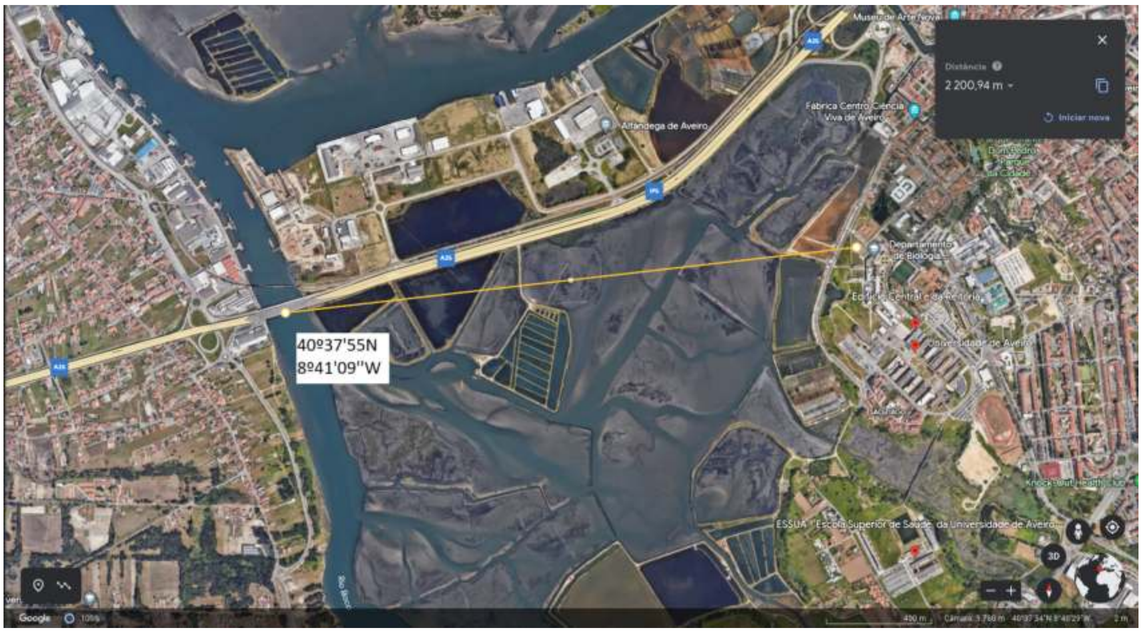


Figure 4.47: Distance between the receiver and the transmitter.

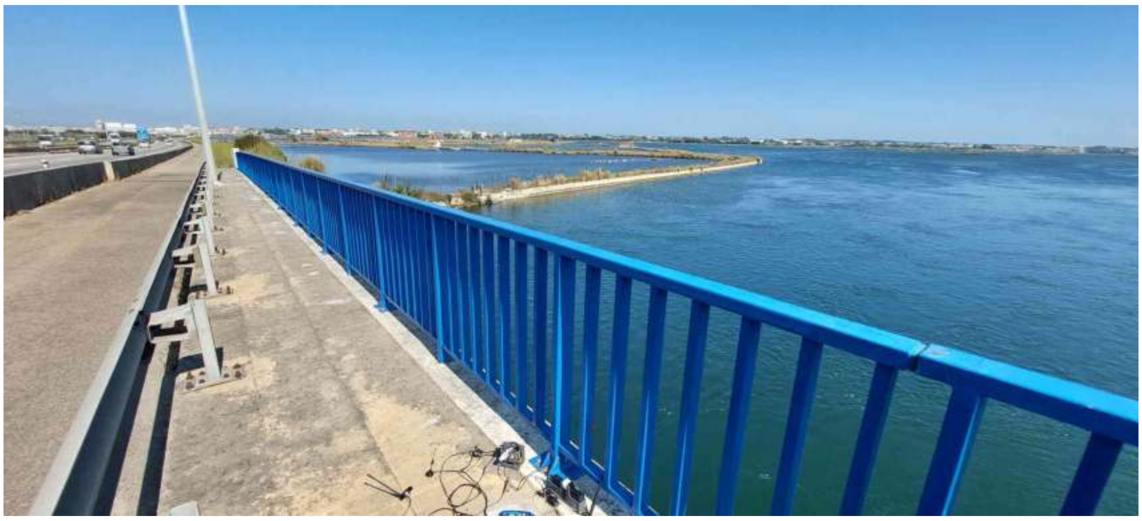


Figure 4.48: Transmitter setup.

