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

TREBALL DE FI DE GRAU

TFG TITLE: Development of a drone-based miniaturized Flexible Microwave Payload (FMPL) for GNSS-Reflectometry and L-band radiometry

DEGREE: Doble titulació de grau en Enginyeria de Sistemes Aeroespacials i Enginyeria de Sistemes de Telecomunicació

AUTHOR: Andreu Mas Viñolas

DIRECTOR: Hyuk Park

TUTOR: Adrián Pérez

DATE: October 15, 2022

Título: Desenvolupament d'una Càrrega Útil de Microones Miniaturizada i Flexible (FMPL) per drons amb reflectometria de GNSS (GNSS-R) i radiometria en banda L
Autor: Andreu Mas Viñolas

Director: Hyuk Park

Supervisor: Adrián Pérez

Fecha: 15 d'octubre de 2022

Resumen

Aquest projecte s'ha desenvolupat en col·laboració amb el NanosatLab UPC, que desenvolupa CubeSats per fins educacionals i científics, i demostració de noves tecnologies en òrbita. Més concretament, el laboratori està centrat en els sistemes de teledetecció.
En els últims anys, el NanosatLab UPC ha estat desenvolupant la Flexible Microwave Payload (FMPL), la integració de diferents equipaments de teledetecció remota amb microones en un sol sistema: reflectometria amb senyals de Global Navigation Satellite System (GNSS) (GNSS-R) i radiometria de microones (MWR) en banda L. Al 2022, la segona versió d'aquest sistema, el FMPL-2, ja està en òrbita a bord del CubeSat ^{3}Cat5 i ha aportat dades científiques molt valuoses sobre el clima de la terra i l'evolució del canvi climàtic. La primera versió, FMPL-1, es llençarà en els propers mesos a bord del CubeSat ^{3}Cat4. La tercera versió, l'FMPL-3, ja està preparada pel llançament a bord del CubeSat GNSSaS. Des de l'espai, FMPL ha demostrat ser una eina molt últil per estudiar el canvi climàtic.
L'objectiu d'aquest treball és dissenyar, construïr i provar el primer FMPL per drones, el FMPL-D. Aquesta nova plataforma servirà per poder avaluar les noves versions de FMPL. També serà una eina molt útil per a estudiar les característiques del sòl, aigua, gel i/o vegetació localment i amb una resolució espacial molt major que la que es pot obtenir des d'un satèl·lit.
Els resultats que es presenten en aquesta tesi posen en perspectiva la complexitat d'aquests sistemes. En primer lloc, en els resultats del radiòmetre s'ha observat un efecte de distorsió i destrucció de les dades obtingudes per culpa de les interferències de radiofreqüència rebudes durant les campanyes de mesura, posant en evidència la necessitat de sistemes de detecció i mitigació d'interferència per a missions d'observació terrrestre. Pel que fa a l'instrument de reflectometria GNSS, es van realitzar múltiples vols en els que es van recollir grans quantitats de dades, el processament de les quals encara està en procés. Els resultats preliminars indiquen unes bones característiques de la cadena de radiofreqüència.
Aquest Treball de Fi de Grau (TFG) és la primera versió de la FMPL-D, que ha culminat en la primera versió del sistema i en moltes lliçons apreses.

Title : Development of a drone-based miniaturized Flexible Microwave Payload (FMPL) for GNSS-Reflectometry and L-band radiometry
Author: Andreu Mas Viñolas

Advisor: Hyuk Park

Supervisor: Adrián Pérez

Date: October 15, 2022

Overview

This project has been developed in collaboration with the NanosatLab UPC, which develops CubeSats for educational and scientific purposes and in-orbit technology demonstration. More specifically, the laboratory is focused on remote sensing systems.
In recent years, the NanosatLab UPC has been developing the Flexible Microwave Payload (FMPL), the integration of different microwave remote sensing equipment in a single system: reflectometry Global Navigation Satellite System (GNSS) signals (GNSS-R) and microwave radiometry (MWR) in L-band. In 2022, the second version of this system, FMPL-2, is in orbit on board the CubeSat ^{3}Cat5, which has provided precious scientific data on the climate of the earth and the evolution of climate change. The first version, FMPL-1, will be launched in the coming months aboard CubeSat ^{3}Cat4. The third version, FMPL-3, is now ready for launch on board the CubeSat GNSSaS. From space, FMPL has proven to be a very useful tool for studying climate change.
This work aims to design, build and test the first FMPL for drones, the FMPL-D. This new platform will be used to evaluate new versions of FMPL. It will also be a valuable tool to study the characteristics of soil, water, ice and vegetation locally and with a spatial resolution much greater than that which can be obtained from a satellite.
The results presented in this thesis put the complexity of these systems into perspective. Firstly, in the results of the radiometer, an effect of distortion and destruction of the data obtained due to the radio frequency interference received during the measurement campaigns has been observed, highlighting the need for detection and mitigation systems interference for ground observation missions. For the GNSS reflectometry instrument, multiple flights were conducted in which large amounts of data were collected, the processing of which is still in progress. Preliminary results indicate good characteristics of the radio frequency chain.
This Final Degree Project (TFG) is the first version of the FMPL-D, culminating in the system's first version and many lessons learned.

To my family, and to the NanosatLab friends. Thank you.

CONTENTS

Acronyms . . . 1

CHAPTER 1. Introduction . . . 3

1.1. Nanosatellites and the CubeSat standard . . . 3
1.1.1. Nanosatellites . . . 3
1.1.2. The CubeSat standard . . . 3

1.2. UPC NanoSatLab missions . . . 4
1.2.1. ^{3}Cat1 . . . 5
1.2.2. ^{3}Cat2 . . . 5
1.2.3. ^{3}Cat4 . . . 6
1.2.4. ^{3}Cat5 - FSSCat . . . 6
1.2.5. ^{3}Cat6 - RITA . . . 6

1.3. Passive Microwave Remote Sensing . . . 7

1.4. Unmanned Aerial Vehicles (UAV) . . . 7

1.5. Motivations and objectives . . . 8

1.6. Methodology . . . 8

CHAPTER 2. The Flexible Microwave Payload for Drones . . . 11

2.1. Introduction . . . 11

2.2. Global Navigation Satellite System (GNSS) and GNSS-Reflectometry . . . 11

2.3. L-Band Microwave Radiometry (MWR) . . . 17
2.3.1. Types of radiometers . . . 17

CHAPTER 3. Hardware design . . . 19

3.1. Introduction . . . 19

3.2. Antennas . . . 20
3.2.1. GNSS-R . . . 20
3.2.2. L-Band Radiometry . . . 22

3.3. Front-end . . . 23
3.3.1. GNSS-R . . . 24

3.3.2. L-Band Radiometry . . . 25

3.4. Software-Defined Radio (SDR) . . . 28

3.5. Computer and storage . . . 30

3.6. Power management . . . 30

3.6.1. Power source of the drone . . . 30

3.6.2. Power source of the payload . . . 31

3.7. Integration to the drone . . . 32

3.7.1. Version 1 . . . 32

3.7.2. Version 2 . . . 33

3.8. Drone platform . . . 34

3.8.1. Features . . . 35

3.8.2. Lift studies . . . 35

3.8.3. Condor Beta drone . . . 38

CHAPTER 4. Acquisition software design . . . 41

4.1. Introduction . . . 41

4.2. Buffer structure . . . 42

4.3. Positioning data . . . 43

4.4. GNSS-R . . . 44

4.4.1. SDR Parameters . . . 44

4.4.2. Code . . . 45

4.4.3. Data generation . . . 45

4.5. L-band Microwave Radiometry (MWR) . . . 46

4.5.1. SDR Parameters . . . 46

4.5.2. Code . . . 47

4.5.3. Data generation . . . 48

CHAPTER 5. Processing software design . . . 51

5.1. Introduction . . . 51

5.2. Positioning data . . . 51

5.3. GNSS-R . . . 52

5.3.1. Raw data processing. . . . 52

5.3.2. Correlation with the position . . . 60

5.3.3. Computation of the specular reflections . . . 60
5.3.4. Plotting the results in QGIS . . . 61

5.4. L-band radiometry . . . 62
5.4.1. Codes . . . 62

CHAPTER 6. Measurement campaign and results . . . 65

6.1. Introduction . . . 65
6.1.1. Measuring campaigns carried out . . . 66

6.2. L-Band Radiometry . . . 68
6.2.1. Campaign . . . 68
6.2.2. Results . . . 69

6.3. GNSS-R . . . 71
6.3.1. Campaign . . . 71
6.3.2. Results . . . 71

Conclusions and lessons learned . . . 75

6.4. Future work . . . 76

Bibliography . . . 77

APPENDIX A. Schematics of the NADS Bottom PCB . . . 81

LIST OF FIGURES

1.1 Number of satellites launched into orbit since 1960 [1] . . . 3
1.2 Standard CubeSats sizes. [2] . . . 4
1.3 Number of nanosatellites launched by organizations [3]. . . . 4
1.4 ^{3}Cat1 CubeSat. . . . 5
1.5 ^{3}Cat2 CubeSat. . . . 5
1.6 ^{3}Cat4 CubeSat. . . . 6
1.7 FSSCat CubeSats. . . . 6
1.8 V-Model template followed along the project. . . . 8
1.9 Concept of the Flexible Microwave Payload for Drones (FMPL-D) system. . . . 9

2.1 Illustration of a simplified two-dimensional GNSS situation . . . 12
2.2 Constellation of a BPSK modulation. . . . 13
2.3 Spread spectrum frequency distribution . . . 13
2.4 Output of the correlation operation . . . 14
2.5 GNSS-R basic principle [15]. . . . 15
2.6 Noise-free simulated DDM. . . . 16
2.7 Two different GNSS-R techniques. . . . 16

3.1 Diagram of the GNSS-R diagram . . . 19
3.2 Diagram of the radiometer hardware. . . . 19
3.3 Bias-tee. . . . 20
3.4 Setup to measure the LHCP patch antennas. . . . 21
3.5 Modifying an off-the-shelf GPS antenna to be LHCP. . . . 21
3.6 GPS antennas. . . . 22
3.7 S_{11} parameter of the LHCP antenna . . . 22
3.8 GPS module used for testing: U-blox LEA-6S. . . . 22
3.9 Testing the LHCP antenna. . . . 23
3.10 MWR antenna. . . . 23
3.11 Cheaper and lighter bias-tees. . . . 24
3.12 Plot of the resulting sinusoidal signal after having passed through the cheap bias-tees. . . . 24
3.13 Total Power Radiometer (TPR) system. . . . 25
3.14 Testing the NADS bottom PCB version 1.2 (QM). . . . 26
3.15 Spectrum when the Hot Load (HL) is selected. . . . 27
3.16 Spectrum when the Active Cold Load (ACL) is selected. . . . 27
3.17 Spectrum when the antenna is selected. . . . 27
3.18 RTL-SDR USB dongle . . . 29
3.19 ADALM-Pluto SDR . . . 29
3.20 Simplified block diagram of ADALM-Pluto's receiving chain. . . . 30
3.21 Raspberry Pi zero W 2, the processing system for the FMPL-D. . . . 30
3.22 DC/DC converter. Input: 11.1V, output: 5V. . . . 32
3.23 First version of the interface structure. . . . 32
3.24 First version of the interface structure attached to the drone. . . . 33
3.25 Second version of the interface structure. . . . 33

3.26 Second version of the interface structure, with GNSS-R and LoRa integrated. . 34
3.27 3D Robotics Iris . . . 35
3.28 Pixhawk 1 and 3DR GPS+compass module . . . 36
3.29 Thrust theoretical study. . . . 37
3.30 Results of the static lift test. . . . 37
3.31 Flight test to determine the drone's thrust. . . . 38
3.32 Condor Beta. . . . 39

4.1 Flowchart of the common acquisition software structures for GNSS-R and MWR systems. . . . 41
4.2 Receiving a test signal. . . . 42
4.3 ADALM-Pluto, two channels buffer structure. . . . 43
4.4 PRN sequence sampling with different sampling rates. . . . 45
4.5 Flowchart of the GNSS-R acquisition code. . . . 46
4.6 Flowchart of the MWR acquisition code. . . . 48
4.7 Radiometer's time diagram. . . . 49

5.1 Basic UBX frame. . . . 51
5.2 Skyplot at the moment of the flight. . . . 53
5.3 Compute the angle of maximum Doppler. . . . 55
5.4 Parallel Code Phase Search (PCPS) implementation. . . . 56
5.5 DDM of a reflected signal. . . . 56
5.6 DDM of a reflected signal. . . . 57
5.7 Flowchart of the main GNSS processing system. . . . 58
5.8 DDM of a signal with very low CN0. . . . 59
5.9 DDM of a signal with low CN0. . . . 59
5.10 DDM of a signal with the minimum CN0. . . . 59
5.11 DDM of a signal with a high CN0. . . . 60
5.12 Comparison between different Earth altitude reference frames [23]. . . . 61

6.1 Selected area for the campaign. In purple is the selected spot. [24] . . . 66
6.2 First measuring campaign: Iris ready for take-off. . . . 66
6.3 First measuring campaign: the team. . . . 67
6.4 Second measuring campaign. . . . 67
6.5 The team of the 3rd measuring campaign with the Condor Beta drone with the payload integrated. . . . 68
6.6 Path of the drone in the radiometer flight. . . . 69
6.7 MWR results from the first campaign. . . . 69
6.8 Result of the static test, zooming into the radiometer antenna power. . . . 70
6.9 Spatial resolution as a function of the altitude and incidence angle. . . . 72
6.10 Path of the drone in the GNSS-R flight. . . . 72
6.11 Results of the GNSS-R measuring campaign. . . . 73

A.1 Schematic of the NADS Bottom PCB. . . . 81
A.2 PCB design of the NADS Bottom. . . . 81
A.3 3D view of the NADS Bottom PCB. . . . 82

LIST OF TABLES

1.1 Classification and remote sensing uses of the microwave spectrum [6]. 7

3.1 Summary table of the weights of the integrated payloads 34
3.2 Results of the flight test. 38

4.1 SDR configuration for GNSS-R acquisition. 44
4.2 SDR configuration for MWR acquisition. 47
4.3 TPR truth table. 48

ACRONYMS

ACL Active Cold Load. xiii, 25–28, 47, 48, 63, 69

ADC Analog to Digital Converter. 18, 28, 42, 45, 47

AGC Automatic Gain Control. 29

BPSK Binary Phase Shift Keying. xiii, 12, 13

CDMA Code Division Multiple Access. 13

cGNSS-R conventional GNSS-R. 15, 16, 28, 29, 44

CN0 Carrier-to-Noise ratio. 57, 60

CTR Controlled Traffic Region. 65

DC Direct Current. 20

DDM Delay-Doppler map. xiii, xiv, 15–17, 53–60

DR Dicke Radiometer. 18

ECEF Earth-Centered Earth-Fixed. 61

FFT Full Functional Test. 65

FMPL Flexible Microwave Payload. iii, v, 5, 6, 8, 18

FMPL-D Flexible Microwave Payload for Drones. xiii, 5, 8, 9, 11, 22–25, 28, 30, 32, 33, 35, 39, 51, 55, 57, 69, 70, 75, 76

FPGA Field-Programmable Gate Array. 29–31

GNSS Global Navigation Satellite System. iii, v, ix, xiii, xiv, 11, 12, 14, 15, 20, 21, 24, 29, 52, 58, 61, 71

GNSS-R GNSS-Reflectometry. iii, v, x, xiii–xv, 6, 7, 9, 11, 14–16, 19, 20, 24, 29, 31–33, 41, 42, 44, 46, 47, 51–53, 57, 60–63, 71–73, 75, 76

GPIO General Purpose Input/Output. 25, 30

GPS Global Positioning System. xiii, 11–13, 21, 22, 52, 58, 60, 61, 63

HL Hot Load. xiii, 25–28, 47, 48, 63, 69

iGNSS-R interferometric GNSS-R. 16, 29

LEO Low Earth Orbit. 14

LHCP Left Hand Circular Polarization. xiii, 20–23

LNA Low Noise Amplifier. 20, 21, 23–25

LO Local Oscillator. 28, 44, 46, 47

MEO Medium Earth Orbit. 12, 14, 52

MWR Microwave Radiometry. ix, x, xiii–xv, 6, 7, 9, 17, 18, 20, 22, 23, 25, 28, 29, 31–33, 41, 46–48, 51, 63, 68, 69, 71, 75, 76

NADS Nadir Antenna and Deployment Subsystem. 25, 31, 46

NF Noise Figure. 20

NIR Noise Injection Radiometer. 18

PCPS Parallel Code Phase Search. xiv, 55, 56

PRN Pseudo-Random Noise. 13–15

QM Qualification Model. xiii, 18, 25, 26

RF Radio Frequency. 18, 20, 23–25

RFI Radio Frequency Interference. 62, 65, 69, 70

RHCP Right Hand Circular Polarization. 12, 20, 21

RITA Remote sensing and Interference detector with radiomeTry and vegetation Analysi. 22

RMS Root Mean Square. 16

RPi Raspberry Pi. 30

SDR Software-Defined Radio. x, xiii, xv, 19, 23–25, 28–30, 32, 42, 44–47, 54, 56, 62

SMIGOL Soil Moisture Interference-pattern GNSS Observations at L-band. 14

SNR Signal to Noise Ratio. 57, 58

TPR Total Power Radiometer. xiii, xv, 18, 25–28, 30, 46–48

CHAPTER 1. INTRODUCTION

1.1. Nanosatellites and the CubeSat standard

1.1.1. Nanosatellites

The space industry was born during the Cold War when in 1957, the Soviet Union launched the first artificial satellite in Low Earth Orbit (LEO), the Spútnik 1. Since then, thousands of satellites with very diverse proposes have been launched into space.

Fig. 1.1 depicts the number of satellite launches from 1957 to our days, depending on their mass. It can be seen that light (<10kg) satellites started booming around the 2010s. Nowadays, small satellites can perform tasks that some years ago could only be done with heavy and expensive hardware.

The main reason behind the increase in small satellite launches is the definition of the CubeSat standard (1.1.2.).

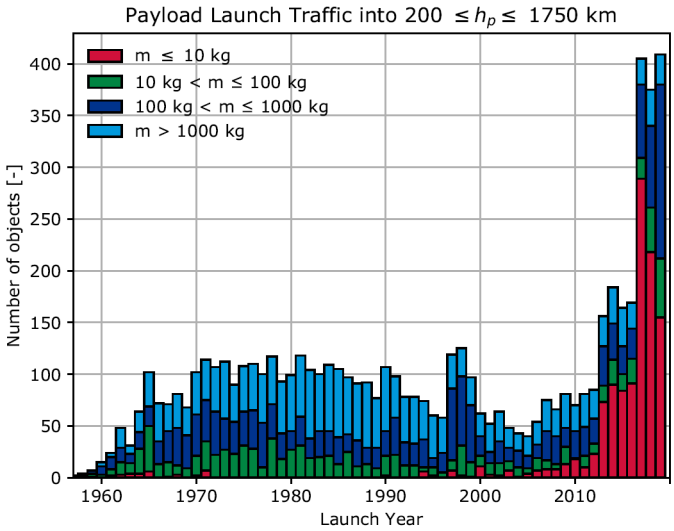


Figure 1.1: Number of satellites launched into orbit since 1960 [1]

1.1.2. The CubeSat standard

CubeSats are a class of nanosatellites with standard size and form factor to be launched inside a specific container. These containers are usually launched together with a big satellite as a secondary payload (piggyback), and the launcher authority offers spots for CubeSats in the containers. This way, a small satellite's launch cost is dramatically reduced. Due to the small size of the CubeSats, the development time is also very short compared to more extensive missions, taking between 1 and 3 years (average). Moreover, since the form factor is already defined, there are available off-the-shelf components already tested in orbit, reducing CubeSats' complexity and development time. A CubeSat of one unit, or "1U", measures 10x10x10 cm and weighs no more than 2 Kg. CubeSats are

scalable, meaning that bigger satellites can be built by adding several standard units. Not all form factors are permitted due to the container's requirements; the most common ones are 1U, 1.5U, 2U, 3U, 6U and 12U (Fig. 1.2).

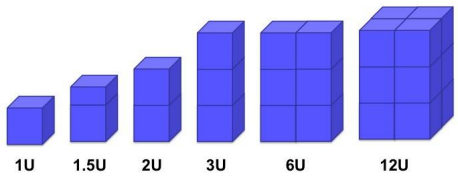


Figure 1.2: Standard CubeSats sizes. [2]

In 1999, two professors of Stanford University, Jordi Puig-Suari and Bob Twiggs, defined the CubeSat reference, and as years passed, it became a standard. Thanks to the dramatic reduction of the development, production and launch costs, universities worldwide started using the CubeSat standard for educational, science, and technology in-orbit demonstrating purposes. In recent years, private companies have begun developing CubeSats for commercial purposes. Figure 1.3 shows the nanosatellite launches by organizations. In this context, the NanoSatLab was born in 2007 to design remote-sensing payloads and test them in orbit using the CubeSat platform.