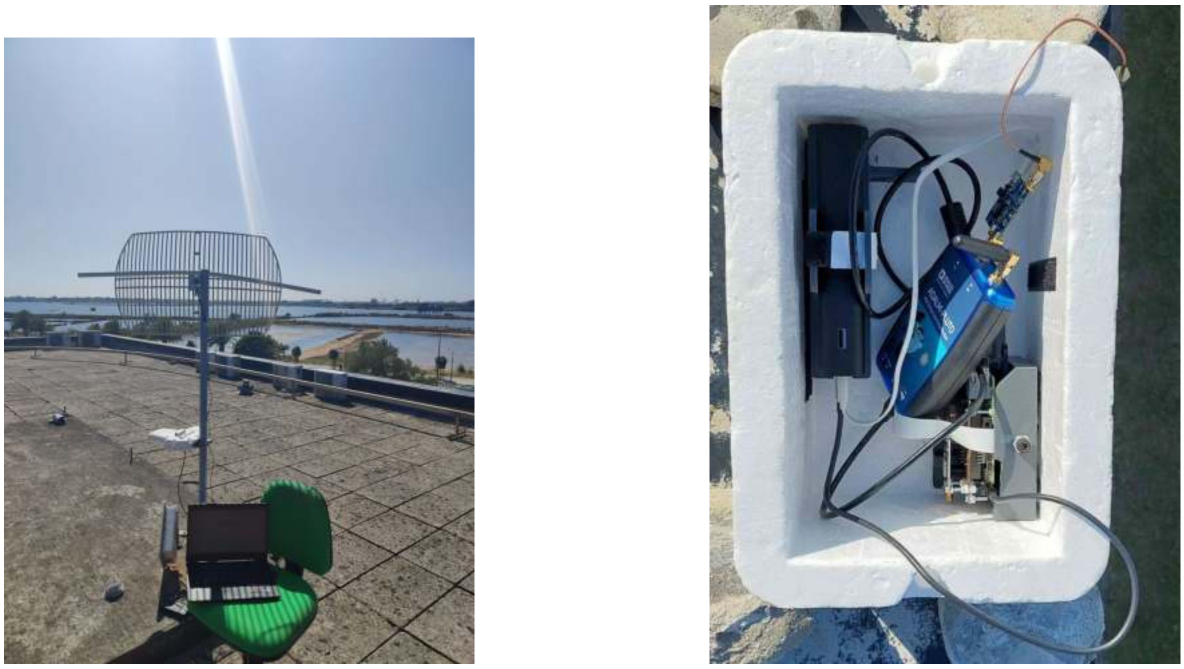


Figure 4.49: Receiver setup and last version of payload.

This was the test with the best results yet. At 2.2km and with a line of sight, the symbols, seen in Fig. 4.50, seemed better than when it was tested inside the lab or near the lab. And to try to simulate some turbulence we picked up the box and moved it around and the symbols did not cross each other.

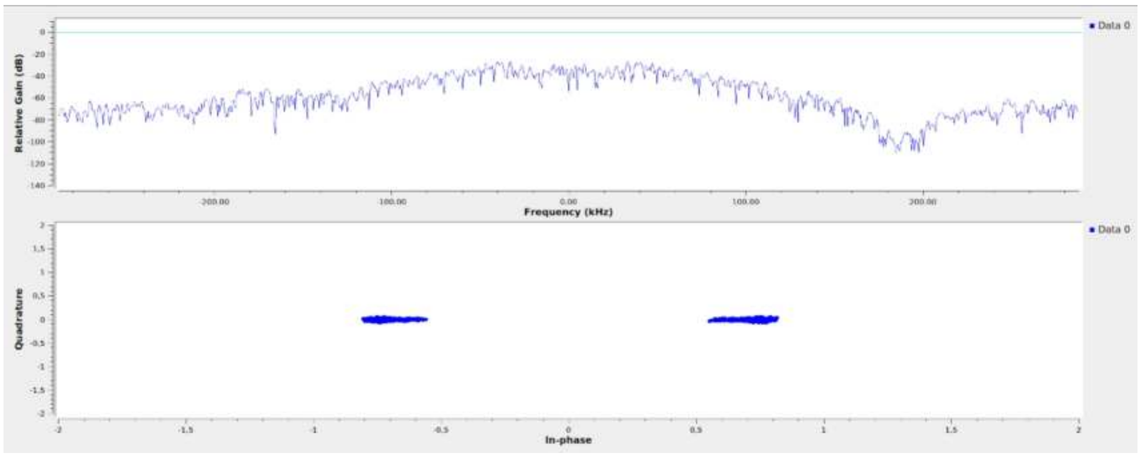


Figure 4.50: Long distance test constellation.

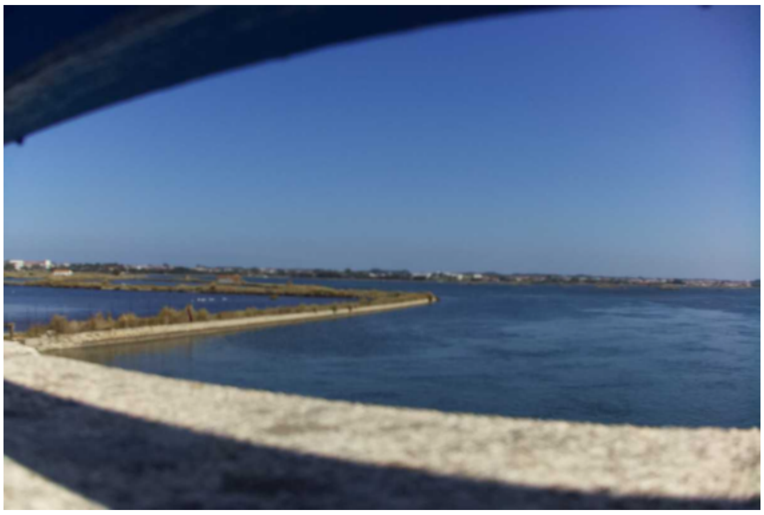
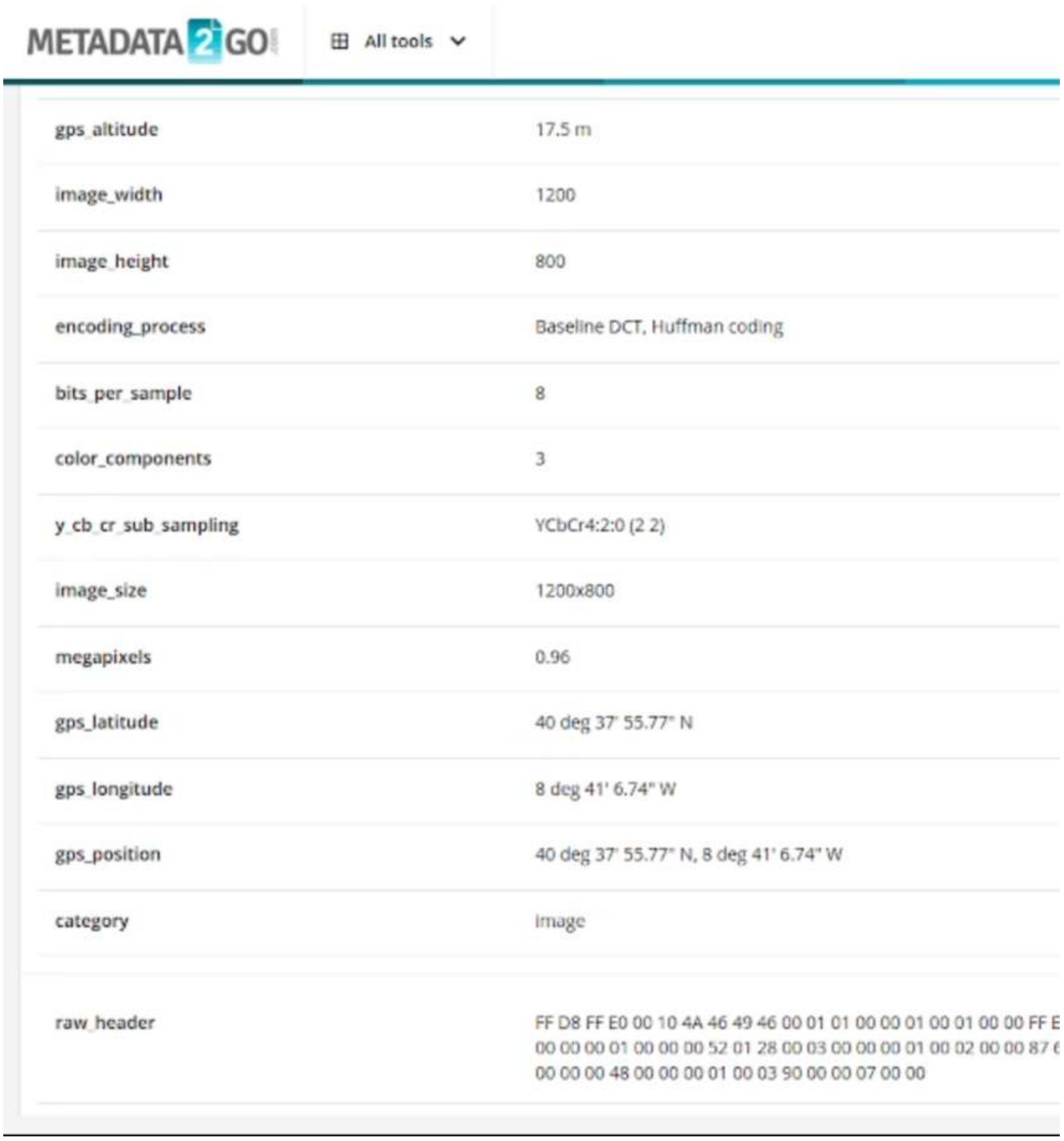


Figure 4.51: Long distance test received image.