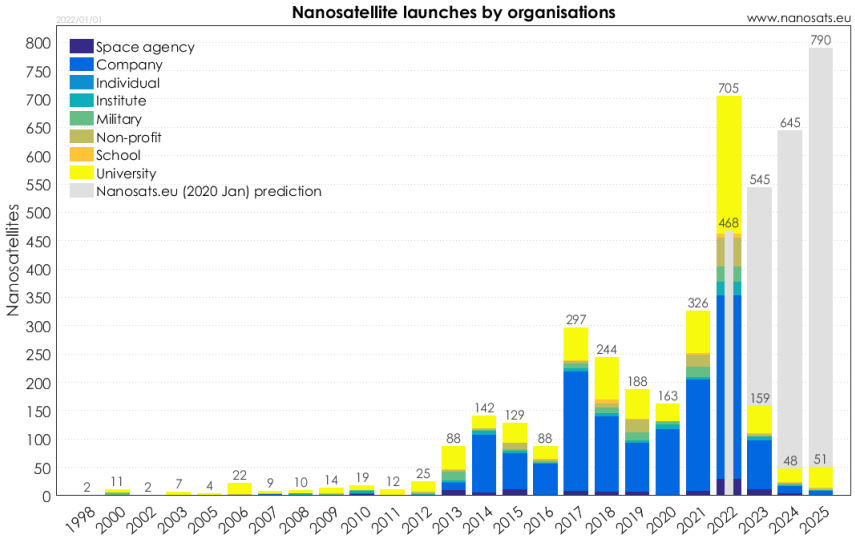


Figure 1.3: Number of nanosatellites launched by organizations [3].

1.2. UPC NanoSatLab missions

The Nano-Satellite and Payload Laboratory (UPC NanoSatLab) is a cross-departmental initiative belonging to the Barcelona School of Telecommunications Engineering. The lab

is focused on developing nano-satellite missions, specifically on exploring innovative small spacecraft system concepts and developing and integrating subsystems for Earth Observation. Until now (2022), the lab has already launched 4 CubeSats.

The NanosatLab missions have in common that they carry Earth-observation and remote-sensing payloads. ^{3}Cat4 and ^{3}Cat5 carry a payload called Flexible Microwave Payload (FMPL), and the team is already working on the new version of this payload: FMPL-3. The drone's version of this payload is developed in this project, the Flexible Microwave Payload for Drones (FMPL-D).

This final degree thesis uses a lot of knowledge and even components of some of the NanoSatLab's missions. The different missions that the laboratory has been working towards can be found below.

1.2.1. ^{3}Cat1

^{3}Cat1 (Fig. 1.4) is the first satellite developed in Catalonia and the first in the NanosatLab's ^{3}Cat series, and it was launched in 2018. It integrates 7 payloads in 1U, whose objectives are education, in-orbit technology demonstrators, and scientific experiments (further details can be found in [4]).

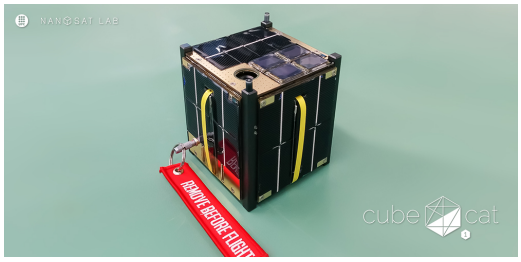


Figure 1.4: ^{3}Cat1 CubeSat.

1.2.2. ^{3}Cat2

^{3}Cat2 (Fig. 1.5) is the second satellite of the series, although it was launched in 2016. It carries 4 payloads in a 6U CubeSat, whose objectives are: in-orbit technology demonstration, education and science.

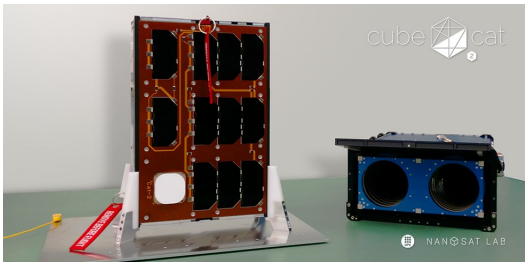


Figure 1.5: ^{3}Cat2 CubeSat.

1.2.3. ^{3}Cat4

^{3}Cat4 (Fig. 1.6) is a one-unit (1U) CubeSat developed in NanoSatLab, accepted for the ESA's Fly Your Satellite! programme. The satellite has on board the first version of the FMPL, the FMPL-1, an Earth Observation payload that features GNSS-R and an L-Band MWR, as well as an Automatic Identification Services (AIS). A PCB design specifically for the L-Band radiometer of this satellite is used in this thesis. The satellite will be launched in 2023 in Ariane's 6 maiden flight.

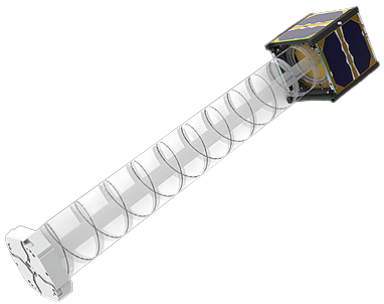


Figure 1.6: ^{3}Cat4 CubeSat.

1.2.4. ^{3}Cat5 - FSSCat

FSSCat (^{3}Cat5) is the winner of the 2017 *Copernicus Master ESA Small Satellite Challenge S3̂ and Overall Winner*. It is a mission consisting of two six-unit (6U) CubeSats in support of the Copernicus Land and Marine Environment services. Their payload, developed in NanoSatLab UPC, is the second version of the FMPL (FMPL-2), which combines a GNSS-Reflectometry (GNSS-R) and an L-Band radiometer in a single instrument. The satellites were launched in 2020 and provided successful results [5].

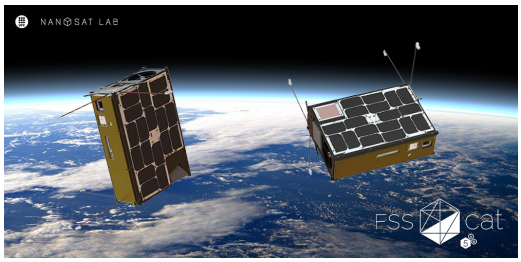


Figure 1.7: FSSCat CubeSats.

1.2.5. ^{3}Cat6 - RITA

RITA is a 1U payload whose mission is to study global warming and the state of the world's vegetation. The payload consists of an L-Band MWR to study soil's humidity, a hyperspectral camera to acquire vegetation measurements, and radio-frequency interference detection and characterization. There will also be a novel multi-level acquisition strategy from ground sensors using LoRa protocol, an in-orbit technology demonstrator. The payload occupies 1U of a 3U CubeSat called AlainSat-1.

1.3. Passive Microwave Remote Sensing

Remote sensing encompasses all the techniques that allow measurements to be made at a certain distance from them, which is achieved by analysing the signals coming from the targets. Remote sensing techniques can be classified according to the origin of the signals:

- Active remote sensing techniques are based on sending a signal towards the target. The measurement information is in the reflected signal, which shall be processed. Radars are an example of active remote sensing systems.
- Passive remote sensing instruments do not radiate new signals to the target; instead, they receive and analyze signals coming from either natural sources (as MWR) or signals of opportunity (as GNSS-R).

Since passive sensors do not transmit signals, the power consumption is significantly reduced, allowing small payloads to achieve long-range remote sensing with great detail. Passive microwave remote sensing focuses on analysing the signals present in the microwave part of the spectrum, approximately from 1cm to 1m in wavelength. The microwave spectrum is divided into a range of frequency bands, each represented with a letter, and each band is more suitable for each application. Table 1.1 depicts the classification of the microwave band and relates it with its more common remote sensing application.

Band	Wavelength (cm)	Frequency (GHz)	Examples of ground applications
P (UHF)	100 - 30	0.3 - 1	Archeology, buried bodies
L	30 - 15	1 - 2	Soil moisture
S	15 - 7.5	2 - 4	Upper soil, vegetation structure
C	7.5 - 3.8	4 - 8	Vegetation structure, ground movement
X	3.8 - 2.4	4 - 12.5	Canopy, base soil (subsidence, landslides)
Ku	2.4 - 1.67	12.5 - 18	Bare soil (subsidences, landslides)

Table 1.1: Classification and remote sensing uses of the microwave spectrum [6].

The primary interest of this project is to study the soil moisture; therefore, the systems operate in L-Band. This band present several key advantages for spacecraft missions.

- Very low atmospheric losses.
- Day and night coverage. These systems do not require visible light to work.
- Not affected by clouds nor rain.

1.4. Unmanned Aerial Vehicles (UAV)

UAV is the generic term for vehicles that fly without a human pilot on board. The term drone is the most common term to refer to them.

Thanks to the technological advancements in miniaturization, electronics and algorithms, in the last 10 years, drones have rapidly evolved, and their uses moved from the military

domain to commercial interest. The market evolved during the 2010s, and today the main uses for civilian drones are agriculture, construction and mining, insurance, and media and telecommunications (according to [7]).

Drones are small, fast, and easily adaptable for any payload. This thesis uses a drone as a platform for the new FMPL-D. The drone will be a flexible platform to test these systems fast, cheap, and in a real environment. In addition, it will be a platform for scientific purposes, permitting the study of the soil without depending on a satellite's revisit time.

1.5. Motivations and objectives

From space, remote sensing systems such as radiometry and GNSS-R have demonstrated impressive results in measuring soil humidity, soil reflectivity, ocean altimetry and oceanic wind speed ([8], [9], [10]). Although remote sensing from space has proven an impressive spatial resolution, it is still too high to detect field changes since it strictly depends on the aircraft or spacecraft's altitude. A typical altitude of a CubeSat with GNSS-R is 600km, which usually provides a footprint of hundreds of squared kilometres [11]. A drone-based remote sensing payload can cover small areas with better spatial resolution.

This project aims to design, develop and test a version of the Flexible Microwave Payload (FMPL) for drones, FMPL-D from now on. The main objective is that the drone platform becomes a test bench for future FMPL systems for coming spacecrafts. This project aspires to develop the first version of the FMPL-D that will be improved in the future.

1.6. Methodology

The result of this project will be a complex functional system composed of several subsystems and units. To deal with it, the development has been structured based on a V-Model such as the one depicted in Fig. 1.8.

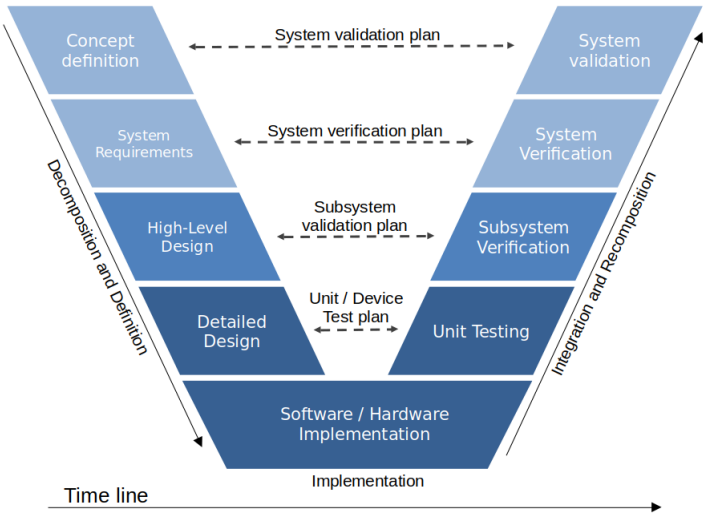


Figure 1.8: V-Model template followed along the project.

First, the concept of operations of the overall system has been defined to understand the project's objectives and the proper requirements clearly. When functional, the system shall be able to create a soil moisture map of the area where the drone flies using GNSS-R and/or Microwave Radiometry (MWR). The system concept is summarized in Fig. 1.9.

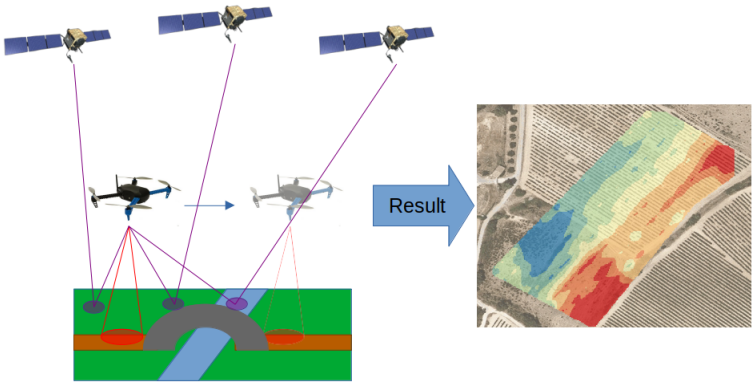


Figure 1.9: Concept of the Flexible Microwave Payload for Drones (FMPL-D) system.

The system-level requirements are:

- REQ-F-10. The FMPL-D shall make GNSS-R measurements.
- REQ-F-20. The FMPL-D shall make MWR measurements.
- REQ-F-30. The FMPL-D system may integrate both GNSS-R and MWR in the same payload.
- REQ-F-40. The FMPL-D structure shall be mounted on a drone.
- REQ-F-50. The result of the flight campaigns shall be a soil moisture map.
- REQ-F-60. The FMPL-D shall store all the campaign measurements.
- REQ-F-70. The system may be capable of operating for at least 15 minutes.

After having a clear idea about the system and the requirements, the high-level design started. This phase is where the design of the system takes place. Segmenting the system into ”subsystems” and defining how they would interact with each other. After that, a more detailed design of the subsystems was done. The hardware was selected and studied if the selected pieces could work together at the system level.

After the hardware design was completed, the implementation stage followed. During the implementation, the necessary boards were procured when applicable, and the structure was manufactured in-house using CAD models and a 3D printer.

Each part of the system was tested individually. Then, the system was integrated, and a full-functional test was performed on the ground. Finally, the whole system functionality was validated in a sequence of measuring campaigns.

CHAPTER 2. THE FLEXIBLE MICROWAVE PAYLOAD FOR DRONES

2.1. Introduction

The FMPL-D consists of two instruments; GNSS-R and Radiometry. These systems are passive microwave remote sensing in L-Band, which has several advantages in measuring various geophysical parameters, including soil moisture, ocean salinity, and vegetation. Currently, two satellites use microwave radiometry to study ocean salinity and soil moisture (SMOS and SMAP). Nanosatlab aims to launch two more payloads with microwave radiometry technology, 3Cat4 and 3Cat6 (RITA). 3Cat4 uses GNSS-R and L-Band radiometry to measure ocean salinity and soil moisture; meanwhile, RITA uses L-band radiometry and RFI. FMPL-D is a drone-based platform to test all those systems on the ground and become a testbench for the coming payloads, as well as a valuable tool to make scientific measurements of specific areas with high spatial resolution.

2.2. Global Navigation Satellite System (GNSS) and GNSS-Reflectometry

GNSS

The US military force designed the GNSS to provide a reliable worldwide positioning and timing system. The first GNSS system was the United States Global Positioning System (GPS), operative since 1980 for military aims and liberated for civilian uses in 2000. Since then, other countries have been releasing their GNSS systems. Russia has GLONASS, China BeiDou, Europe Galileo, India IRNSS, and Japan QZSS. The GNSS-R system designed in this project only uses the US's GNSS system, GPS, since these instruments require less bandwidth and, therefore, cheaper hardware. Therefore, the following lines will explain how this system works, being the same concepts can be extrapolated to other GNSS systems.

Introduction to GPS

A GNSS involves a constellation of satellites orbiting Earth, continuously transmitting signals that enable users to determine their three-dimensional (3D) position with global coverage [12].

The following example (inspired by [13]) summarizes, for a two-dimensional (2D) case, the basic ideas involved in GNSS positioning. Consider a flat surface with two GNSS transmitters and a receiver, all in the same plane. The two transmitters are simultaneously transmitting their position and the time-stamp of each message. The receiver has a clock (with a particular bias) that stops when it receives the signal. Now the receiver knows the two time-stamps; the transmitting and the receiving. The difference between these two times is the trip-time of the signal, and by knowing the speed of light, the receiver can

compute its distance from the transmitters accounting for the error made by its clock. This distance is called pseudo-range. Fig. 2.1 shows a graphical representation of how does the receiver compute its position based in the two pseudo-ranges.

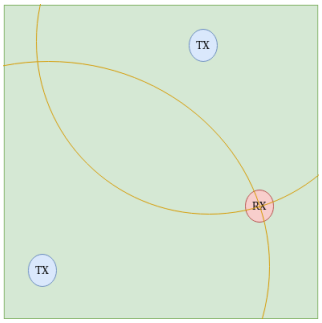


Figure 2.1: Illustration of a simplified two-dimensional GNSS situation

Notice that two possible solutions fulfil the pseudo-range condition. This is not a problem in GNSS since the other position is usually at an impossible place for the receiver, such as outer space. The other problem that arises is that the bias of the receiver's clock is very high, to solve it, another transmitter must be added. In a two-dimensional example, three transmitters were required to estimate the receiver's position, GPS works in three dimensions. Therefore, it needs at least four satellites.

The GPS system is composed of three segments:

- Space segment: The GPS constellation is formed by thirty-two satellites at Medium Earth Orbit (MEO), at an altitude of approximately 20200 km, providing worldwide coverage 24 hours a day. All the satellites are perfectly synchronized and constantly transmitting their ephimeris at a rate of 50 bits/s.
- Control segment: GPS has six monitor ground stations spread around the world, responsible for the system's proper operation. Their primary functions are: to control and maintain the status and configuration of the satellite constellation, to predict the ephemeris and satellite clock evolution, to keep the corresponding GNSS time scale (through atomic clocks), and to update the navigation messages for all the satellites.
- User segment: The user segment is composed by all the user receivers that use the transmitted signals of the satellites to compute their position.

2.2.0.1. GPS signals

GPS satellites transmit signals at a frequency of 1575.42 MHz (L-Band) with a Right Hand Circular Polarization (RHCP). The transmitter can use both "I" and "Q" components to code its data. In the case of L1 C/A signals (), GPS satellites use the in-phase part of the signals to put their messages, leaving the "Q" branch at zero.

2.2.0.2. Modulation

GPS satellites transmit its ephimeris using a Binary Phase Shift Keying (BPSK) modulation. This modulation encodes the information in the phase between the real and imaginary

part of the signal (I and Q), coding the logical one as a phase 0 and logical zero as a phase π. Since GPS keeps the quadrature component of the signal at zero, the BPSK modulation is achieved by only modulating the in-phase component. Fig. 2.2 shows the constellation of a BPSK modulation.

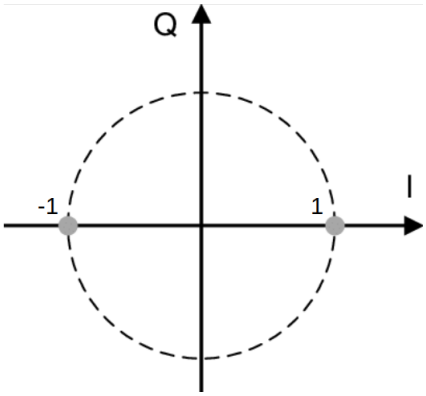


Figure 2.2: Constellation of a BPSK modulation.

2.2.0.3. Coding

All GPS satellites transmit simultaneously at the same frequency, so they interfere with each other. A CDMA channel access method permits the satellites to share the spectrum. It does so by multiplying the bits modulated in BPSK by a Pseudo-Random Noise (PRN), the *Gold Codes*.

Each satellite has its PRN sequence, called C/A code. Each C/A code is 1023 chips long and is transmitted at a rate of 1.023 kHz. The chip rate of the C/A code is much higher than the rate of the BPSK modulation (50 bps). Therefore, the bandwidth of the signal increases greatly. Since now the signal's power is more spread along the spectrum, its peak power decreases greatly, as depicted in fig. 2.3. The signal's peak power could drop lower than the noise power. Still, the receiver can recover the information by processing the signal.

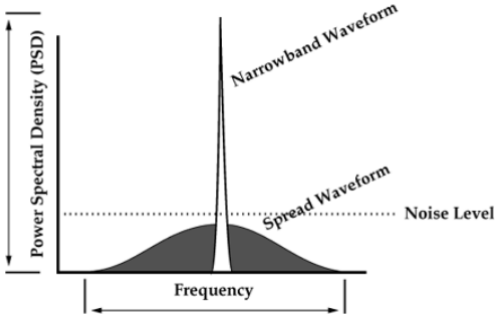


Figure 2.3: Spread spectrum frequency distribution

2.2.0.4. *Signal recovery*

To recover the signal, the receiver has to apply an auto-correlation operation between the received signal and a locally generated replica of the PRN after a proper compensation of the Doppler frequency shift f_d. Equation [2.1] is the operation that the receiver does.

$$Y^C(t,\tau,f_d) = \frac{1}{T_c}\int_t^{t+T_c} S_R(t')a^*(t'-\tau)e^{-j2\pi(F_c+f_d)t'}dt' \tag{2.1}$$

Fig. 2.4 depicts the result of the correlation operation (2.1) between two identical PRN codes (auto-correlation), and two different ones (cross-correlation). When doing the auto-correlation, a peak with a high correlation gain appears, which stands out from the noise floor.

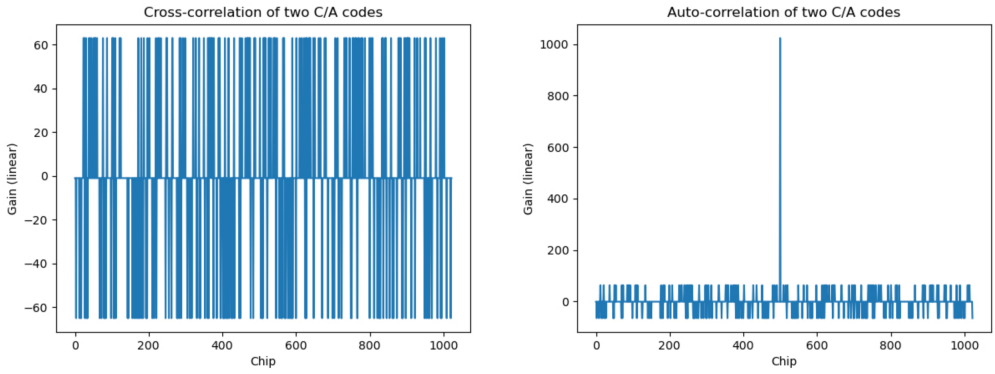


Figure 2.4: Output of the correlation operation

2.2.0.5. *GNSS-R*

Since all the constellation satellites constantly transmit L-band signals worldwide, Earth observation scientists have seen the significant potential for building passive bi-static radars using these signals. That is the main idea behind GNSS-R.

SMIGOL instrument [14] based on GNSS-R retrieves Land Geophysical Parameters. From space, GNSS-R systems have proven excellent results in soil's moisture estimations, such as UPC NanoSat Lab's FSSCat mission [10]. FSSCat is a nanosatellite orbiting in Low Earth Orbit (LEO), at an altitude around 600 km, a much lower altitude than GNSS satellites (MEO).

2.2.0.6. *Passive bi-static radars*

A radar system consists of a transmitter that emits radio waves and a receiver that receives the echoes. Then, the received echo is compared with the original signal, and after some processing, the characteristics of the targets can be assessed, such as their position, speed, etc. It is an active remote sensing system. There are different kinds of radars:

In mono-static radars, the transmitter and the receiver are co-located. On the other hand, in bi-static radars, the transmitter and the receiver are separated by a distance comparable

to the expected target. Passive radars are systems that can detect objects by processing reflections from non-cooperative sources, such as GNSS signals.

2.2.0.7. GNSS-R basic principles

As mentioned before, GNSS-R is a remote sensing technique based in a bi-static radar whose source are GNSS signals. Since GNSS constellations work in L-band, soil's moisture is one of the main outputs from these signals. Fig. 2.5 depicts the basic principle of a GNSS-R system.

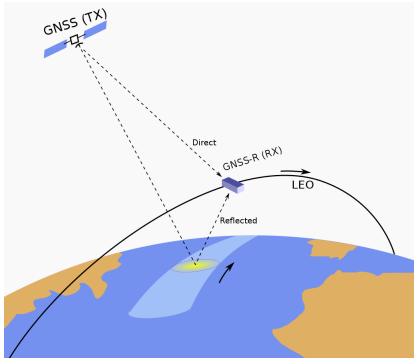


Figure 2.5: GNSS-R basic principle [15].

When the GNSS signal hits a surface, it suffers a diffusive scattering that depends on the surface's roughness. The so-called Delay-Doppler map (DDM) of the scattered signal is the principal output of GNSS bi-static radar. To obtain the DDM, the processing entails cross-correlation of the recorded signal from the down-looking antenna with a duplicate of the GNSS satellite's PRN code for a set of distinct time lags and carrier frequency offsets. This cross-correlation is obtained over a period known as a coherent integration time, which should be smaller than the noisy scattered signal's coherence time.

The DDM creates an image of the scattering coefficient in the delay-doppler domain. The information about the soil's characteristics is subtracted from this image. One of the parameters that can be assessed with the DDM is the wind speed over the ocean. Fig. 2.6 shows an example of a noise-free DDM generated using the CYGNSS End to End Simulator (E2ES) considering a receiver's altitude of 500 km (LEO orbit), reflection incidence angle of 20º, 11 dBi receiver antenna gain and a wind speed of 5 m/s [16]. The figure also shows the power with which the signals are received, a maximum of -139.2 dBm, a very faint signal. Also, the wind direction can be assessed. The signal is scattered when the delay is higher, spreading the power in frequency. The scattering of the signal is related to surface roughness. If the reflected signal comes from the ground, the power received is related to its water content.

GNSS-R data acquisition techniques

The most popular GNSS reflectometers (conventional GNSS-R (cGNSS-R)) correlate coherently for T_c seconds (depends on the altitude) the reflected signal with a locally produced copy of the transmitted signal (open C/A codes only) after compensating for the Doppler frequency shift f_d and for several Doppler frequencies (resulting with the DDM).

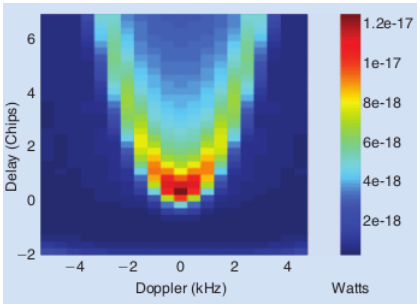
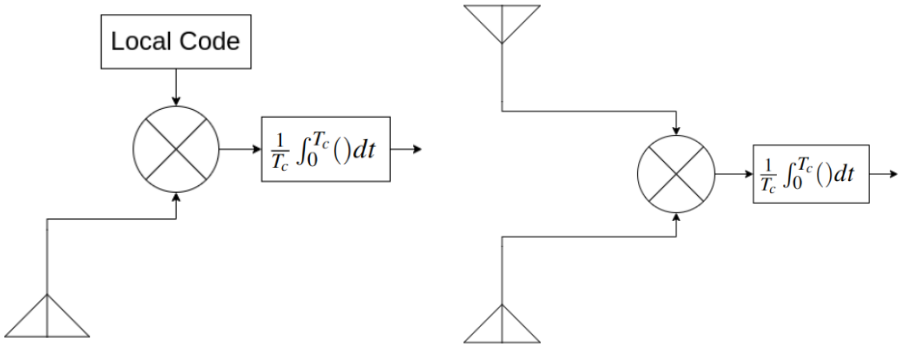


Figure 2.6: Noise-free simulated DDM.

One of the limitations of this technique is that for the same SNR, the larger the RMS bandwidth (β), the better the achievable range resolution. This is a significant problem if GNSS-R wants to be used for altimetry since it limits the range resolution. Fig. 2.7(a) illustrates the concept of a common cGNSS-R instrument.

The so-called interferometric GNSS-R (iGNSS-R) is one way to get around the bandwidth restrictions. This technique cross-correlates the direct signal with the reflected signal after making the necessary Doppler frequency and delay changes. Fig. 2.7(b) illustrates the concept of a iGNSS-R instrument.



(a) Basic concept of a conventional GNSS-R instrument (cGNSS-R).

(b) Basic concept of an interferometric GNSS-R instrument (iGNSS-R).

Figure 2.7: Two different GNSS-R techniques.

Each one of these methods has its pros and cons. In cGNSS-R, the code replica is generated locally, which allows for discrimination from which satellite belongs to each signal, so the signals of each satellite can be tracked; it inherently has an infinite SNR, so smaller antennas can be employed. However, as mentioned previously, it is not practical for altimetry. In contrast, iGNSS-R eliminates the requirement for C/A code knowledge since the reflected and direct signals are immediately correlated. This enables the system to employ satellite radio, satellite television, and other available sources of opportunity with higher transmission power, wider bandwidth, and superior SNR in addition to GNSS signals. For altimetry, the range resolution is greatly improved. The main drawback is the need for higher SNRs, usually solved using larger antennas. Newer strategies have been developed to get around some of the drawbacks of the older ones. Nevertheless, they will not be covered in this work.

In this project, the technique used is cGNSS-R. The resulting DDM is compared with an-

other DDM computed from the direct signal. The power received from the direct signal should be more or less constant at any point, and the reflected is the one that will change depending on the soil's characteristics.

2.3. L-Band Microwave Radiometry (MWR)

Every object at a temperature different from 0 K emits electromagnetic radiation in a specific wavelength range. According to Planck's Law and its approximation at low frequencies (e.g. L-band), a black body (perfect absorber) emits all the absorbed power isotropically. Eq. 2.2 defines the spectral brightness density.

$$B_f = \frac{2 \cdot h \cdot f^3}{c^2} \cdot \frac{1}{\frac{h \cdot f}{e^{k_B \cdot T_{ph}}} - 1} \approx \frac{2 \cdot h \cdot f^3}{c^2} \cdot \frac{k_B \cdot T_{ph}}{h \cdot f} = \frac{2 \cdot k_B \cdot T_{ph}}{\lambda^2}, \tag{2.2}$$

where B_f is the spectral brightness density, which is the amount of power being radiated per unit of surface, per unit of spectral width, and unit of solid angle. k_B is the Boltzmann's constant, f is the frequency in Hertz, T_{ph} is the physical temperature in Kelvin, and c is the speed of light.

The black body is a perfect absorber, so it is an idealized source. A "real" body does not absorb all the incident power; some part is reflected, and some are transmitted into the body. Therefore, a real body radiates less than a black body at the same physical temperature. The emitted brightness is related to the "brightness temperature" (T_B) and the physical temperature, following Eq. 2.3.

$$e_f(\theta, \phi) = \frac{T_B(f, \theta, \phi)}{T_{ph}}. \tag{2.3}$$

In this case, the emissivity of a perfect reflective material is zero, while for an ideal absorber (i.e., blackbody) is one. At L-band, the sea, ice and land have different emissivities and, therefore, different brightness temperatures. The sea surface has a $T_B \sim 100K$, and the sea ice emits in a range depending on its thickness $\sim 120 - 220K$. Land emits different brightness temperatures depending on its surface water content and temperature between 180 and 260 K, and the drier the land surface is, the higher the emitted T_B. The emission of land is the interest of this project.

At low frequencies, T_B is measured with antennas. The radiation emitted by the surface reaches the antenna passing through the atmosphere, which attenuates and scatters it. Also, the radiation power of the antenna modifies the apparent power received.

2.3.1. Types of radiometers

Radiometers can be classified depending on the antenna configuration: the *real aperture radiometers* or the *synthetic aperture radiometers*. The main difference between them is that the resulting pixel size is precisely the half-beam antenna footprint in the first one, so the more directive the antenna is, the better the spatial resolution will be. The synthetic

aperture radiometers cross-correlate the signals collected by different antennas, producing several synthetic beams and thus, providing a complete image of the scanned area.

In this project, a *real aperture* is used. The three main topologies implementing this kind of radiometer are:

- Total Power Radiometer (TPR): The simplest concept of a MWR. The collected radiation is amplified, filtered, IQ modulated to baseband, sampled by an Analog to Digital Converter (ADC), and its power is computed digitally. This output is then compared to calibration values to estimate the antenna temperature. This design is sensitive to thermal variations, which vary the voltage and, therefore, the estimated temperature. These measurement errors can be mitigated by regularly applying proper calibrations.
- Dicke Radiometer (DR): This radiometer aims to minimize the gain fluctuations produced in the TPR. To do so, it includes an Radio Frequency (RF) switch that commutes between the antenna and a known thermally stable matched load. In this way, the power measured is not proportional to T_R anymore but to the difference between the antenna temperature and the reference temperature T_{ref}. However, if the antenna received temperature is different to the reference temperature, which is usually the case, the gain fluctuations are not totally compensated.
- Noise Injection Radiometer (NIR): This radiometer is the evolution of the DR, which corrects its main problem by injecting noise into the antenna so that the $T'_R = T_{ref}$. This radiometer is robust against noise fluctuations, but the noise source must be very stable.

In this project the type of radiometer used is a TPR, and the PCB that implements it is the QM of the FMPL 1 radiometer, [3]Cat4's payload (p. 16-19 from [11]).

CHAPTER 3. HARDWARE DESIGN

3.1. Introduction

In this chapter, the developed hardware of the payloads is presented. Each component is explained independently: Antennas, front-ends, Software-Defined Radio (SDR), processing system, power system, and integration to the drone.

Fig. 3.1 and Fig. 3.2 show a simplified diagram of the hardware and the connections of both systems. Notice that most of the payload is common. This way, integrating the two payloads into one will be easier.

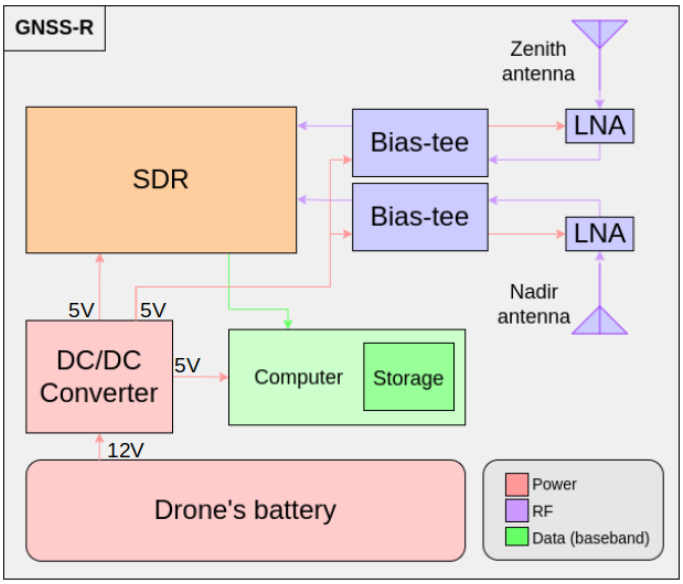


Figure 3.1: Diagram of the GNSS-R diagram

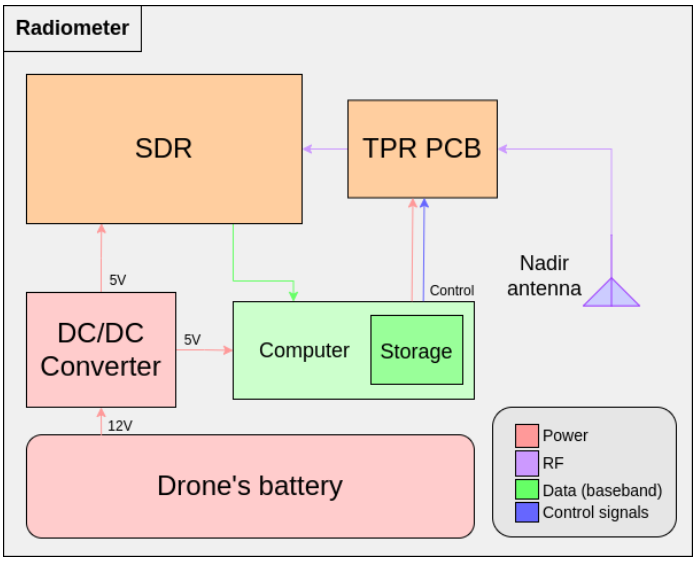


Figure 3.2: Diagram of the radiometer hardware.

The following sections go through all the components, their capabilities, and the selection criteria used to choose them.

3.2. Antennas

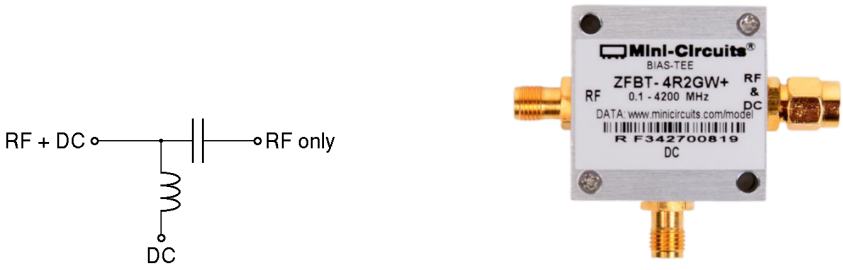
The operation frequencies for GNSS-R and MWR are 1575.42 MHz (GPS-L1) and 1413 MHz respectively (L-band). Patch antennas at these frequencies are small enough to fit under the drone, while their flat form factor makes them easier to integrate.

3.2.1. GNSS-R

According to the Friis formula for noise 3.1, the Noise Figure (NF) of a receiver is mainly determined by the NF and gain of the first component. Therefore, to receive the weak GNSS signals correctly, it is imperative to add an Low Noise Amplifier (LNA) with a low noise figure and high gain as close as possible to the antenna, also known as active antennas. The LNA of an active antenna has to be fed in DC using a bias-tee.

$$F_{total} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + ... + \frac{F_n - 1}{G_1 G_2 ... G_{n-1}} \tag{3.1}$$

A bias-tee is a three-port device that adds a DC bias with an RF signal without disturbing other components. Fig. 3.3 depicts a bias tee's equivalent circuit next to a real one.



(a) Bias-tee equivalent schematic.

(b) Mini-Circuits ZFBT-4R2GW+ bias-tee used in the project.

Figure 3.3: Bias-tee.

Since they are usually used for navigation, there are plenty of GNSS active antenna options in the market. Moreover, GNSS signals are RHCP, and all the commercially available antennas are matched to this polarization, so up-looking is a commercial antenna. Nevertheless, every time an RHCP signal hits an object, their reflections swap the polarization to LHCP, making commercial antennas impractical to detect them.

LHCP antennas at the GPS band are not a commercial-viable product, but for research on GNSS-R purposes. Therefore, the patches had to be explicitly ordered for this reason. To ensure that the antennas were well-matched at 1575.42 MHz with an LHCP polarization, the S_{11} parameter of some of them was measured. To measure the patch antenna, it is good to solder it on a grounded conductive plane. Fig. 3.4 shows the setup used to

measure the patch antennas. Although none of them was perfect, the better one was chosen. Fig. 3.7 depicts the measured S_{11} parameter of the chosen antenna.

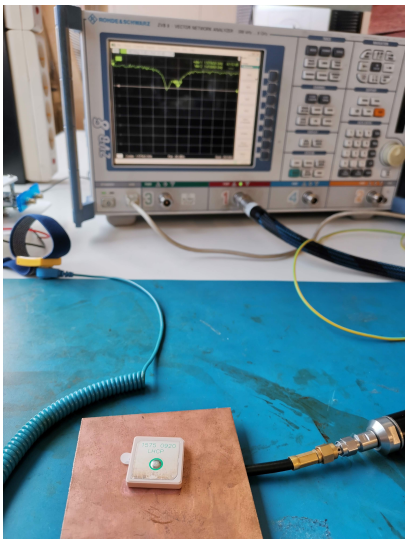


Figure 3.4: Setup to measure the LHCP patch antennas.

As previously stated, GNSS antennas need to amplify the very faint signal adding as little noise as possible. GPS commercial antennas have all the required electronics in a compact casing, so the easiest and cleanest way to solve the problem was by swapping their RHCP patch antenna for the LHCP. Fig. 3.5 shows the process of changing the patch antenna from the commercial GPS antenna.

(a) Original GPS antenna.

(b) LHCP antenna soldered with the LNA.

Figure 3.5: Modifying an off-the-shelf GPS antenna to be LHCP.

Finally, a new casing for the modified antenna had to be 3D-printed. Fig. 3.6 shows the used up-looking and down-looking antennas.

Both antennas were verified by transmitting a sinusoidal wave with a signal generator, receiving it with the antenna, and plotting it with the spectrum generator. This test can prove that the antenna and the LNA work properly, but it can not verify that the LHCP antenna mainly received reflected signals. To prove that, a commercial GPS receiver was used (u-blox LEA-6S - Fig. 3.8), then, using the software ”u-center”, was tested if the receiver could compute its position with each antenna.

As was to be expected, when using the RHCP antenna, the receiver rapidly locked the position when it was pointed upwards. On the other hand, when using the LHCP antenna,

(a) Down-looking LHCP antenna.

(b) Up-looking RHCP antenna.

Figure 3.6: GPS antennas.

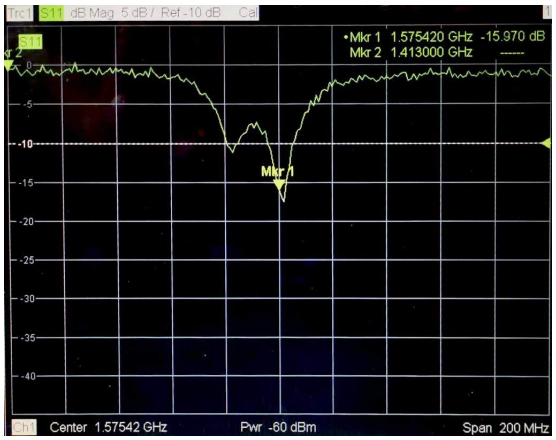


Figure 3.7: S_{11} parameter of the LHCP antenna

Figure 3.8: GPS module used for testing: U-blox LEA-6S.

the locking was done just when it pointed downwards or against a reflective surface. Fig. 3.9 shows a picture of the antenna testing.