The geolocation in the metadata, as seen in Fig. 4.52, of the received image in Fig. 4.51, matches the coordinates of Fig. 4.47 except for the altitude. During previous tests, we noticed that the altitude measurement was not reliable. It is possible to add an altimeter module to the HAT and measure the barometric pressure, to determine the altitude.



gps_altitude	17.5 m
image_width	1200
image_height	800
encoding_process	Baseline DCT, Huffman coding
bits_per_sample	8
color_components	3
y_cb_cr_sub_sampling	YCbCr4:2:0 (2 2)
image_size	1200x800
megapixels	0.96
gps_latitude	40 deg 37' 55.77" N
gps_longitude	8 deg 41' 6.74" W
gps_position	40 deg 37' 55.77" N, 8 deg 41' 6.74" W
category	image
raw_header	FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01 00 01 00 00 FF E 00 00 00 01 00 00 00 52 01 28 00 03 00 00 00 01 00 02 00 00 87 00 00 00 48 00 00 00 01 00 03 90 00 00 07 00 00

Figure 4.52: Metadata of received image.

The second test was conducted from an even further distance of 10.5 km, as seen in Fig. 4.53, but it was unsuccessful.

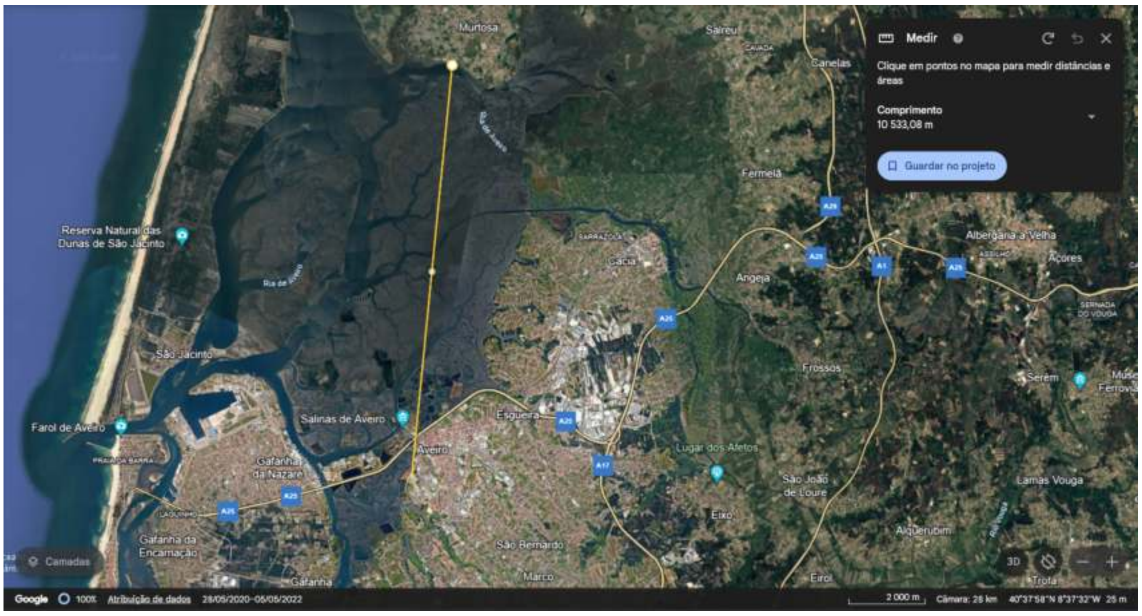


Figure 4.53: Last long-distance test.

There are two possible causes for the failure of this last test. Either the transmission power is too low for this distance and frequency, which is the most likely, or the test conditions were not the best. It is possible for the highway A25 to be obstructing the line of sight, as well as some buildings in the way. No more long-distance tests were conducted after this one.

4.4 Summary

This project aims to develop a low-cost solution for earth image acquisition, obtained from a HAB or a UAV. This work considered the development of a ground station dedicated to the reception of images provided by payload capturing data/information from active fires, and of the development of said payload. It consisted of the implementation of two antennas, LNA, PA, SDR, LoRa, GPS, and GNU Radio with some Python scripting to automate the transmission, reception, and processing of data.

The final work is best described with the following Fig. 4.54:

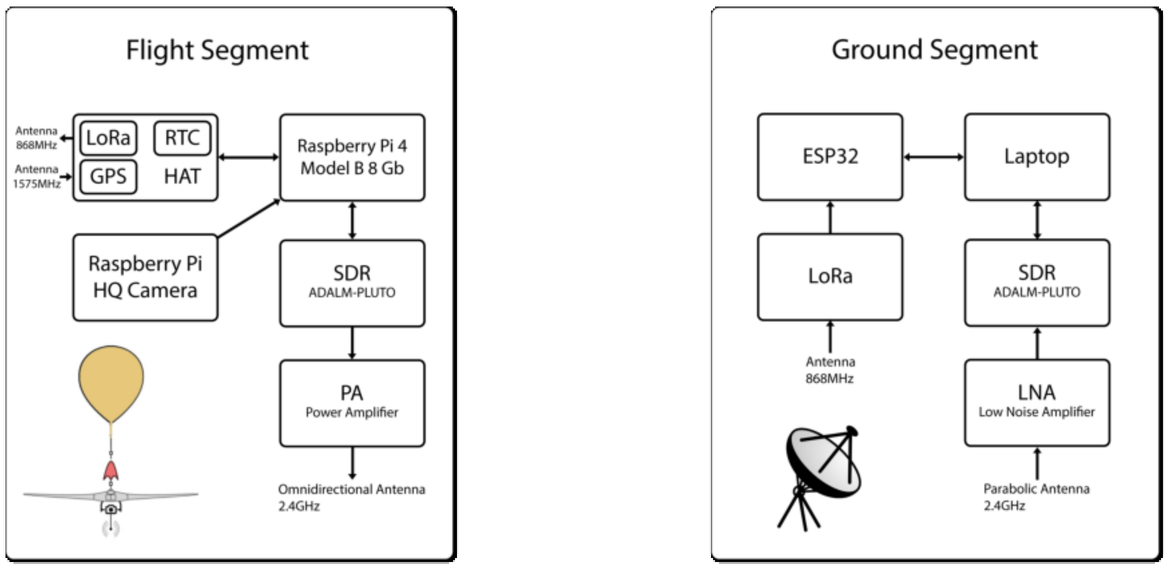
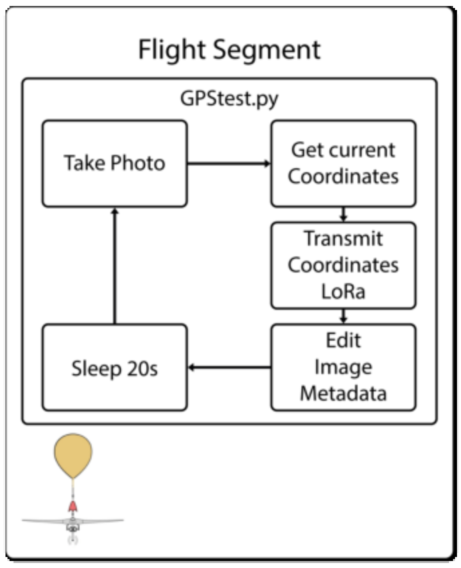
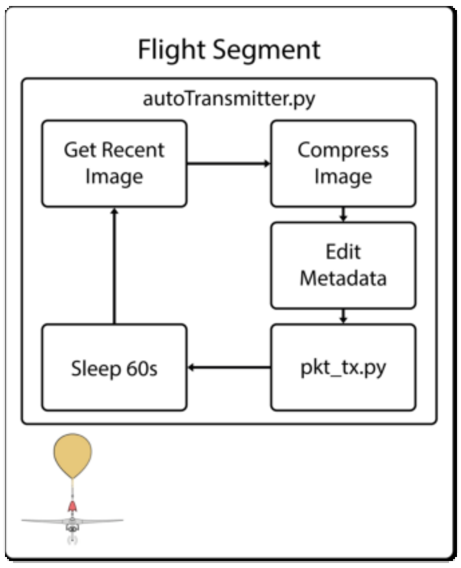


Figure 4.54: Block diagram for the hardware setup of the Flight and Ground Segments.

On the hardware diagrams, it is possible to see, on the flight segment, how the antenna, PA, SDR, Raspberry Pi and the HAT with the LoRa, GPS and RTC modules are connected and on the ground segment how the antenna, LNA, SDR, laptop, ESP32 and the LoRa module are connected.





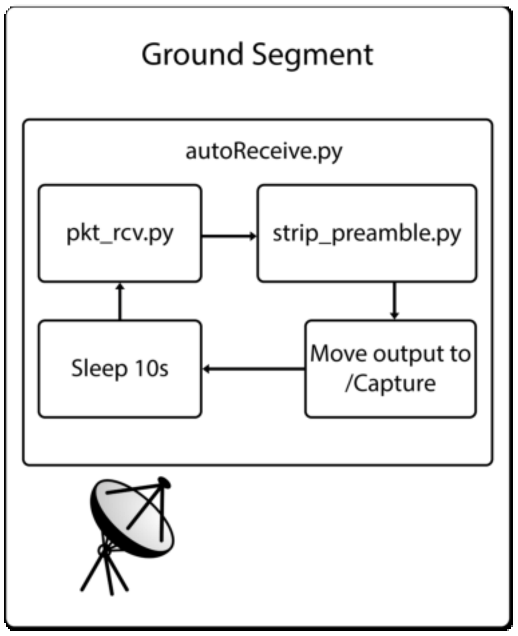


Figure 4.55: Block diagrams for the software setup of the Flight and Ground Segments.