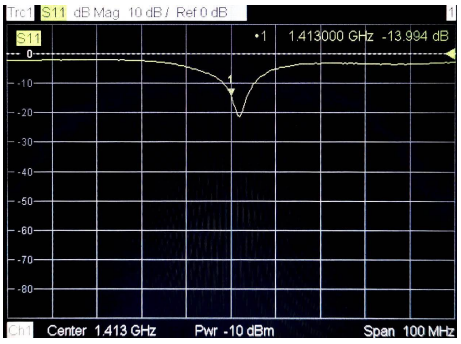
3.2.2. L-Band Radiometry

For the case of MWR, the used antenna is one of the patch antennas designed for the RITA payload ([3]CAT-6). This antenna is slightly bigger since its resonant frequency is lower (1413 MHz). Moreover, it is placed on a ground plane that improves its characteristics. The antenna is built in copper and a RO3010 dielectric with a high dielectric constant and stability. RITA payload uses an array of three of these antennas to get a narrower radiation pattern. In the FMPL-D system, only one of them is used; thus, its beam width will be

Figure 3.9: Testing the LHCP antenna.

wider. Fig. 3.10 is a picture of the antenna next to its measured S_{11} parameter.

(a) Radiometer antenna.



(b) S_{11} parameter of the radiometer antenna.

Figure 3.10: MWR antenna.

In this case, the antenna is passive, meaning it doesn't have an integrated LNA close to it.

3.3. Front-end

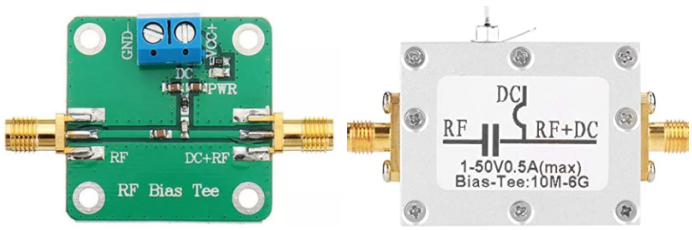
An RF front-end is the phase of an RF system where the signal is conditioned. In the case of the FMPL-D, the two instruments have different Front-Ends. On the one hand, GNSS-R's Front-End is essentially the LNA integrated with the antenna. Thanks to it, the signal conditioning that the SDR needs to do is very little. On the other hand, the MWR's Front-End is implemented in a PCB inherited from [3]Cat4, as explained in the following subsections.

The front-end part shared by the two systems is the one present in the SDR, which consists of a set of filters, amplifiers, and mixers that condition the signal to be sampled. This is explained in more detail in 3.4..

3.3.1. GNSS-R

The GNSS-R front-end consists of the LNAs present in the active antennas and the bias-tees used to feed them. The RF port of the bias-tees is directly connected to the SDR using a coaxial cable. The signal conditioning is done by the SDR.

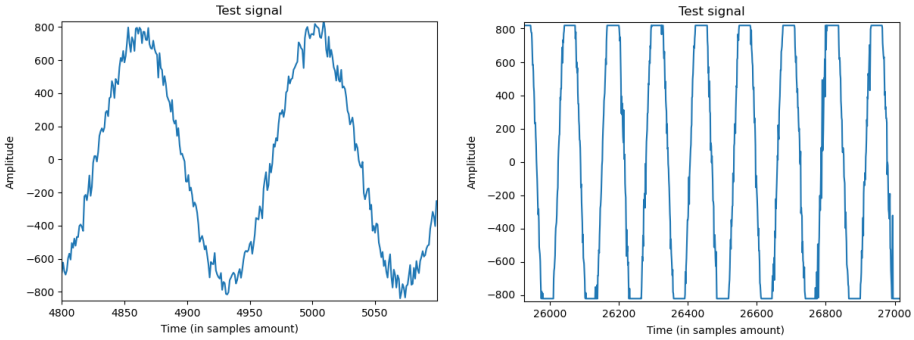
The bias-tees available in the laboratory were too heavy for the drone, so two lighter, cheaper ones were bought. Two different models were selected in case one of them was not good enough. Fig. 3.11 depicts the two bias-tees bought.



(a) Cheap bias-tee 1. (b) Cheap bias-tee 2.

Figure 3.11: Cheaper and lighter bias-tees.

Both devices are wide-band since they can operate between 10 and 6000 MHz. Their S_{12} parameter was measured using a vector-network analyzer (VNA) to qualify the bias-tees. The result was that the parameter was approximately one in all the operational bands. After that, a test was performed. A low-power sinusoidal signal was transmitted using a signal generator. An active antenna (antenna with LNA) was connected to the RF+DC port of the bias-tee, ADALM-Pluto was connected to the RF port, and the bias-tee was fed with 3.3V. Fig. 3.12 show the results of this test.



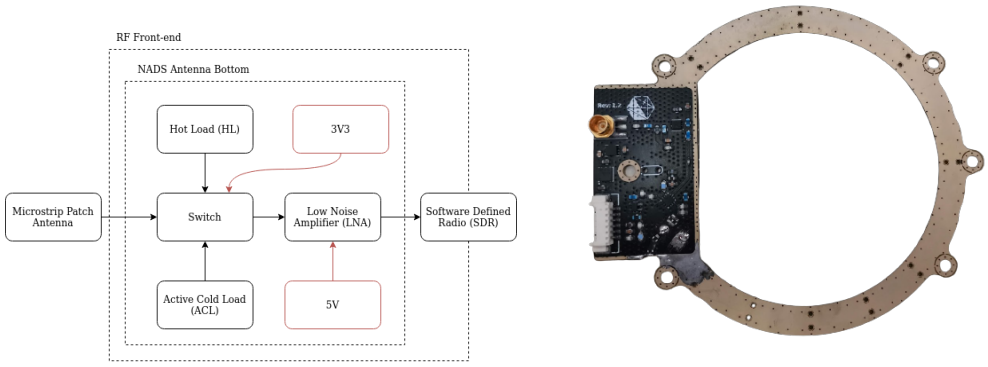
(a) Cheap bias-tee 1. (b) Cheap bias-tee 2.

Figure 3.12: Plot of the resulting sinusoidal signal after having passed through the cheap bias-tees.

The result is a perfect sinusoidal wave for bias-tee 1 (without casing), whereas the signal is *clipped* for bias-tee 2 (with casing). This result discards the use of bias-tee 2 in the FMPL-D since the GNSS signal would also be distorted. In the lab, there was a spare ”expensive” bias-tee (Fig. 3.3(b)) that could be used. The same tests were performed on this one, and the results were positive. The main concern was the weight, but since the other bias-tee was very light, the sum of both was inside the margin.

3.3.2. L-Band Radiometry

As explained in 2.3. the Total Power Radiometer (TPR) is the type of radiometer selected. The TPR schema has a calibration subsystem that consists in an RF switch that commutes between the MWR antenna, an ACL and a HL. A PCB that implements the TPR was developed by the NanosatLab team for the 3Cat4 CubeSat subsystem Nadir Antenna and Deployment Subsystem (NADS). For the FMPL-D system, the Qualification Model (QM) of this PCB has been used. Fig. 3.13(b) is the NADS bottom PCB used in this project. Fig. 3.13(a) shows the block diagram of the NADS TPR PCB.



(a) Block diagram of the TPR front-end.

(b) QM PCB of the TPR system.

Figure 3.13: Total Power Radiometer (TPR) system.

The signal received by the antenna travels through a coaxial cable to the NADS antenna bottom. When the switch is set in the antenna signal position (RF_DWN), the signal is amplified by an LNA to minimize the system's noise figure. Then, it is sent to the SDR that conditions the signal and samples it.

3.3.2.1. TPR PCB

The TPR Front-End is in charge of amplifying and filtering the signal (conditioning). It also holds the calibration loads (HL and ACL) and has a switch that selects the output of the radiometer. The switch needs to swap between three options (HL, ACL or MWR), so just two bits are needed. The Raspberry Pi controls the switch's state by two GPIOs connected with a PicoBlade to the TPR. The software that controls the switch is defined in 4.5.2.

The ACL is implemented with an inverted LNA. Since it is an active device, the manufacturer gives its noise figure, not its internal losses. On the other hand, the HL is implemented with a matched load (a 50 Ω resistor).

Before assembling the PCB with FMPL-D, its functionalities had to be verified. As previously stated, the version of the PCB used is the Qualification Model (QM) of the ^{3}Cat4's NADS bottom PCB, which is the latest version (1.2). The schematics used in the debugging are in the annexe A of the project (Fig. A.1, A.2 and A.3).

The first issue was that some lines and components were missing. The components had to be soldered, and the lines were fixed by adding more tin, assuming degradation in the performance.

Then, a verification test with the device isolated was done. Fig. 3.14 shows the setup for the testing. The Raspberry Pi was used to set the PCB's switch to the calibration loads and the antenna. The picture shows that the power supply for the PCB comes from the Raspberry Pi. This caused problems because the output of the Raspberry Pi was not entirely stable and affected the amplifier, which produced spurious. So for testing purposes, the PCB was fed with a power supply, which also provides information about the current consumption.

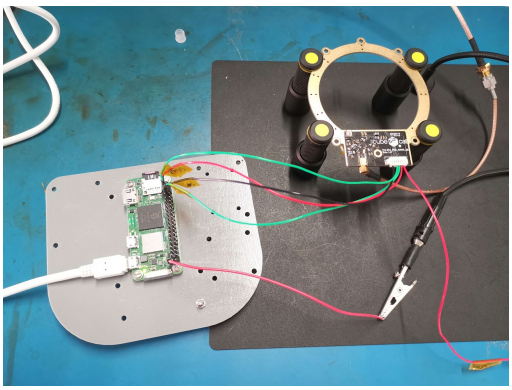


Figure 3.14: Testing the NADS bottom PCB version 1.2 (QM).

The test consisted of switching between the calibration loads and the antenna while watching the signal spectrum at the band of interest. To make the test more realistic, the spectrum was plotted using the ADALM-pluto, as it would be in the actual system. To plot the spectrum, GNU radio was used [17]. When a radiometer works appropriately, the HL and the antenna powers are similar. As the HL is essentially a matched load (50 Ω), if in the antenna port it is connected a matched load instead of the antenna, the HL and the antenna should present the same power. If the values are different, then this proves issues. For this reason, a matched load was been connected to the antenna port.

- Hot Load (HL) (Fig. 3.15): The mean value of the noise floor is -83.23 dB.
- Active Cold Load (ACL) (Fig. 3.16): The mean value of the noise floor is -88.62 dB.
- Radiometer (Fig. 3.17): The mean value of the noise floor is -80.75 dB.

The test results show consistency in the difference between the calibration loads, being the power of the hot load 5.39 dB higher than the cold. However, the radiometer measurement should have been similar to the matched load, which turned out to be higher. The main concern is the RFIs that could interfere with the PCB's strip lines. Recall that one of the lines was missing and had to be covered with tin.

3.3.2.2. TPR calibration

To have the radiometer fully calibrated, it is necessary to know the temperature values T_{HL} and T_{ACL}; for it, a test was conducted to obtain the equivalent noise temperature of the receiver (T_R). According to the Johnson-Nyquist theorem, a resistor's thermal noise output is inversely correlated with its actual temperature. Assuming this, a matched load

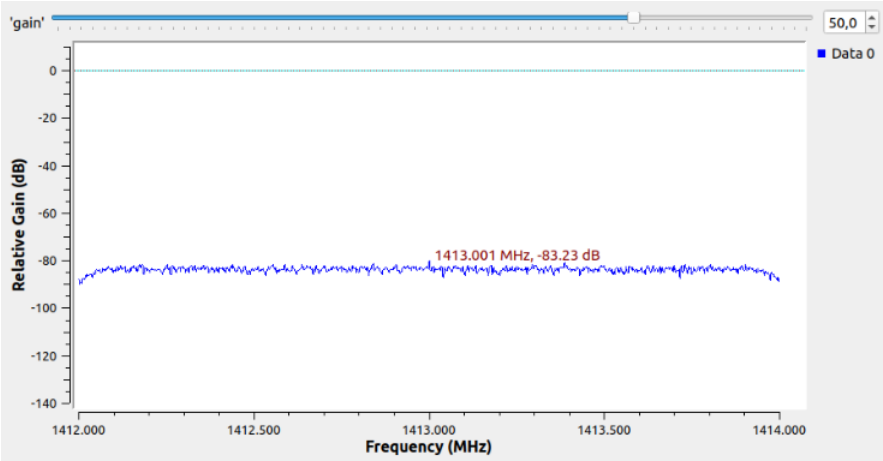


Figure 3.15: Spectrum when the Hot Load (HL) is selected.

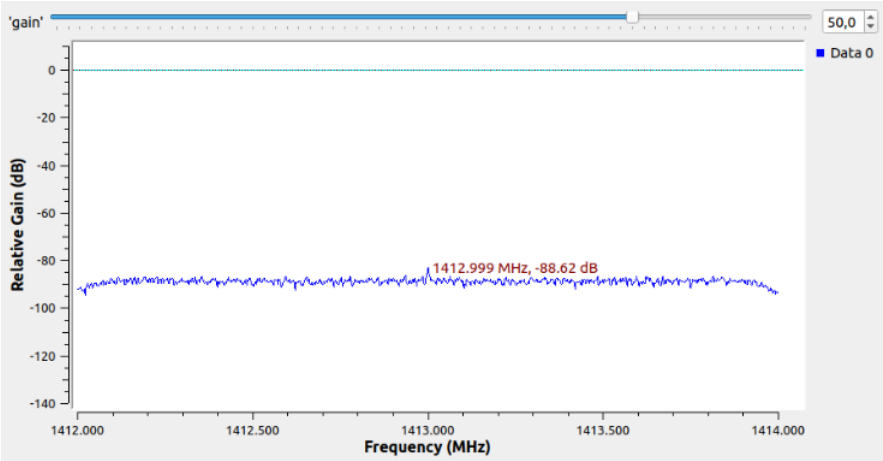


Figure 3.16: Spectrum when the Active Cold Load (ACL) is selected.

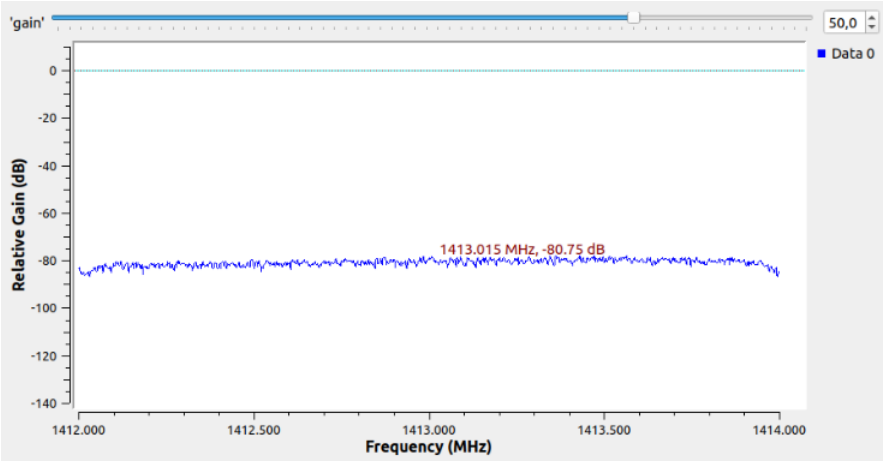


Figure 3.17: Spectrum when the antenna is selected.

was placed at the radiometer port and was heated and cooled at known temperatures. For the cold case, it was submerged in liquid nitrogen (LN2) at $T_{cold} = 77K$, while for the hot case, it was heated up to $T_{hot} = 348K$. At these temperatures, the TPR output noise

power was measured with a spectrum analyzer, resulting in $P_{N_{cold}} = -61.1dBm$ for the cold case, and $P_{N_{hot}} = -58.4dBm$ for the hot case. By substituting these values in Eq. 3.3 we obtain $T_R = 237.5K$. From Eq. 3.5 and Eq. 3.4 the receiver noise figure can be computed, resulting in $NF = 2.597dB$.

$$P = k_B \cdot T \cdot B \cdot G \tag{3.2}$$

$$\frac{P_{N_{cold}}}{P_{N_{hot}}} = \frac{T_{A_{cold}} + T_R}{T_{A_{hot}} + T_R} \tag{3.3}$$

$$NF = 10 \cdot log_{10}(F) \tag{3.4}$$

$$F = \frac{T_R}{T_0} + 1 \tag{3.5}$$

Using the spectrum analyzer, the output noise power of the calibration loads is measured, being $P_{HL} = -60.1dBm$ and $P_{ACL} = -65.15dBm$. So the available dynamic range between the HL and the ACL is 3,05 dB. From the modified Johnson-Nyquist noise equation (Eq. 3.2 the ACL temperature, where k_B is the Boltzmann constant, T the system temperature ($T = T_{ACL} + T_R$), B the system bandwidth and G the system gain. By substituting $P_{ACL} = -93.15dBW$ (ACL noise power measure in dBW), $B = 10MHz$ (spectrum analyzer bandwidth more restrictive), $T_R = 237.35K$, $G = 43.45dB$ (measured gain of the TPR), yields to an ACL temperature of $T_{ACL} = 158.45K$. The same procedure is applied to the HL, but in this case, the noise level is $P_{HL} = -90.1dBW$, resulting in a HL temperature of $T_{HL} = 319.83K$, which corresponds to the physical temperature at which the matched load was subjected during the measurement.

3.4. Software-Defined Radio (SDR)

An Software-Defined Radio (SDR) is a radio communication system where the hardware components are configurable by software (i.e. the gain of the amplifiers). Practically, an SDR is an Analog to Digital Converter (ADC) that can work at multiple frequencies by using an internal Local Oscillator (LO) and a filter bank. These radios are small, light, cheap, and very flexible, making them the obvious choice for the FMPL-D.

For the FMPL-D, an SDR capable of working at frequencies between 1413 MHz and 1575 MHz is needed. The chip RTL2832U has many small-format and low-cost SDR implementations. It can work between 24 and 1850 MHz at a sample rate of up to 2 MSps I&Q, enough for FMPL-D purposes. This SDR has a complex sampling resolution of a byte (8 bits). The chip is implemented as a 5€ USB dongle such as the one shown in Fig. 3.18.

Although the previously presented advantages, this SDR also has one main drawback that has prevented its use. The RTL-SDR dongle has only one receiving input; therefore, the signals from just one antenna could be received coherently, which is enough for cGNSS-R and MWR. As explained in 2.2.0.7., the cGNSS-R technique requires some sort of reference measurement to assess the soil's moisture, which can be done by measuring a surface with known characteristics (i.e. water). Another way to do it is by comparing

Figure 3.18: RTL-SDR USB dongle

the reflected signal with the direct signal. This is the most precise manner to implement cGNSS-R; however, it requires two signals (a direct and a reflected signal) to be received coherently. For this reason, an SDR capable of doing so has been chosen.

The chip AD9363 also has a small format and is implemented in ADALM-Pluto. This receiver works between 325 and 3800 MHz and can work at sampling rates over 2 MSps, allowing for the reception of GNSS signals. The revision C of the ADALM-Pluto board has two transmitting channels and two receiving channels that can work coherently, allowing the implementation of iGNSS-R with just one board. On top of that, ADALM-Pluto features a superior sampling resolution of 12 bits, which allows for better-quality measurements. Moreover, ADALM-pluto features a Field-Programmable Gate Array (FPGA), a small CPU, and a small memory. Fig. 3.19 is a picture of the ADALM-Pluto used.

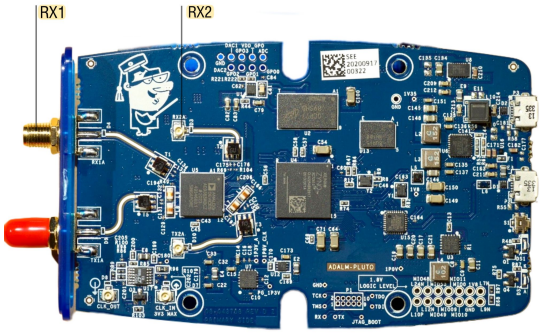


Figure 3.19: ADALM-Pluto SDR

One of the main advantages of software-defined radios is that one hardware can be used in many different applications by changing its parameters via software. Fig. 3.20 shows a very simplified block diagram of the receiving chain of the ADALM-Pluto SDR. In red are the parameters that can be modified by the user, and in green, the feedback given to the Automatic Gain Control (AGC). For this project, the AGC is not used since the power of the received signal has to be known, and if the AGC kept changing the gain, it would not be possible.

Since the SDR's parameters are set by software, the specification of the ones used either for GNSS-R and MWR is justified in section 4.