The software diagrams, seen in Fig. 4.55, demonstrate the flow of the script. On the flight segment, GPStest.py runs at boot and takes photos periodically. It gets the current position, as soon as it takes the photo, sends it via LoRa, and edits the metadata of the image with it. In parallel, it starts autoTransmitter.py at boot and it finds the most recent image, reads the coordinates, saves it in temporary variables, compresses the image, and edits the metadata with the saved coordinates. It then runs pkt_tx.py, which is an edited script generated from a

GNU Radio flow graph and transmits the image with the SDR on the 2.4GHz band. It then sleeps for one minute, after finishing the transmission and it starts the cycle again.

The main challenges in the first version were the difficulty of finding documentation for this kind of application with this hardware and the limitations of the used protocol and the hardware as well. As for the second version the main challenge was to set up and learn how to use GNU Radio with the ADALM-PLUTO.

Regarding the antennas, one consists of an omnidirectional antenna tuned for a frequency of 2.4GHz, with a 6dBi gain and an adjustable angle between 0^{o} to 90^{o} for an easier setup on the payload. As for the other antenna, it is a grid dish directional antenna with a 24dBi gain and a beamwidth of 8^{o}. It was to be mounted on a pointing mechanism based on SatNOGS but, due to manufacturing delays, it was manually pointed to the target while performing the tests. It has a PA module with a gain of 20dB, in the payload, and a LNA on the ground segment with a gain of 8.1dB at 2.4GHz.

The signal acquisition and transmission is performed by two ADALM-PLUTO learning module SDR, on each end. The process consists of taking airborne photos with the geolocation in the metadata, while it sends that same data via LoRa so that the ground segment can track it, for better reception. It then picks the most recent image, compresses it, and transmits it via the SDR. It can send a 102KB image in 2 minutes, however, it fails with files bigger than 250 KB.

The Eye In the Sky project aims to provide aerial images of an active fire, to help fire-fighters better plan their attack, by giving them a different point of view and updating them on the progress of the fire, on the spot.

To improve the system a better SDR should be used and to improve the reliability of the transmission a TCP protocol should be implemented, even though it could impact the speed of the transmission.

Chapter 5

Conclusions and Future Work

The main objective of this project was the design of two low-cost Earth observation devices, each with different purposes. One is used for meteorological analysis, to provide weather forecasts in remote areas to aid with their agriculture, or by warning about incoming storms. For that, a ground station powered by a Raspberry Pi 3 Model B was developed to receive weather data from NOAA satellites, which can be decoded into images that can show the clouds from above. It also can do a multi-spectral analysis to determine the probability of rain from certain clouds. It uses TLE data, updated daily, to predict the passages of the NOAA Satellites. The entire setup is made of COTS materials and it is cheap to construct. It is an ideal solution as a weather station for remote locations where access to the internet is not available. It could warn the farmers about incoming storms or rain and thus, they can use that information to make better decisions about their agriculture. It is also a great solution as a backup on a Space Observatory Station, to provide weather data, in case it loses connection to the internet. Like that, it can continue its missions while the internet connection is restored. It is currently mounted in a NGO at Maputo. The project results were presented at IAC-22 – 73rd International Astronautical Congress – Paris, France. The total cost was 154.50€. The next step of this project is to receive HRPT which provides images with an even higher resolution. Since it is broadcast in the L band a directional antenna and a pointing mechanism to track the satellite are needed. without counting the enclosure.

Concerning the other main objective, the second low-cost Earth observation device goal is to assist firefighters by providing aerial images of ongoing wildfires and keep them updated on the fire progress, to better plan and be more efficient in their attack. A payload and a ground station were developed. The payload is able to report its position and transmit its own captured images. The payload is equipped with a Raspberry Pi 4 Model B 8GB, with an HQ camera attached on one side and a custom-made HAT to connect the GPS, LoRa and RTC modules to the Raspberry Pi GPIO. Connected via USB is a transceiver, ADALM-PLUTO SDR connected to an omnidirectional antenna, which is able to transmit the captured images in the 2.4GHz band. A LoRa module is used to transmit its position to the ground station so that the HAB can be tracked. Every image has, in its metadata, the geolocation and the time of when it was captured. A 20000mAh power bank powers all our electronic modules and the total weight is 1.4 kg. The ground station is made of a laptop with an ADALM-PLUTO SDR connected to a 2.4GHz directional antenna, as well as an ESP32 with a LoRa module connected via USB. It is capable of receiving a 102KB compressed JPG image size in 2 minutes and the payload consumes, on average, 1070mAh. The total cost of the payload and ground station is 1030.33€ without counting the price of the tracking mechanism, which, due to manufacturing delay, could not be tested with the current setup. For confidential reasons, its price cannot be disclosed, but it is far cheaper than the practiced prices on the market. The LoRa modules did not fulfill the requirements, failing to transmit a message over 25 meters. Despite this, a test was conducted where an image, with the payload location stored in the metadata, was received at a distance of 2.2 km. In the future, a better LoRa module should be used to ensure the transmission of telemetry data over long distances. A better SDR not only intended for learning purposes should be selected as well as a more powerful PA. To cut on the weight, a smaller and lighter power bank can be used, as well as reducing the length of the used cables. Besides, a TCP type protocol should be implemented, to make sure the received images are not corrupted during the transmission.