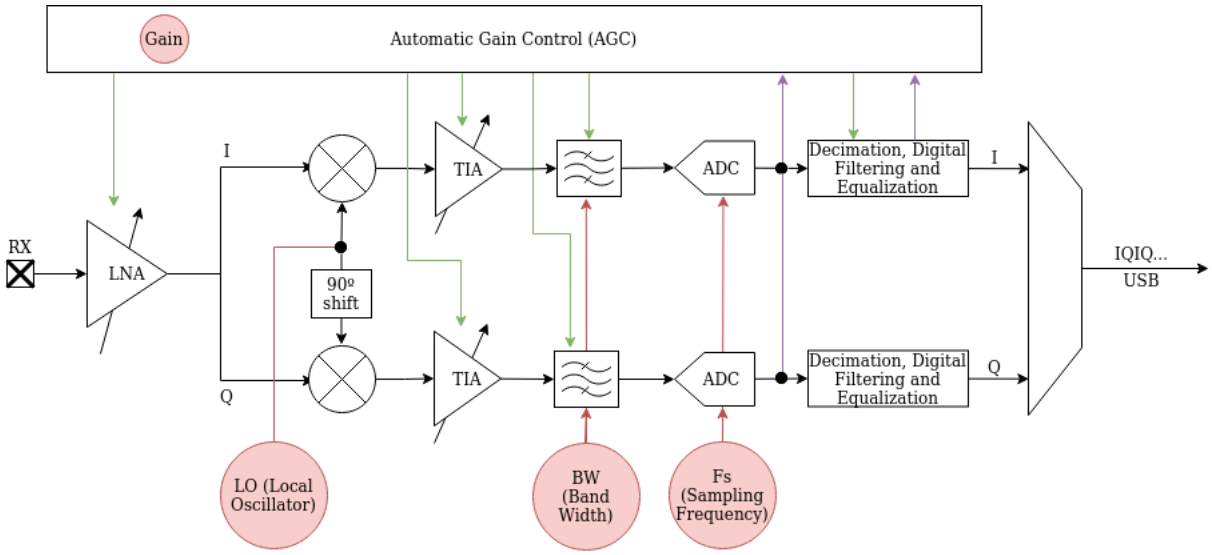


Figure 3.20: Simplified block diagram of ADALM-Pluto's receiving chain.

3.5. Computer and storage

The FMPL-D needs a processing system to manage the information acquired and store it. ADALM-Pluto already has the FPGA that could process the samples. However, the SDR does not have enough storage capacity to save all the data of one flight. Moreover, the radiometer needs to make power-consuming calculations and control the switch of the TPR calibrating subsystem, for instance, via a set of General Purpose Input/Output (GPIO)s.

A Raspberry Pi (RPi) zero W 2 has been chosen for this purpose. This miniaturized computer features a 4-core ARM Cortex-A53 and 512 MB of RAM. It has two micro-USB ports, one for power supply and one that can be used to interact with the SDR. The Operative System (OS) and the data of the flights are stored in a micro-SD card placed in the RPi. Fig. 3.21 is a picture of the Raspberry Pi used.

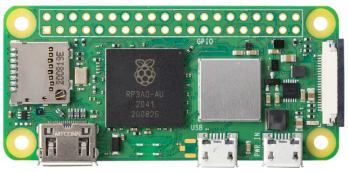


Figure 3.21: Raspberry Pi zero W 2, the processing system for the FMPL-D.

3.6. Power management

3.6.1. Power source of the drone

A typical mid-sized drone battery usually uses LiPo batteries (Lithium Polymer batteries). These batteries provide high specific energy, which is crucial for applications where weight is restricted. Each cell of a drone LiPo battery has a nominal voltage of around 3.7V, and typically mid-sized drones use three cell batteries, resulting in a nominal voltage of 11.1V.

When flying, a drone needs to change its power consumption dynamically as a function of the requirements of the flight. If, for instance, it needs to climb fast, the amount of power required will boost. Therefore, the discharge speed of the battery is also a crucial feature, meaning how quickly can the battery deliver its power. A typical value of maximum discharge capacity is 50C, which means that a fully charged battery with a capacity 5000mAh should be able to provide a maximum of 250A for 1.2 minutes, signifying that at 11.1V, the maximum power delivered by this battery would be around 2.77KW.

According to [18], the nominal power consumption for a drone that weighs 1600g (drone + payload) doing a horizontal flight is around 245W. If the UAV is equipped with a 3-cell LiPo battery (11.1V), the current drain will be around 22A. Therefore, if the battery capacity is 5200mAh, the flight time will be around 14.2 minutes.

3.6.2. Power source of the payload

A study of the payload consumption was conducted in order to determine the best-suited power source. The payload components that need to be fed are different for the two payloads:

- For GNSS-R: The Raspberry Pi, the ADALM-Pluto SDR, and the LNAs for both the up-looking and down-looking antennas.
- For MWR: The Raspberry Pi, the ADALM-Pluto SDR, and the NADS bottom PCB.

A test was conducted to evaluate the amount of power that each payload needs. Both systems (separately) were fed using a power supply that can inform about the output current, then run the code of both systems and see what current they are consuming. The results were that both payloads consume a meagre amount of power, permitting them to be fed using the drone's main battery without reducing the flight time significantly:

- Current consumption of GNSS-R: 600mA.
- Current consumption of MWR: 426mA.

Recall that the payload uses a nominal voltage of 5V, so the power consumption for GNSS-R and MWR are 3 W and 2.13 W, respectively. Taking the more consuming payload (GNSS-R) and adding this power to the drone's nominal power consumption results in the flight time being reduced to 13.8 minutes, which theoretically corresponds to a 24-second reduction, being therefore negligible.

The previous results consider that the payload's power consumption is constant, which it is not. During the initialization of the system, the current consumption peaks as high as 2A at 5V (10W) due to the initialization of the ADALM-Pluto FPGA. If the power source can not provide this peak power when needed, the ADALM-Pluto will not start, and the system will not work. This problem was common during the testing phase since the power source used were the PC's USB ports, and the solution was to move to a more powerful power source. Since the drone's battery peak power is 2.77KW, there will not be problems with the initialization.

As previously discussed, the battery has a nominal voltage of 11.1V, and the payload needs 5V. To reduce the voltage for the payload, a DC/DC converter like the one depicted in fig. 3.22 has been added.

Figure 3.22: DC/DC converter. Input: 11.1V, output: 5V.

3.7. Integration to the drone

3.7.1. Version 1

In order to mount the payloads on the drone, an interface structure was designed using SolidWorks and was then 3D-printed. This structure ought to be light, flexible, and modular, and it needs to be used for both GNSS-R and MWR systems. On top of that, the centre of mass of the drone must be unaltered. Consequently, the structure was designed for the main components to be in the centre of mass, distributed along the same axis. Fig. 3.23 is a render of the drone-payload interface structure, and 3.24 is a picture with the radiometer integrated and ready for flight.

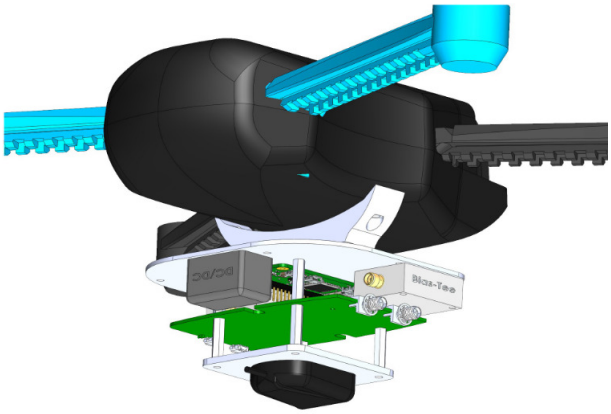


Figure 3.23: First version of the interface structure.

The main idea behind this structure is that not all the structure needs to be changed to change some components. The exact design could be adapted to another drone by just changing the drone adapter. By changing the antenna adapter, another antenna could be used. If needed, there is room for more components in the main plate.

This structure version has the main drawback that the most sensitive components, such as the SDR or the Raspberry Pi, are highly exposed to any accident. This problem motivated the development of a second version of the structure for the FMPL-D.

Figure 3.24: First version of the interface structure attached to the drone.

3.7.2. Version 2

The second version of the FMPL-D is more compact, and the components are protected by 3D-printed cases. Changing from GNSS-R to MWR it is also fast, only requiring to change four bolts. This payload also permits to fix to the drone using bridles, which makes faster to mount it on the drone. On top of that, it permits to stack other payloads on top, such as a LoRa receiver. Fig. 3.25 is a render of the second version of the FMPL-D structure equipped with the radiometry system.

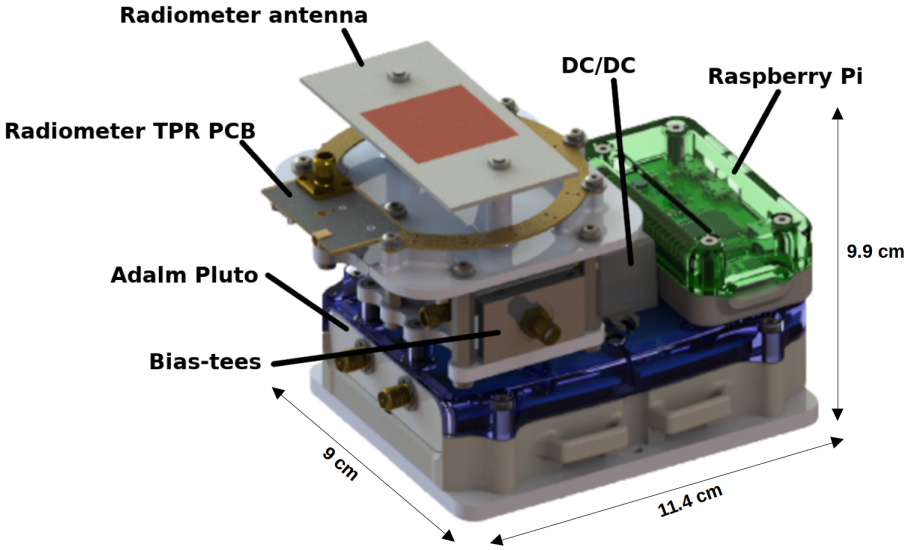


Figure 3.25: Second version of the interface structure.

Fig. 3.26 is a picture of the GNSS-R system integrated together with the LoRa transceiver and antenna ready for flight.

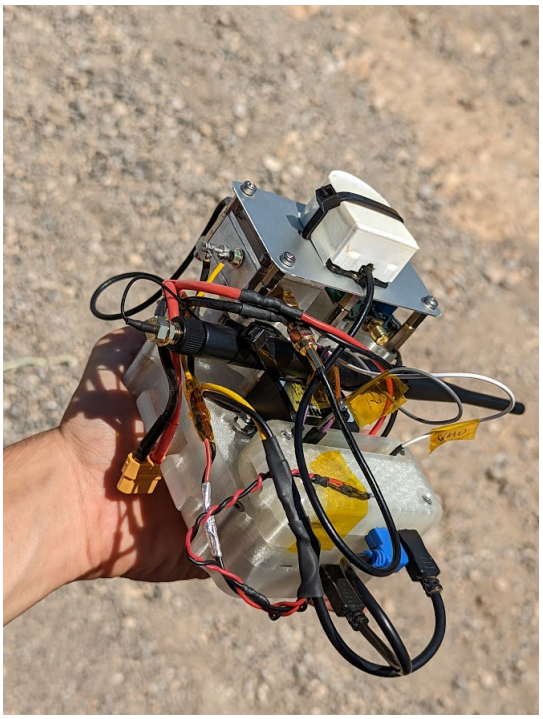


Figure 3.26: Second version of the interface structure, with GNSS-R and LoRa integrated.

3.8. Drone platform

The drone platform must have enough lift to comfortably move around at the desired speeds. Thanks to the low weight of the selected components, the requirements of the drone platform decrease. Both payloads have been weighted; the results are depicted in table 3.1.

Item	Weight
Common items for MWR & GNSS-R	290g
Only GNSS-R items	140g
Only MWR items	40g
MWR total	430 g
GNSS-R total	330 g

Table 3.1: Summary table of the weights of the integrated payloads

With the weights mentioned earlier, the 3D Robotics Iris drone (fig. 3.27) was chosen for this campaign. Several studies on this drone have been conducted to be sure that it fits the requirements.

After selecting the drone, it was noticed that the required long legs were very heavy. Reducing this weight would give more margin to fly safer and with a heavier payload. For this reason, a set of new legs were developed. The legs consist of 1-cm diameter aluminium cylinders. The piece that served as the interface between the drone and the new legs was designed by getting the CAD of the original legs and modifying it. Then the piece was 3D-printed, and the legs were attached to it using epoxy. Fig 3.24 shows the Iris drone with the new legs.

Figure 3.27: 3D Robotics Iris

3.8.1. Features

3D Robotics Iris drone is a quad-copter platform powered by open-source hardware, software and firmware.

The CAD models of all the pieces of the drone are available in the web, so anyone who wants to build the drone can download the CAD and 3D print it. If any drone piece breaks, it can be changed by a 3D printed version for free.

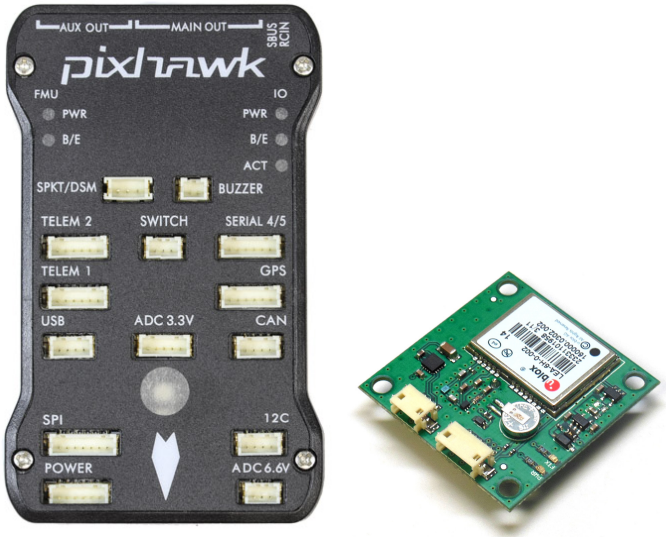
The drone's controller is the 3DR Pixhawk 1 autopilot that runs PX4 on the NuttX OS. It has an integrated gyroscope, accelerometer/magnetometer, and barometer. For the FMPL-D to get useful data, it is important to have the position of each measurement. The drone has the 3DR GPS+compass module that integrates a GPS receiver and a compass on the same board, which sends GPS and compass information periodically to the board. Pixhawk uses this information to control the altitude, position, pitch, and yaw. It can also be used to follow a flight plan or land automatically. Fig. 3.28(a) depicts the Pixhawk board and 3.28(b) the GPS+compass module.

This module has been used to retrieve the drone's position at each time-stamp to avoid redundancy. Six cables are used to connect the GPS to the controller. Two cables for power, two for I2C connection (disabled), and two for serial connection (UART). The board periodically asks for GPS data from the module through its RX pin, and the module responds through its RX pin. Therefore, to get the positioning information of the module, another cable was soldered to the RX pin and connected to the Raspberry Pi. Since the information is transmitted through voltage impulses, connecting the Raspberry Pi to the GPS module's ground was also necessary.

3.8.2. Lift studies

Three lift studies were conducted in order to determine whether the drone can lift the payload or not:

1. A theoretical study.
2. A static test in a controlled environment.



(a) Pixhawk 1.

(b) 3DR GPS+compass module.

Figure 3.28: Pixhawk 1 and 3DR GPS+compass module

3. A flight test.

The drone weighs 965 grams. When the tests were performed, the battery weighed 250 grams (then it was changed by a better one since this was in poor condition).

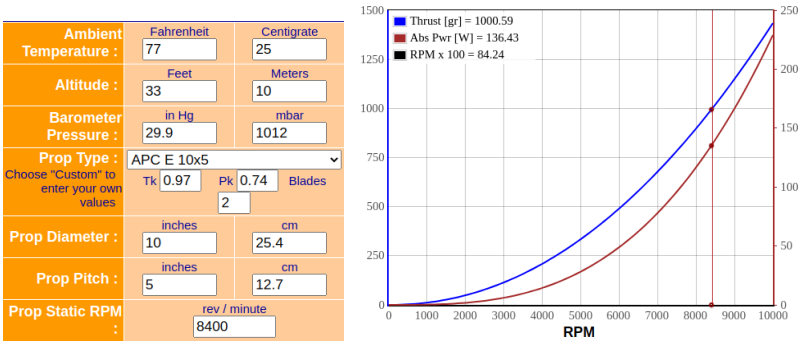
3.8.2.1. Theoretical study

One of the most critical parameters that must be controlled in a drone is its thrust-to-weight ratio. This is the ratio between the weight the motors can lift off the ground over the weight of the whole system. In most cases, this ratio should be maintained for a 2:1 to allow the drone to hover at just half throttle.

The weight of the drone and the payload is already known: 965g (drone) + 250g (battery) + 430g (heavier payload) + 10% margin = 1.8Kg. The lift of the drone has to be determined:

3D Robotics Iris drone features four motors with 850 Kv, meaning that the motor spins at 850 rpm for each volt. As the battery supplies 11.1V of nominal voltage, the engines can spin up to $850 \times 11.1 = 9435rpm$. Since the voltage decays when the battery discharges, this study has considered the worst-case scenario where the motors can spin up to $850 \times 9.9 = 8415rpm$.

The web calculator [19] has been used to estimate the thrust the propellers are generating. This tool can give a first perspective on whether the drone will be able to fly or not. The thrust of a propeller depends on many parameters that are hard to control. However, the tool reduces it to the atmospheric conditions and the propeller shape. For this study, it has been assumed that the flight temperature will be 25 degrees at an altitude of 10 meters Above Sea Level (ASL). The drone's propellers are APC E 10x5, with two blades, a diameter of 25.4cm, and a pitch of 12.7cm. As previously stated, the motors are assumed to spin at around 8415 rpm. All this parameters are summarized in table 3.29(a). The results can be seen in fig. 3.29(b), which shows that each propeller will be able to generate around 1Kg of thrust, and therefore, the whole drone will have a maximum of 4Kg of thrust.



(a) Parameters. (b) Result.

Figure 3.29: Thrust theoretical study.

With this results, the thrust to weight ratio is: $\frac{4Kg}{1.8Kg} = 2.2 > 2$. Hence, the drone platform should be able to deal with a payload up to: 2000g (maximum weight for thrust to weight ratio = 2) - 1250g (drone) = 750 grams. The gap between the theoretical analysis and the real-life world is usually more significant than we would like. For this reason, a test was conducted.

3.8.2.2. Static test

The experiment aims to estimate the maximum lift that the drone can provide without having to fly the drone (for safety reasons). The idea was to overload the drone so that it could not take off when the throttles were at full power and put the drone on a weighing machine. The drone was loaded with a box full of lead weights with a total weight of 5875 grams. Adding the drone and battery, it was 7100 grams. Then, spin the propellers at different speeds and see what weight the scale read. The measurement of the scale will be proportional to the lift of the drone. The test results are depicted in table 3.30.

THRUST	WEIGHING SCALES RESULT (g)	LIFT (g)	USEFUL PAYLOAD (g)
0	7100	0	-1225
0.25	6600	500	-725
0.5	5800	1300	75
0.75	5250	1850	625
1	5220	2000	775

Figure 3.30: Results of the static lift test.

The trust control was done by hand, moving the thrust lever at approximately the amounts described in the table. The useful payload is the difference between the generated lift, the drone's weight, and the battery's weight. The first conclusion taken from the results was that the thrust that the drone supplies is not linear with the lever position. The thrust depends mainly on the software. The drone was capable of generating a lift of 2000 grams, which could be able to lift the payload, but with very poor manoeuvrability since the thrust-to-weight ratio would be close to one.

Having seen the results, one could conclude that the drone is incapable of doing this job. However, the theoretical study argues the opposite. This discrepancy may be because the test conditions were not accurate. The Iris drone computer uses many data from the sensors to adjust the flight to optimal conditions. Since the computer was not detecting an

upwards acceleration, it knew something was wrong with the flight. On the other hand, the proximity from the ground and other interfering objects, such as the ropes used to hold the payload, could create turbulence that worsens the lift generation.

In order to be sure that the drone could lift the payload, it was necessary to perform a lift test in a more realistic environment, thus, a flight test.

3.8.2.3. Flight test

The drone would try to fly a dummy payload, and we would keep taking out weight until the drone would be capable of flying with comfortable manoeuvrability. The chosen dummy weight was a one-litre bottle of water since it can be quickly emptied. The bottle would be in a bag held by a dummy structure designed as a CAD model and then 3D-printed. Fig. 3.31(a) shows the CAD model of the ”dummy adapter”.

A low-altitude flight was more than enough to prove the system's lift. The test was conducted in an area far away from buildings or people. Fig. 3.31(b) is a picture of the flight test.

(a) ”Dummy adapter” for the test payload.

(b) Drone flying during the lift flight test.

Figure 3.31: Flight test to determine the drone's thrust.

As the theoretical study suggested, the flight results are depicted in 3.2, showing that the drone can comfortably fly with a payload of 700 grams.

Weight	Performance	Description
1200	BAD	The drone could not take off.
1000	MARGINAL	The drone could take off, but the manoeuvres were slow. The landing was hard due to the overweight.
700	GOOD	The drone could take off and land smoothly. The manoeuvrability was also good.

Table 3.2: Results of the flight test.

3.8.3. Condor Beta drone

Although 3D Robotics Iris should have been enough for the measuring campaigns, it was decided to change. The reasons behind the change on the drone platform are justified in **??**. The selected drone is the Condor Beta 3.32. This drone provides superior specs:

- Stability. Small drones like Iris (55 cm motor-to-motor) require big pitch and roll angles to stabilize themselves against the wind or harsh manoeuvres. When high

pitch and roll angles, the payload antennas no longer point to the ground, losing all the data. Condor beta is a much bigger and more stable drone.