Bibliography

[1] DOPPLER. http://doppler.av.it.pt/.

[2] G. Pfotzer. History of the use of balloons in scientific experiments. *Space Science Reviews*, 13(2):199–242, June 1972.

[3] F. Stansbury Haydon and Tom D. Crouch. *Military Ballooning during the Early Civil War*. Johns Hopkins University Press, Baltimore, revised edition, July 2000.

[4] Craig S. Herbert. *Eyes of the Army: A Story about the Observation Balloon Service of World War I*. C.S. Herbert, 1986. Google-Books-ID: M9QNHQAACAAJ.

[5] France militaire. bataille de fleurus. | national air and space museum. https://airandspace.si.edu/collection-objects/france-militaire-bataille-de-fleurus/nasm_A20140500000.

[6] NOAA US Department of Commerce. Weather Balloons. https://www.weather.gov/bmx/kidscorner_weatherballoons. Publisher: NOAA's National Weather Service.

[7] Bureau of Public Affairs Department Of State. The Office of Electronic Information. The Launch of Sputnik, 1957, April 2008. Publisher: Department Of State. The Office of Electronic Information, Bureau of Public Affairs.

[8] October 1957 - Sputnik Launched - NASA. https://www.nasa.gov/image-article/october-1957-sputnik-launched/.

[9] Diogo Silva and José Silva. piNOAA: an independent daily Earth Observation service using a Raspbery-Pi data processing platform. IAC-22 – 73rd International Astronautical Congress - Paris, France, September 2022.

[10] Barry G. Evans, Paul T. Thompson, Giovanni E. Corazza, Alessandro Vanelli-Coralli, and Enzo Alberto Candreva. 1945–2010: 65 Years of Satellite History From Early Visions to Latest Missions. *Proceedings of the IEEE*, 99(11):1840–1857, November 2011. Conference Name: Proceedings of the IEEE.

[11] Matyas Dr. Palik and Máté Nagy. Brief History of UAV Development. *Repüléstudományi Közlemények*, 31:155–166, May 2019.

[12] Bander Alzahrani, Omar Sami Oubbati, Ahmed Barnawi, Mohammed Atiquzzaman, and Daniyal Alghazzawi. UAV assistance paradigm: State-of-the-art in applications and challenges. *Journal of Network and Computer Applications*, 166:102706, September 2020.

[13] Anabi Hilary Kelechi, Mohammed H. Alsharif, Damilare Abdulbasit Oluwole, Philip Achimugu, Osichinaka Ubadike, Jamel Nebhen, Atayero Aaron-Anthony, and Peerapong Uthansakul. The Recent Advancement in Unmanned Aerial Vehicle Tracking Antenna: A Review. *Sensors*, 21(16):5662, January 2021. Number: 16 Publisher: Multidisciplinary Digital Publishing Institute.

[14] Jonas Gustafsson. UAV Tracking Device using 2.4 GHz video transmitter.

[15] Kathleen Riesing. Orbit Determination from Two Line Element Sets of ISS-Deployed CubeSats.

[16] CelesTrak: NORAD Two-Line Element Set Format. `https://celestrak.org/NORAD/documentation/tle-fmt.php`.

[17] NOAA KLM USER'S GUIDE Section 4.2. `https://web.archive.org/web/20130214000432/http://www2.ncdc.noaa.gov/docs/klm/html/c4/sec4-2.htm`, February 2013.

[18] Noaa user guide. `https://noaasis.noaa.gov/NOAASIS/pubs/Users_Guide-Building_Receive_Stations_March_2009.pdf`.

[19] Quadrifilar helicoidal antenna - Javascript on-line calculator. `https://jcoppens.com/ant/qfh/calc.en.php`.

[20] Quadrifilar helicoidal antenna - Simulation of several versions. `http://jcoppens.com/ant/qfh/sim.en.php`.

[21] Quadrifilar helicoidal antenna diagram. `http://jcoppens.com/ant/qfh/img/coax/coaxconn1.jpeg`.