- Flight time. Condor beta can stay in the air for 40 minutes with a payload like the FMPL-D, permitting more and longer flights in the campaigns.
- Cargo capacity. Condor beta can lift up to 5 Kg of useful payload, a very comfortable safety margin for the FMPL-D.
- Telemetry in real-time. The interface for the pilot is a map where you can see the position of the drone, its altitude, battery life, etc. With it, the pilot can fly more precisely over the areas of interest.
- Saves the log files. The drone automatically saves all the flight data, including the altitude, position and time. Therefore, it is no longer necessary to save the positioning data with the Raspberry Pi, simplifying the payload.

Figure 3.32: Condor Beta.

CHAPTER 4. ACQUISITION SOFTWARE DESIGN

4.1. Introduction

This chapter discusses the software in charge of acquiring the data for further processing. Firstly, it is explained the code of the GNSS-R system. And secondly, the code for the MWR.

The Raspberry Pi runs the acquisition code by connecting with ADALM-Pluto to get the samples and storing them in the micro-SD card. The C library ”libiio” was developed by Analog Devices to simplify the software development for Linux Industrial I/O devices such as ADALM-Pluto, abstracting the low-hardware specifics of the devices. On top of libiio, the NanosatLab developed another library called ”iio-handler”, which uses libiio and simplifies, even more, the use of iio systems. ”iio-handler” is developed in C++. For this reason, the acquisition code has been done in C++.

Roughly, the structure of both GNSS-R and MWR is the same, and it is shown in the flowchart in 4.1. However, they present differences, as exposed in the following sections.

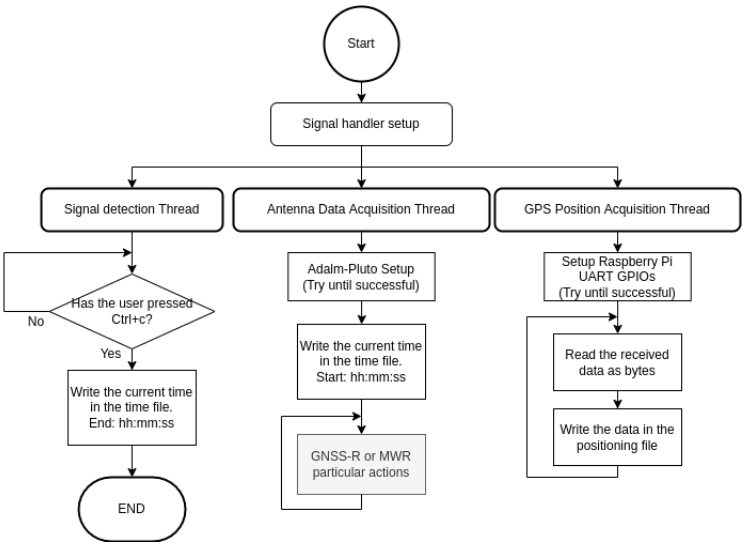


Figure 4.1: Flowchart of the common acquisition software structures for GNSS-R and MWR systems.

The code execution starts with a computer connected via WiFi and ssh with the Raspberry Pi. The computer loses the connection when the drone moves away, and the Raspberry Pi should keep running the code. To deal with it, the program is started with ”nohup”, which will keep running the program even though the user ends the session.

When the program starts, the code stays aware of any interruption signal sent by the user pressing Ctrl+C, which would finish the program. Then, the execution of two simultaneous threads starts:

- Positioning data thread.
- Antenna data thread.

What happens in these two threads is explained in the following sections.

4.2. Buffer structure

The ADALM-Pluto SDR can receive, sample, and send signals to a computer in real-time. To do so, it features two IIO buffers. Buffer 1 will be the first buffer, where the SDR stores the data just after sampling it. Buffer 2 is the one to which the user has access to read. Once the ADALM-Pluto starts receiving, buffer 1 is filled until it is complete. When the user asks for it, this data is moved to buffer 2, and the user starts reading it while buffer 1 fills again. By default, the SDR sets the buffer refill mode to ”blocking”. If the computer empties buffer 2 faster than the SDR can fill buffer 1, it will wait until it has enough data to read. This feature makes it possible to continuously acquire data. However, a problem arises.

If the computer reads the data from buffer 2 slower than the time it takes to fill a buffer, data will be lost. Exactly this happens in the GNSS-R system, producing a loss of data for every buffer-size time. To confirm it, we plotted a sinusoidal signal transmitted by a signal generator and received and sampled by the SDR. The result is depicted in Fig. 4.2. Red lines are put at each buffer size, set to 2046 samples. The signal stopped being continuous at each buffer size time, producing a random change in the phase of the sinusoidal signal.

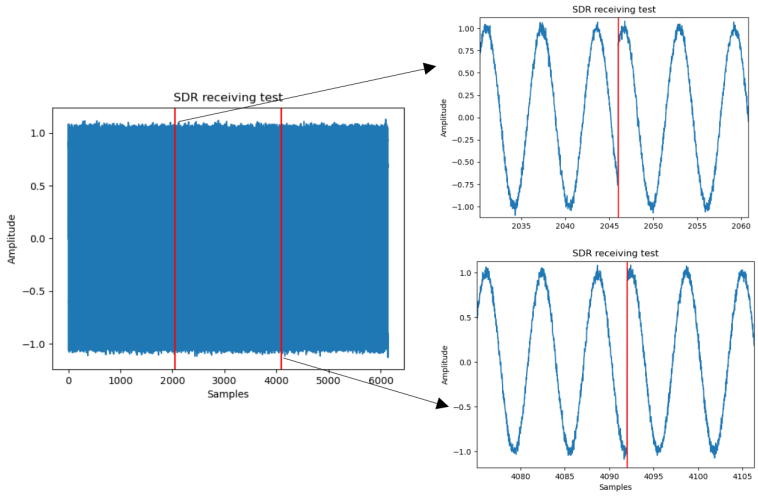


Figure 4.2: Receiving a test signal.

This problem only occurs when receiving coherently with both channels, and it is attributed to a bug in the acquisition software that could not be solved yet. For this reason, to minimize the data loss, the buffer size has been set to the highest multiple of 2046 (one millisecond) that does not slow down the data acquisition: 30×2046. The data loss has been considered in the signal processing, as detailed in 5.3.. Since the radiometry system only uses one receiving channel, it does not face this problem.

ADALM-Pluto’s ADC has a 12-bit resolution, and the in-phase (real) and the quadrature (imaginary) components of the signal are stored separately. Therefore, four bytes are needed to store just one sample; 16 bits for the in-phase and 16 for the quadrature component. As the data is received, it is stored consecutively; first, both components of the first channel, in-phase (IA) and quadrature (QA); then, both components of the second

channel, in-phase (IB) and quadrature (QB), and so on. Fig. 4.3 shows the structure of one buffer.

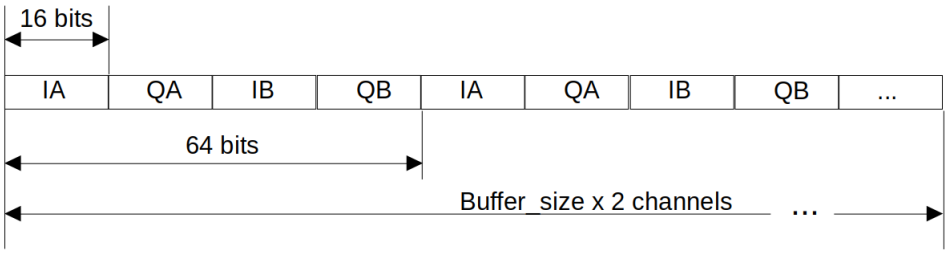


Figure 4.3: ADALM-Pluto, two channels buffer structure.

4.3. Positioning data

As explained in 3.8.1., the positioning data is retrieved from the messages that send the GPS module to the Pixhawk board. The GPS receiver is a u-blox NEO-7 module, and it can generate the output message either in NMEA or UBX protocol. However, even though NMEA is a more user-friendly protocol, the Pixhawk controller uses UBX to communicate with the receiver, and for this reason, this is the protocol that will be read and decoded. The UBX protocol specification is detailed in 5.2.. In this section, the main interest is to describe how the message is acquired.

The protocol uses 8-bit binary data (1 byte) and is structured in different kinds of messages. The messages are usually represented in hexadecimal, each byte representing a set of two hexadecimal digits. Therefore, the data is read and written in a file in hexadecimal format.

The GPS module and the Pixhawk are connected via a serial port using the UART protocol; therefore, the Raspberry Pi has to read the information through its pre-defined UART pin present in the GPIO. One of the main characteristics of the UART protocol is that it can transmit at very different speeds depending on the application. In the datasheet of the GPS receiver, it is stated that the default speed for this model is 9600 bauds, the most common transmission speed. However, when testing the system, it was noticed that the baud rate was different since there was no coherence in the messages received. To check the baud rate at what the device was actually transmitting, a tool called ”Saleae” was used. Saleae is a piece of hardware that can record digital and analogue signals and decode their protocols (if present). By connecting Saleae to the RX cable where the data was flowing, the actual baud rate could be computed: 230400 bauds.

WiringPi is the library used to access the Raspberry Pi’s GPIOs functionalities. Specifically, since the purpose is to use the serial GPIO, it has only been used the WiringSerial, which is a simplified serial port handling library. With this library, the code needed to use the RPi’s serial port is reduced to two function calls:

- `int fd = serialOpen(char *device,int baud)`. This function opens and initializes the serial device ”device” with the baud-rate ”baud”. The return is the file descriptor (-1 for any error).

- `int out = serialGetchar(int fd)`. This function returns the next character available on the serial device.

Putting the function `seralGetChar()` in a loop, converting its output to hexadecimal, and writing the result in a file, all the data the GPS receiver generates is saved for further processing.

4.4. GNSS-R

The first approach of the GNSS-R system was to implement a cGNSS-R model with an RTL-SDR USB dongle. For this reason, the first approach of the acquisition was thought for this schema. There is an open-source software that simplifies data acquisition with RTL-SDR systems; it is called rtl-sdr and posted on Github ([20]). This code was downloaded and modified to be used for GNSS-R. However, this code was never used because of the need to receive from two channels coherently.

The acquisition code for the GNSS-R system is very simple. It only has to acquire raw data from both channels and store it in files.

4.4.1. SDR Parameters

Both channels of the SDR are configured with the parameters depicted in Table 4.1.

Parameter	**Value**
Local Oscillator (LO) Frequency	1575.42 MHz
Sampling Frequency	2.046 MSps
Bandwidth	2.046 MHz
Gain	50 dB
Buffer size	2046 x 30 Samples

Table 4.1: SDR configuration for GNSS-R acquisition.

LO is the frequency at which Pluto's internal mixer demodulates the signal to the base-band, so its value corresponds to GPS L1, the frequency that the GNSS-R system uses. Fig. 3.20 illustrates it. However, the SDR is not perfect, and the mixer does not multiply the received signal by the exact same frequency set because of an internal bias. This will be fixed in the processing chain 5.3..

The sampling frequency is set to 2.046 MSps. The GPS C/A code is transmitted at 1.023 MHz, so sampling it at 1.023 MHz would be enough to get one sample per chip. However, the reflected signals are very weak, and a higher sampling rate can improve the correlation gain with the locally generated PRN code. For this reason, it has been decided at 2×1.023 MHz to get two samples per chip. The selected sampling rate also defines the bandwidth that the signal will have without provoking aliasing. The bandwidth of a pulsed signal can be approximated by the expression $BW \sim \frac{1}{\tau}$, being $\tau = \frac{1}{ChipRate} = \frac{1}{1023KHz}$, so the bandwidth of the signal is approximately 1023 kHz. The Nyquist-Shannon criterion states that, to avoid aliasing, the sampling rate of a signal must be at least two times its bandwidth. Applying the criterion, if $f_s = 2046kSps$, the maximum bandwidth is 1023 kHz,

which agrees with the signal's bandwidth. Fig. 4.4 illustrates chip sampling with two different sampling rates. As Fig. 3.20 illustrates, the sampling frequency is a parameter of the ADC.

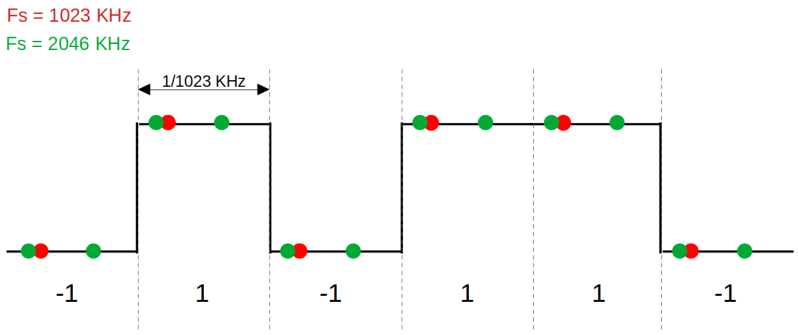


Figure 4.4: PRN sequence sampling with different sampling rates.

The bandwidth parameter defines the bandwidth of an adaptable filter at the input of the SDR, as shown in Fig. 3.20. Since the sampling frequency has been set at 2046 kHz, to prevent the filter from deleting some of the desired signals, the safest value is to set it at 2046 kHz as well.

The gain range of the ADALM-Pluto goes from 0 to 70 dB. Given that the GNSS signal has low power, the gain must be high. Acknowledging that the SDR is not perfect, some power from the local oscillator gets to the output and contaminates the signal. If the SDR gain is high enough, the noise floor can cover this interference and, therefore, avoid it. Visualizing the spectrum of the ADALM-Pluto's output, a gain can be set high enough to correct those issues. Fig. 3.20 shows the effect of the gain parameter on the ADALM-Pluto.

Buffer size is the number of complex samples that ADALM-Pluto can store at a time. One PRN sequence has 1023 chips, which are transmitted at a rate of 1023 kHz; therefore, 1023 chips last for one millisecond. Since the sampling frequency used is 2046 kHz, 2046 is the number of complex samples stored in one millisecond. The buffer size is set to 30 milliseconds, and the reason for it is justified in the following subsection (4.2.).

4.4.2. Code

Fig. 4.5 shows a flowchart of the C++ code that runs in the Raspberry Pi. It is a very simple diagram since it does not process any data.

ADALM-Pluto's buffers are filled, read, and each sample is stored in two local buffers (one for each IQ component). Then the data is copied to two binary files, one for each antenna: "data1.bin" and "data2.bin".

4.4.3. Data generation

The amount of data that the system generates is important to assess whether there will be enough storage or not in the SD card.

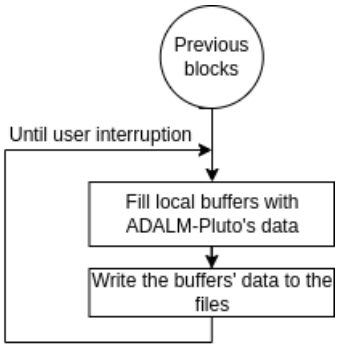


Figure 4.5: Flowchart of the GNSS-R acquisition code.

As previously stated, the reflectometer system samples and stores the data as it comes without any further processing. This is extremely inefficient since, with some processing, the amount of data to store could be drastically reduced. However, the speed is very important to prevent any data losses, and since raspberry's CPU power is very limited, adding real-time processing to the data could slow down the whole system. This was one of the trade-offs that had to be dealt with. To make the decision, the amount of data the system would generate if no processing were performed was computed:

The sampling frequency is $f_s = 2046kSps$. The resolution of the ADC is 12 bits, but since the computer packs the bits in bytes (8 bits), each sample is stored in a 16-bit integer variable. Hence, *32.736 Mbps* is generated, corresponding to *4.092 MBps*. Therefore, for a 30-minute flight, GNSS-R would acquire 7.37 GB.

With these results, the trade-off is solved. A 32 GB SD card can handle the data of four 30-minute flights, more than enough to fulfil the project's objectives. Therefore, the onboard processing can be avoided.

It is worth mentioning that the error discussed in 4.2. affects the data generation, reducing it significantly.

4.5. L-band Microwave Radiometry (MWR)

The MWR code was for the radiometry system of the [3]Cat6 payload (RITA). The code has been adapted to work with the Raspberry Pi and the NADS TPR Front-End.

4.5.1. SDR Parameters

For the MWR system only one channel is configured with the parameters depicted in Table 4.2:

LO is the frequency at which Pluto's internal mixer demodulates the signal to the base-band. Its value corresponds to the L-Band working frequency of the radiometer, 1415 MHz. As explained in 2.3. at this band, information about soil's humidity can be retrieved.

The sampling frequency (f_s) is set to 2 MSps.

The bandwidth parameter defines the bandwidth of an adaptable filter at the input of the

Parameter	Value
Local Oscillator (LO) Frequency	1415 MHz
Sampling Frequency	2 MSps
Bandwidth	2 MHz
Gain	60 dB
Buffer size	4096 Samples

Table 4.2: SDR configuration for MWR acquisition.

SDR, as shown in Fig. 3.20. Since the sampling frequency is 2 MHz, to prevent the filter from filtering out some of the desired signals, the safest value is 2 MHz as well.

The gain range of the ADALM-Pluto goes from 0 to 70 dB. Given that the radiometer needs to detect signals at very low power, the gain must be high. 60 dB was proven to perform well.

Buffer size is the number of samples that ADALM-Pluto can store at a time. In this case, there is only one active channel in the SDR, so there is no continuity problem as in the GNSS-R system. A 4096-sample buffer is enough for the acquisition to be continuous without losing data.

4.5.2. Code

The radiometer code is in charge of:

- Acquire data from the antenna or the calibration loads.
- Compute the power and write it in a file.
- Change the state of the TPR's calibration switch.

First, ADALM-Pluto's buffers are filled, read, and each sample is stored in two local buffers. Then, the power of the samples stored in each buffer is computed with the following formula:

$$P = \frac{1}{BufferSize} \sum_{i=0}^{BufferSize} \Re(S_i)^2 + j \cdot \Im(S_i)^2$$

Where P is the complex power, *BufferSize* is 4096, and *S* is the buffer with all the samples. This formula makes the average of all the samples present in the buffer, thus, the coherent integration time of the radiometer is $\frac{SampleAmount}{f_s} = \frac{4096}{2\times10^6} = 2.048ms \sim 2ms$. The computed value is not physical power; thus, it does not have power units. The samples are in units referenced to the full scale of the SDR, so they do not have power units. ADALM-Pluto's ADC has 12 bits, so the maximum power would be when all 12 bits are one and the minimum when zero. To relate these values with a physical power calibration such as the one explained in 3.3.2.2. is needed.

This process is repeated for 100 ms for the radiometer antenna and 50 ms for the calibration loads, ACL and HL. Therefore, the antenna will store 48 samples for each loop, and the calibration loads 24 each.

After saving the results, the TPR's switch needs to change its state. As detailed in **??**, the TPR's switch has two control bits that select the output. Two of the Raspberry Pi's GPIOs

have been programmed according to Table 4.3. The software waits for 50 ms to switch correctly to avoid saving corrupted data during the toggle.

Input	V1 (GPIO 2)	V2 (GPIO 3)
OFF	0	0
MWR	1	0
HL	0	1
ACL	1	1

Table 4.3: TPR truth table.

Fig. 4.6 is a simplified flowchart for the acquisition of the MWR data.

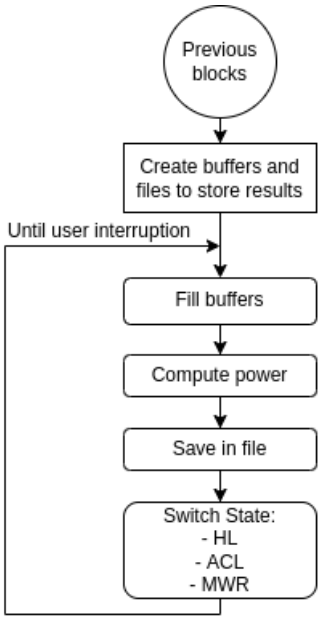


Figure 4.6: Flowchart of the MWR acquisition code.

4.5.3. Data generation

The amount of data that the system generates is important to assess whether there will be enough storage or not in the SD card.

The radiometer has a sampling frequency of $f_s = 2MHz \Rightarrow 2Msps$. The resolution of the ADC is 12 bits, but since the computer packs the bits in bytes (8 bits), each sample is stored in a 16-bit integer variable. Hence, $32Mbps$ are generated. As explained in the previous section (4.5.2.), while the data is being generated, the radiometer fills the 4096-bits buffer, integrates all the samples coherently, and computes their average power. Then, this information is stored in two files: one for the calibration loads and one for the radiometer measurements. Each line of the file is a calculated power, and it looks like this:

- `PM: 559.223755, 926.617859`
- `HL PM: 665.785095, 849.640381`

- `ACL PM: 252.950974, 212.384064`

The first line is saved in the *mwr_pwr_acquisitions.txt* file, and it corresponds to the *Power Measurement* (radiometer). The other lines are saved in the *calibration_pwr_acquisition.txt*. The second corresponds to the *Hot Load*, and the third to the *Active Cold Load*.

The sampling, integrating and storing process is repeated for 100 ms for the radiometer and 50 ms for the calibration loads, and 50 ms for the transitions. Fig. 4.7 shows how the time is managed.