[22] Nooelec - Nooelec NESDR Mini 2 USB RTL-SDR and ADS-B Receiver Set, RTL2832U and R820T2 Tuner, w/ Antenna. MCX Input. Low-Cost Software Defined Radio Compatible with Many SDR Software Packages, ESD-Safe - SDR Receivers - Software Defined Radio. `https://www.nooelec.com/store/sdr/sdr-receivers/nesdr-mini-2.html`.

[23] Nooelec - Nooelec SAWbird+ NOAA - Premium SAW Filter & Cascaded Ultra-Low Noise LNA Module for NOAA Applications. 137MHz Center Frequency. `https://www.nooelec.com/store/sawbird-plus-noaa-308.html`.

[24] Nooelec. *SAWbird+ NOAA Low Noise Amplifier (LNA)*, 6 2022. `https://www.nooelec.com/store/sdr/sdr-addons/sawbird/sawbird-plus-noaa-308.html`.

[25] Raspberry Pi Foundation. *Raspberry Pi 3 Model B+*, 2018. `https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf`.

[26] Dr Paul Brewer. rtlsdr-automated-wxsat-capture. `https://github.com/DrPaulBrewer/rtlsdr-automated-wxsat-capture`, September 2023. original-date: 2014-09-26T04:36:14Z.

[27] Gpredict: Free, Real-Time Satellite Tracking and Orbit Prediction Software. `http://gpredict.oz9aec.net/`.

[28] Gqrx SDR – Open source software defined radio by Alexandru Csete OZ9AEC. `https://gqrx.dk/`.

[29] noaa-apt image decoder. `https://noaa-apt.mbernardi.com.ar/`.

[30] PREDICT - A Satellite Tracking/Orbital Prediction Program. `https://www.qsl.net/kd2bd/predict.html`.

[31] Rtl_fm Guide: Updates for rtl_fm overhaul. `http://kmkeen.com/rtl-demod-guide/`.

[32] Ulrich Klauer. chirlu/sox. `https://github.com/chirlu/sox`, October 2023. original-date: 2012-09-09T20:06:01Z.

[33] WXtoIMG. `https://wxtoimgrestored.xyz/`.

[34] Wxtoimg user guide. `https://usradioguy.com/wp-content/uploads/2020/05/wxtoimgcommand-line.pdf`.

[35] Celestrack noaa tle. `http://celestrak.org/NORAD/elements/gp.php?GROUP=noaa&FORMAT=tle`.

[36] Dio76453/piNOAA: Automatic image acquisition from NOAA satellites. `https://github.com/Dio76453/piNOAA`.

[37] Eye In The Sky – Using High-Altitude Balloons for Decision Support in Wildfire Operations. `https://adai.pt/eyeinthesky/`.

[38] Portáteis VHF / UHF • TH-D74E Especificações • KENWOOD Portugal. https://www.kenwood.pt/comm/ar/portateis_vhf_uhf/TH-D74E/?view=details.

[39] Raspberry Pi Foundation. *Raspberry Pi 4 Model B*, 2019. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf.

[40] Raspberry Pi Ltd. Raspberry Pi High Quality Camera. https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/.

[41] Josef Stevanus. python-ax25. https://github.com/josefmtd/python-ax25, August 2023. original-date: 2020-01-28T06:43:21Z.

[42] Configuring an AX.25 interface for TCP/IP. https://tldp.org/HOWTO/AX25-HOWTO/x1194.html.

[43] Customizing the Pluto configuration [Analog Devices Wiki]. https://wiki.analog.com/university/tools/pluto/users/customizing.

[44] CN0417 Evaluation Board Guide [Analog Devices Wiki]. https://wiki.analog.com/resources/eval/user-guides/circuits-from-the-lab/cn0417.

[45] OPA Design. *WIDE BAND LOW NOISE AMPLIFIER*, 2019. https://www.passion-radio.com/index.php?controller=attachment&id_attachment=464.

[46] SatNOGS. https://satnogs.org/.

[47] Módulo GPS NEO-7M - UART c/ antena cerâmica. https://www.botnroll.com/pt/gps-gnss/3688-m-dulo-gps-neo-7m-uart-c-antena-cer-mica.html.

[48] Sx1276 datasheet. https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf.

[49] LoRa Radio Module - 868MHz. https://www.dfrobot.com/product-1671.html.

[50] DS3231 RTC Module | RTC | PTR005161. https://www.ptrobotics.com/rtc/5161-ds3231-rtc-module.html.

[51] File transfer using Packet and BPSK - GNU Radio. https://wiki.gnuradio.org/index.php?title=File_transfer_using_Packet_and_BPSK.

[52] duggabe. gr-control. https://github.com/duggabe/gr-control, June 2023. original-date: 2020-12-23T17:26:04Z.