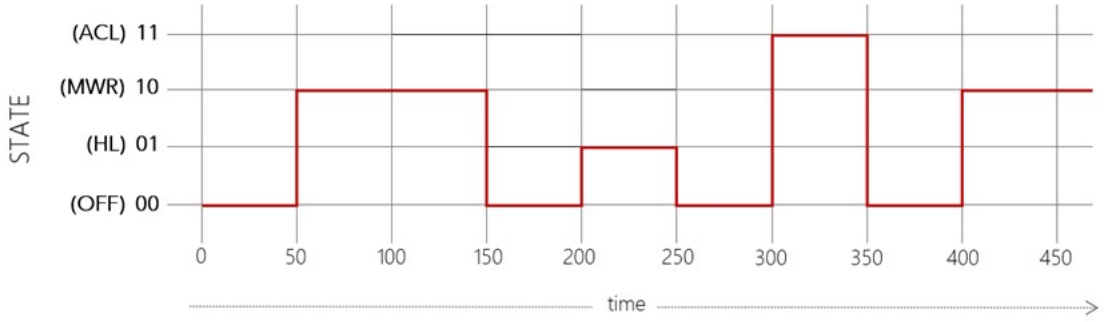


Figure 4.7: Radiometer's time diagram.

An entire period lasts for 400 ms, and in this time, three lines like the ones above are generated. Each line occupies around 30 bytes; therefore, in 400 ms, 90 bytes are generated. To sum up, the radiometer generates: $\frac{90}{0.4} = 225Bps$.

CHAPTER 5. PROCESSING SOFTWARE DESIGN

5.1. Introduction

The data processing is the last part of the FMPL-D design. Once achieved campaign, both GNSS-R and MWR information is kept in the micro SD card, and to make sense of it, it has to be downloaded and processed in a processing chain. This chapter discusses this last part of the FMPL-D system.

The acquisition software was coded in C++ to enhance the efficiency and take the maximum profit from the scarce resources of the Raspberry Pi zero. In the processing chain case, efficiency is not as crucial, so for simplicity's sake, Python 3 and MATLAB were the chosen languages.

The following sections detail the signal processing from the acquired samples to meaningful soil-moisture maps.

5.2. Positioning data

To correctly interpret the remote-sensing data, it is essential to have each sample correlated with its exact position on the map. As explained in 4.3., the acquisition system stores all the GPS data in a file coded in UBX protocol, a u-blox proprietary binary protocol.

The protocol is compact, checksum protected, and modular (uses a 2-stage message identifier). The structure of a basic UBX frame is shown in Fig. 5.1.

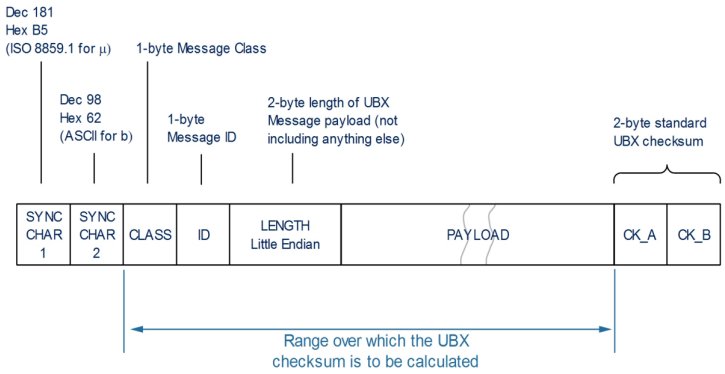


Figure 5.1: Basic UBX frame.

The protocol provides tons of information packed in different messages. The message we are interested in is the NAV-SOL (0x01 0x06), which includes the navigation solution with the time stamp and the drone's position. A python code is in charge of decoding this binary information and storing it into a *.csv* file with the time stamp, latitude, longitude and altitude (above sea level).

The Condor Beta drone already provided this information with a log file (*.csv*).

5.3. GNSS-R

The GNSS-R system generates two binary files. One store the samples of the nadir antenna, and the other the ones of the zenith antenna. These files store the data that the system has detected at the moment of the flight, which is the band centred at 1575.42 MHz with 2 MHz bandwidth; therefore, it has the information of all the GPS L1 satellites in view at the moment of the flight.

The information that we are interested in is the power received from each available satellite. The first piece of code is the one in charge of interpreting the binary files and extracting the maximum amount of information. The second correlates the relevant information with each time-stamp and correspondent position. Finally, the data is visualized using QGIS.

5.3.1. Raw data processing.

This part of the code reads the raw data and extracts the information.

5.3.1.1. Reading binary files.

This code detects the satellites in the file, computes their CN0 and doppler shift, and stores the information in a file.

The samples in the binary files are put sequentially, first the real component (I) and then the imaginary component (Q). There is no separator between samples nor between components. However, knowing that each component has been saved as *int16* (two bytes), it can be separated when reading the file. One C/A code consists of 1023 chips at 1023 kHz, thus one millisecond. Since the sampling frequency is 2046 kSps, one millisecond has 2046 samples. For this reason, the file is read and stored in a matrix of size $N \times 2046$, where N is the non-coherent integration that will be applied.

The problem in the two-channel acquisition has been explained in 4.2.. To minimize the data loss, the buffers are the highest possible: 30 milliseconds (2046×30 samples). Every 30 milliseconds, an unknown amount of samples are lost, so to avoid integrating non-consecutive samples, it is imperative to always read the file 30 milliseconds by 30 milliseconds. Due to this problem, the non-coherent integration must equal Adalm-Pluto's buffer size.

Once averaged, we have a 2046 complex samples list.

5.3.1.2. Candidate satellites

GPS space segment consists on 32 satellites orbiting at an altitude of 20200 km (MEO), so their nominal period is 12h sidereal time (11h 58m 2s). The drone's flights will be maximum of 20 minutes, so not all the satellites will be above the horizon at the moment of the flight. The GNSS processing is computationally and time demanding, so to reduce the processing time makes sense to try the correlation with the satellites that are actually above the horizon. Those will be the candidate satellites.

The candidate satellites are chosen by the user and must be put in a list in the settings file;

those will be the satellites that the code will consider. By doing so, the processing time is dramatically reduced.

To select them, we need to know which ones were above the horizon at the moment of the flight. This can be done with a mobile phone app such as Google's *GnssLogger* that can show a sky-plot of the satellites in view detected by the phone in real-time. Another way to do it is using a web application such as *in-the-sky.org* [21], where the satellite's position at any time can be checked.

Fig. 5.2 is the sky plot on the day of the flight. The "candidate satellites" vector looks like this: `[1,7,8,10,16,21,26,27,32]`.

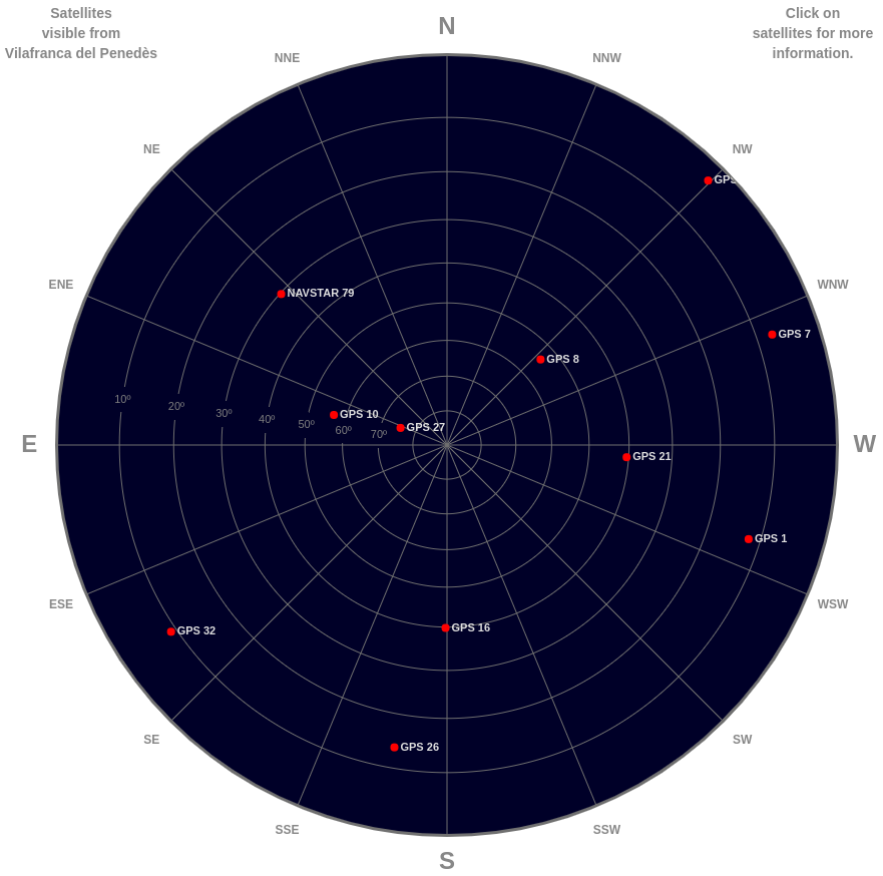


Figure 5.2: Skyplot at the moment of the flight.

5.3.1.3. Delay-Doppler map (DDM) computation

The central part of the GNSS-R processing is the computation of the correlation power as a function of time delay and frequency offset, a.k.a. DDM, and it is the main GNSS-R observable (as detailed in 2.2.0.7.). The DDM is a matrix whose axis is the delay of the signal and the Doppler shift produced due to the movement of the source and receiver. The definition of the axis of the grid is justified below:

DDM grid axis definition

The delay is represented in chips, where each chip last for $\frac{1}{1023kHz} = 977.5ns \sim 1\mu s$. The coherent-integration time has been decided to be 1ms and the sampling frequency 2×1023kHz. Therefore, the delay axis will have 2046 samples, each lasting half of a chip and totalling 1ms.

As justified in 5.3.1.1., the signal that will be processed is 30ms non-coherently integrated and reduced to a 2046 complex-samples vector. The L-band GPS signals present in the samples should be at baseband; however, they are not due to the SDR error and the Doppler effect. For this reason, it is needed to scan different Doppler frequencies to find the signals. The scanning range has been determined by computing the maximum expected Doppler shift. The Doppler shift is proportional to the relative radial speed between the transmitter and the receiver (Δv), the transmission frequency (f_0), and the angle between their movements (θ), as shown in equation 5.1.

$$\Delta f = \frac{\Delta v}{c} f_0 cos(\theta) \tag{5.1}$$

The Doppler shift is maximum when the angle θ is zero, meaning that the transmitter and the receiver are approaching in a straight line, and is minimum when the angle is 90º. Applying this to the GPS satellites, the Doppler shift is maximum when the satellite is close to the horizon. The angle can be computed using trigonometry as shown in Fig. 5.3, deducing:

$$\theta = arccos(\frac{R_{Earth}}{R_{Earth} + h_{sat}}) = arccos(\frac{6371}{6371 + 20200}) = 76,12 \tag{5.2}$$

The speed of the satellite can be deduced from the orbital period, and the speed of the drone is negligible, so the relative speed can be assumed to be the satellite speed:

$$\Delta v \simeq v_{sat} = \frac{2\pi(R_{Earth} + h_{sat})}{SiderealDay} = \frac{2\pi(6.371.000 + 20.200.000)}{11 \times 3600 + 58 \times +2} = 3,9km/s \tag{5.3}$$

The central frequency $f_0 = 1575,42MHz$, and the speed of light $c = 299.792.458m/s$. Therefore, the maximum Doppler shift is:

$$\Delta f = \frac{3900}{299792458} \times 1575.42 \times 10^6 \times cos(76.12) = 4916Hz \tag{5.4}$$

Note that these calculations are only valid for static receivers. If the receiver moves, then the Doppler shift will have higher values. This is the case in receivers mounted on CubeSats, where the Doppler shifts are in the range of $\pm 35kHz$ Since the drone will fly at very low speeds, the receiver is considered static.

The Doppler shift is positive if the transmitter and receiver are approaching and negative if moving away. So the Doppler axis of the DDM must go from 5kHz to -5kHz since the signal will be in this range (considering the receiver static).

Another critical parameter is the Doppler shift for each grid point (Δf). There is a trade-off for this decision; if Δf is too high, the signal may not be detected, but if it is too low, the

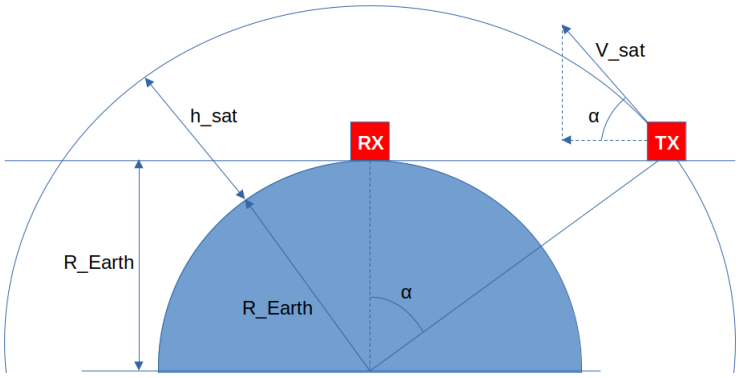


Figure 5.3: Compute the angle of maximum Doppler.

processing becomes less efficient. The frequency response of the correlation is a sinc-shaped curve, and increasing the coherent acquisition time (T_{acq}) narrows the sinc main lobe. A commonly used criterion is limiting the loss in correlation gain to less than 3dB, which is achieved when $\Delta f \leq \frac{2}{3T_{acq}}$. Since the coherent acquisition time selected is one millisecond, $\Delta f = \frac{2}{3\times1\times10^{-3}} = 666,67Hz$. Finally, it has been decided to use $\Delta f = 500Hz$.

Acquisition implementation strategy

There are two main ways to implement the GPS acquisition:

- The *code phase and carrier frequency serial search* implementation. This method is the most straightforward one since it performs the correlation operation of the signal in the time domain and for each frequency bin. It applies the DDM formula (equation 5.5) directly. The correlation in practice is a convolution between two signals.

$$Y^c(t,\tau,f_d) = \frac{1}{T_c}\int_t^{t+T_c} s(t')a^*(t'-\tau)e^{-j2\pi(F_c+f_d)t'}dt' \tag{5.5}$$

  Where Y^c is the correlation output for coherent integration time T_c and at the doppler frequency f_d between the signal s and the C/A code a, and t is the time when the integration starts. This formula is for continuous signals and C/A codes, but in our case, the signal is discrete. The continuous integral is changed by a discrete summation and is implemented by code. The process of summing and shifting all the 2046 samples in the vector is computationally demanding, which translates into long processing times. The other way to implement the acquisition solves this problem.

- The *Parallel Code Phase Search (PCPS)* implementation. This is the one used in the FMPL-D and almost in every GPS receiver, thanks to its enhanced efficiency. In this case, the correlation equation is applied in its discrete Fourier transformed form, and therefore, the convolution becomes a vectorial multiplication, as shown in equation 5.6.

$$Y^c(t,\tau,f_d) = \frac{1}{T_c}IFFT\{FFT\{s\}\times FFT\{a^*e^{-j2\pi(F_c+f_d)}\}\} \tag{5.6}$$

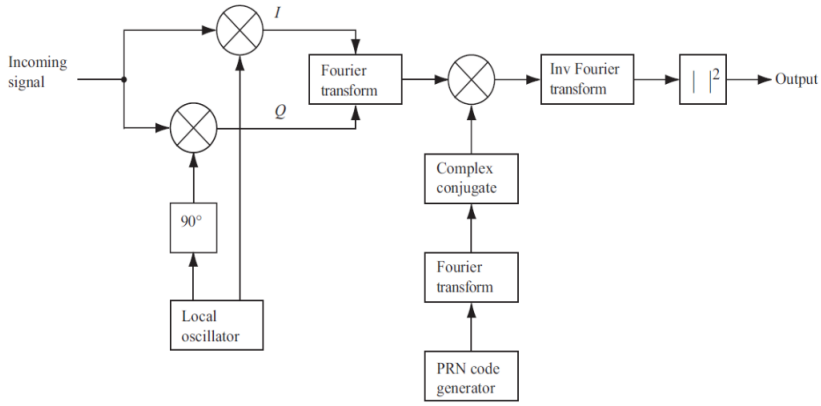


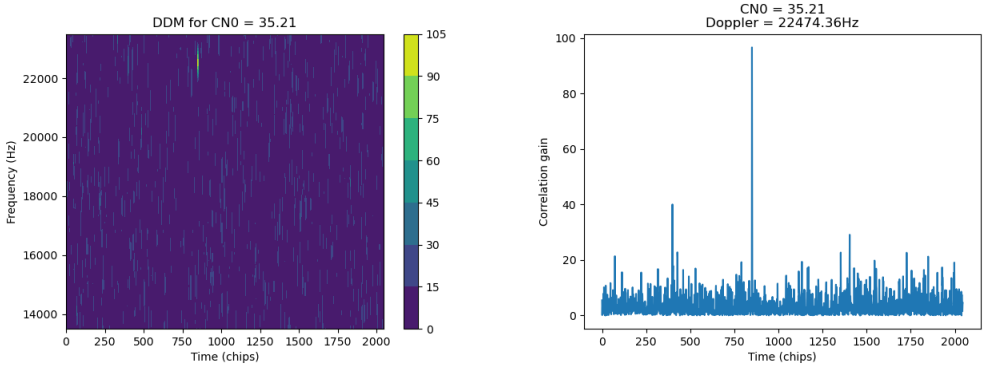
Figure 5.4: Parallel Code Phase Search (PCPS) implementation.

Fig. 5.4 is the block diagram the *PCPS* acquisition strategy implementation.

First, the signal is multiplied by a local oscillator centred at a Doppler frequency bin. In the GPS case, all the information is at the real component (I) of the signal, so from this point, unlike what Fig 5.4 shows, the imaginary part of the signal (Q) is discarded. Then FFT is computed on the signal and multiplied by the Fourier transform of the C/A code replica. The result is anti-transformed using the IFFT and then squared.

This is written in python and done by all the frequency bins (from -5kHz to 5kHz with 500Hz jumps); however, any correlation peak shows up. This is because, as mentioned before, the SDR has some bias in the local oscillator, and the signal is not centred at exactly zero Hz. The frequency at which the SDR is actually centred was empirically determined using the DDM code. The Doppler range was set from -50kHz to 50kHz, and the signals appeared. By running this code for a while for all the candidate satellites and looking for the average Doppler frequency, it was determined that the samples were centred at 18,5kHz. Therefore, the Doppler scanning is done from 13,5kHz to 23,5kHz with 500Hz jumps.

Fig. 5.5 shows the results of the DDM of a strong reflected signal. Fig. 5.5(a) is the DDM, and Fig 5.5(b) is the plot of the correlation gain as a function of time for the frequency bin with the highest peak. This is for a one millisecond coherent integration time and one millisecond incoherent integration time.



(a) DDM for all the scanned Doppler frequencies.

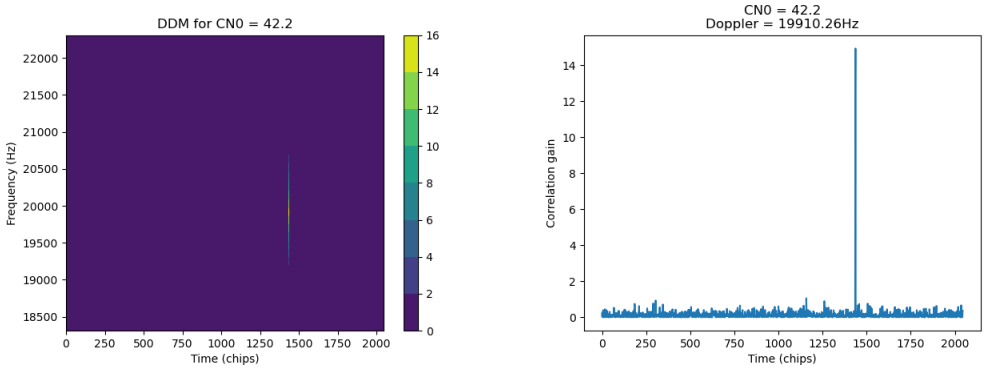
(b) Correlation result for the maximum Doppler frequency bin.

Figure 5.5: DDM of a reflected signal.

The plots above show a clear correlation peak since there was a high specular reflection in this case. However, in the right plot (5.5(b)), it can be seen that there are other high peaks produced by noise since the signal is very weak in amplitude, so the signal-to-(thermal) noise ratio is also very poor. A large number of incoherent averages (N_i) are required in order to improve the signal-to-noise ratio of $Y^c(t,\tau,f_d)$:

$$\langle |Y^c(\tau,f_d)|^2 \rangle \approx \frac{1}{N_i} \sum_{n=1}^{N_i} |Y^c(t_n,\tau,f_d)|^2. \tag{5.7}$$

The incoherent integration time selected is 30ms due to the problem in the data acquisition 4. The averaging has been applied directly to the samples and not to each 1ms correlation result because it makes the same effect of cancelling the random noise components. Fig. 5.6 shows a DDM obtained with an incoherent integration of 30ms, there is a great improvement.



(a) DDM for all the scanned Doppler frequencies.

(b) Correlation result for the maximum Doppler frequency bin.

Figure 5.6: DDM of a reflected signal.

Tracking of the signal

The acquisition method described before is highly efficient; however, there is room for improvement. Once one satellite has been detected with enough CN0, its Doppler frequency will evolve slowly, so it makes no sense to scan all the possible Doppler frequencies every time for every satellite. It is more efficient to reduce the Doppler ranges once a satellite is detected. In the FMPL-D, when a satellite is detected, the Doppler scanning range reduces from 10kHz (from 13,5kHz to 23.5kHz) to 4kHz (from Satellite Doppler + 2kHz to Satellite Doppler - 2kHz) and keeping the spacing between frequency bins (500Hz).

The computation of the DDM for one satellite lasts 60ms, reducing the Doppler frequency range to 4kHz it lasts 20ms, improving the efficiency by a factor of three.

5.3.1.4. CN0 computation

The main parameter for GNSS-R is the Carrier-to-Noise ratio (CN0) of the signal, and it can be computed from the DDM. The SNR has a problem evaluating the signal's quality.

It depends on the receiver's bandwidth, so different receivers would have different SNRs for the same signal power. For this reason, the parameter used to evaluate GPS's signal's quality is the Carrier-to-Spectral density Noise (CN0).

The CN0 is computed as follows:

$$CN_0 = \frac{SNR}{T_{int}} \tag{5.8}$$

Where the SNR is the ratio between the first and the second highest peak of the correlation result. The CN0 is expressed in dB. GPS is designed to have a nominal CN0 of 44dB on the Earth's surface.

5.3.1.5. Raw data processing code

Fig. 5.7 depicts the flowchart of the raw data processing code.

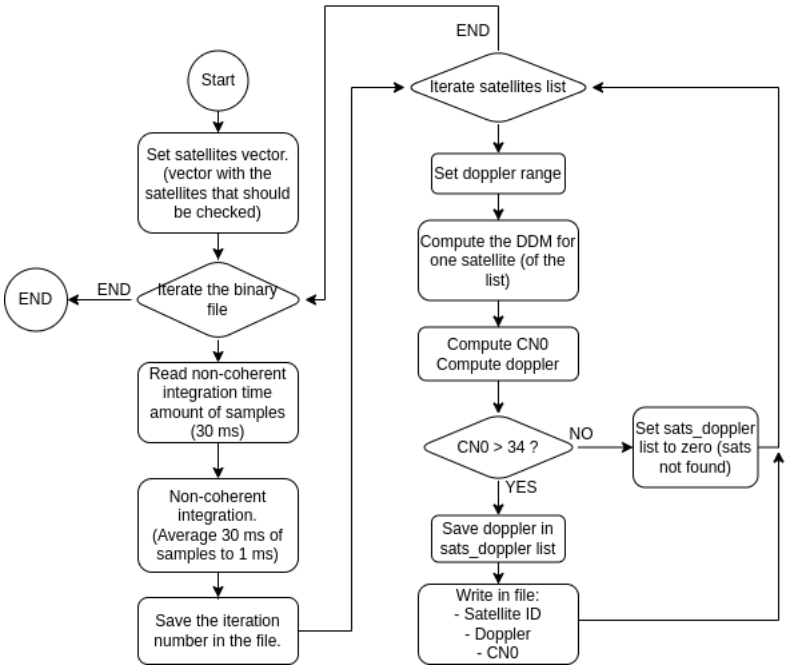
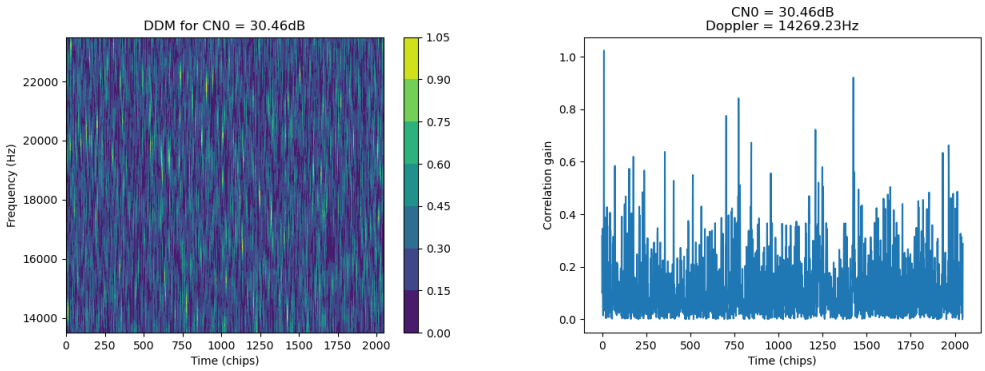


Figure 5.7: Flowchart of the main GNSS processing system.

Notice the condition at the end of the loop: `CN0 > 34`. It means that a satellite is considered detected if the CN0 after the correlation is higher than 34dB. It has been decided by observing the DDM at different CN0s. When the CN0 is lower than 34, the noise is so high that it can be confused by the correlation peak, leading to a false alarm. However, above 34dB, the peak clearly corresponds to a satellite. The following figures () show DDMs with different CN0 values:

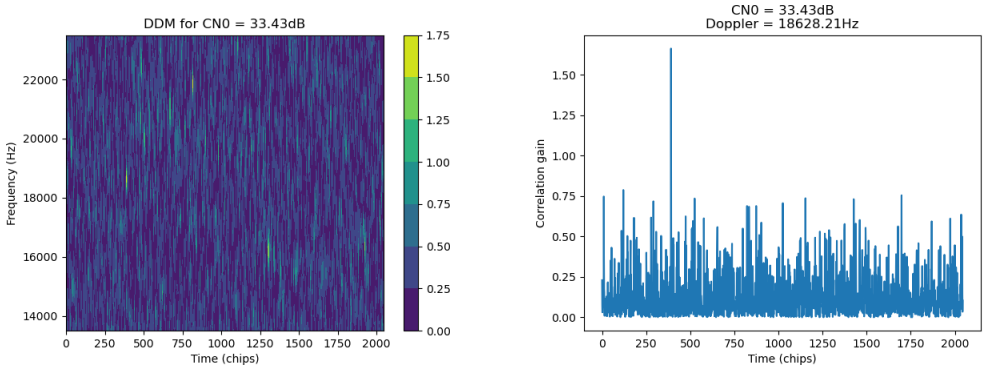
One DDM is computed every incoherent integration time milliseconds (30ms).



(a) DDM for all the scanned Doppler frequencies.

(b) Correlation result for the maximum Doppler frequency bin.

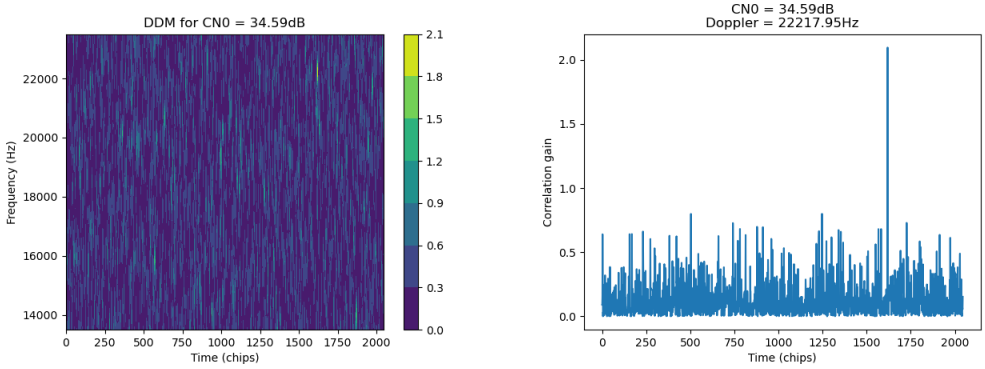
Figure 5.8: DDM of a signal with very low CN0.



(a) DDM for all the scanned Doppler frequencies.

(b) Correlation result for the maximum Doppler frequency bin.

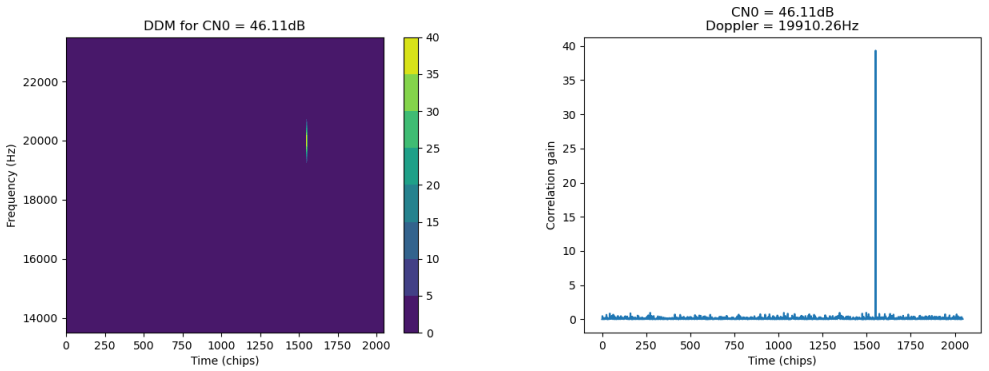
Figure 5.9: DDM of a signal with low CN0.



(a) DDM for all the scanned Doppler frequencies.

(b) Correlation result for the maximum Doppler frequency bin.

Figure 5.10: DDM of a signal with the minimum CN0.



(a) DDM for all the scanned Doppler frequencies.

(b) Correlation result for the maximum Doppler frequency bin.

Figure 5.11: DDM of a signal with a high CN0.

5.3.2. Correlation with the position

The second piece of code is the one in charge of correlating measurement DDM with a time-stamp and a position; it is also written in Python 3. The relevant processed data, such as the satellite ID, the Doppler shift, and the CN0 are stored in a file. This is one of the inputs of this code. The other input is a *.csv* file with the information coming from the GPS module, such as the position (latitude, longitude and altitude) and the time-stamp. Recall that the acquisition stores the start and finish time of the code execution, so the time-stamp of each DDM can be retrieved by interpolating. Then this time-stamp can be matched with the GPS time-stamp, relating the DDM with the GPS information.

The output is the *results.csv* file that contains the CN0 for the up-looking and down-looking signal, the position of the drone, and the position of the received specular reflection for each time-stamp. The computation of the specular reflection is done in another code explained in the next subsection 5.3.3..

Sometimes during the acquisition processing, the code loses track of a satellite and recovers it in the next iteration. When the satellite is lost, its CN0 is set to zero, so if the satellite was correctly tracked before, this measurement should be considered a mistake and thus, discarded. The GPS has a time resolution of 0.1 seconds (100ms), which is lower than the temporal resolution of the GNSS-R system (30ms), so three DDM measurements fit in one position label. From theDDM with the same position label, the ones whose CN0 is zero are automatically discarded, and the others are averaged. This is not a problem regarding soil-moisture results since the drone's speed is low, and the distance it could travel in 100ms is irrelevant.

5.3.3. Computation of the specular reflections

This piece of code complements the previous one. The output of the latter is the *results.csv* file with the time-stamp, the position of the drone, the position of the specular reflection, the direct-signal CN0, and the reflected-signal CN0. However, the other code has no information regarding the specular reflection position, so this code complements the table.

Recall how GNSS-R gets the soil's moisture information. It detects the reflected power of

a GNSS satellite and compares it with the direct signal. To give sense to the reflections information, it is imperative to know where they are coming from, which is computable. Given a transmitter and a receiver, the path that a signal would follow is deterministic. If a reflective surface is added, the signal will follow two paths, a direct and a reflected, and both would be deterministic. The NanosatLab team developed a Matlab code that computes the position of the specular reflection given two positions in Earth-Centered Earth-Fixed (ECEF) coordinates, considering the Earth as the WGS84 ellipsoid. This code has been used in the FMPL-2 payload onboard FSSCat CubeSat, among others.

So the inputs of the function are two coordinates; on the one hand, the drone's position, which is known and documented in the *results.csv* file. On the other hand, the position of each candidate GPS satellite, which must be computed. The GPS satellites continuously transmit their ephemeris from where their own positions can be computed. Some webpages such as *https://celestrak.org/* [22] publishes the ephemeris of all the GPS constellation. From there, the positions of all the satellites can be computed at any time.

As already mentioned, the Matlab code uses the WGS84 ellipsoid-shaped Earth approximation. A change in the receiver's altitude significantly changes the specular reflection point, so it is extremely important to consider the topography's altitude to obtain a correct result. Fig. 5.12 depicts the difference between the actual Earth's topography, the sea-level altitude (geoid), and the WGS84 ellipsoid.

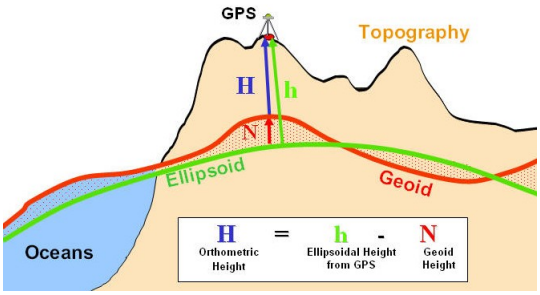


Figure 5.12: Comparison between different Earth altitude reference frames [23].

For simplicity's sake, the ellipsoid altitude is assumed to be similar to the sea-level value. The area's topography is supposed to be constant and set at the height above sea level given by the drone's GPS module at the take-off spot. Recall that GPS modules use the WGS84 ellipsoid to reference the altitude as well.

5.3.4. Plotting the results in QGIS

QGIS is an Open Source Geographic Information System (GIS) licensed under the GNU General Public License. It is a desktop app that makes it easy to analyze and edit spatial information and export graphical maps, so it is perfect to represent the GNSS-R results. It allows operations such as computing the distance between points, nearest neighbour interpolations, using SQL queries...

We aim to represent the acquired data stored in the *results.csv* file on a map, representing received power with different colours. This way, the results of all the work will be captured on a map with data points of different colours, each representing different soil moisture. The resulting maps are depicted and explained in chapter 6.

5.4. L-band radiometry

The radiometry system generates two files during the flight; the *calibration-pwr-acquisitions.txt* and the *mwr-pwr-acquisitions.txt*. As the names suggest, one stores the results of the detected power coming from the calibration loads, and the other the results of the power coming from the radiometer antenna.

This data is not ”as raw” as the one coming from the GNSS-R system, which was directly the binary samples. In this case, they are text files, each raw containing the computed power received with its components (IQ) separated. As explained in 4.5., the samples stored in the file are referenced to the SDR's full scale, and the way to extract physical information from it is by a calibration. The calibration process is detailed in **??**. The data processing considers the calibration values, setting its corresponding temperature to each calibration load. Since it has been assumed linearity between the power received and the physical temperature, with the two calibration loads, the transfer function is fully characterized using equation 5.9:

$$T_{Phy} = a \cdot P_{FS} + b \tag{5.9}$$

$$a = \frac{T_{HL} - T_{ACL}}{P_{FS_{HL}} - P_{FS_{ACL}}} \tag{5.10}$$

$$b = T_{ACL} - a \cdot P_{FS_{ACL}} \tag{5.11}$$

Where T_{Phy} is the physical temperature of the radiometer measurement, P_{FS} the power referenced to the SDR's full scale (value in the file), a the slope of the line, and b the ordinate at the origin. Recall the calibration results: $T_{HL} = 319.83K$ and $T_{ACL} = 158.45K$. This transfer function can translate the readings of the radiometer into physical temperatures.

5.4.1. Codes

The codes that process the radiometer's data are written in python. The objective is to obtain two plots:

- A plot of the physical temperature of the calibration loads and the radiometer antenna as a function of time. It will be used to see the evolution of the received power of the antenna with respect to the calibration loads, facilitating the detection of possible RFI.
- A map with the geolocated data points of the radiometer, the colour which will depend on the physical temperature. This plot is the actual result of the radiometer, showing the soil's moisture map.

The code for the first plot is straightforward. First, both files are read (*calibration-pwr-acquisitions.txt* and *mwr-pwr-acquisitions.txt*) and their powers stored in vectors. The radiometer captures data for 100 milliseconds before switching to the calibration loads. In

such a low time, the difference in soil's humidity will not be relevant, so it is wise to integrate all these samples to improve the precision. After the integration, the power values are translated into temperatures using the transfer function previously computed. The results are stored in the file *results.csv*, which has three columns; HL, ACL and MWR. The file is then plotted.

The code for the map uses the previously generated file (*results.csv*) and completes it with the drone's positions. Like in the GNSS-R case, the acquisition saved the data capture's starting and ending time of the data capture, and the positioning file has the GPS data. Combining this information, the results file can be completed. Once the file is completed, the data points are plotted on a map using QGIS.

CHAPTER 6. MEASUREMENT CAMPAIGN AND RESULTS

6.1. Introduction

Once the system has been developed and tested in the laboratory, verifying all its functionalities in the field is necessary. In this chapter, the measuring campaigns conducted are exposed. Their objective was to verify the whole system: Full Functional Test (FFT). Also, the results of the campaigns are analyzed, and conclusions are taken.

The selected area for the campaign was the Catalan vineyards (Alt Penedès) since it provides three important characteristics:

- It is an area with a low population density
- It is mostly flat. The area needed not to have different terrain altitudes since the data processing does not consider it.
- There are different types of soils with different humidity levels.
- Low presence of Radio Frequency Interference (RFI)s

Alt Penedès is a region in Catalonia in the province of Barcelona. The CTR of Barcelona's airport covers most of the region. Flying drones inside an airport's CTR is legal if the pertinent permissions are given; however, an area outside of it was selected to avoid bureaucracy. Fig. 6.1 depicts a map of the selected area. The red areas on the map are the ones where the airspace is restricted, and to fly the drone, permission should have been asked from the authorities. The biggest red zone corresponds to Barcelona's airport CTR. In purple is highlighted the selected spot for the measuring campaign, which is outside any restricted area.

To efficiently carry on the campaigns, the payload with one of the systems must be already integrated so that the first flight can start fast. Then the payload is mounted on the drone and connected to the power supply. The raspberry pi must be connected to the computer to start the acquisition code. To do so, we create a WiFi hotspot with a phone, the raspberry pi automatically connects to it, and we connect a computer too. Now the computer is connected with the raspberry pi using *ssh*, and the code is started using *nohup* (to make sure it keeps running once disconnected from the computer). Then the drone is flown following approximately the flight plan. When the drone arrives, it is connected to the WiFi hotspot, and the user ends the code by sending an interruption signal to the CPU (pressing Ctrl+C). Between flights, it is imperative to connect to the raspberry and change the output file names because if not, the acquisition code would overwrite them. Once the names are changed, if it is desired to fly this system again, the code can be started, and the drone can be flown again.

Once the first system is fully tested, we can jump to the second one. The payload change can be done while the drone's batteries are changed. When ready, the same procedure is repeated.

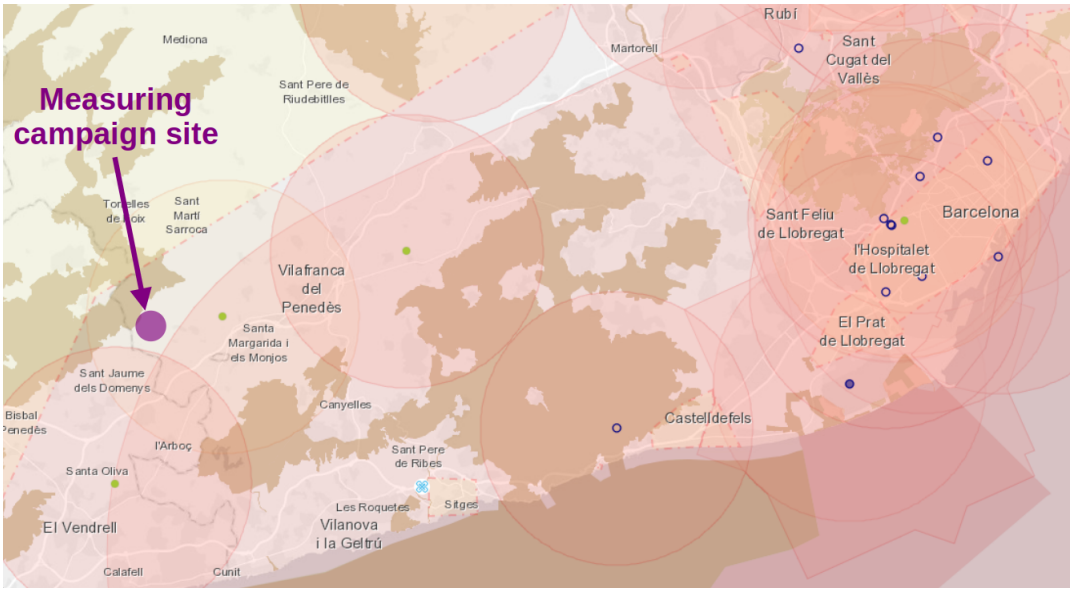


Figure 6.1: Selected area for the campaign. In purple is the selected spot. [24]

The main idea of the project was to perform a single measurement campaign. However, due to the complexity of the systems, problems arose and were solved in the subsequent campaigns.

6.1.1. Measuring campaigns carried out

1st campaign

In the first campaign's flight, the 3DR Iris drone was used 3.27. As the studies and tests confirmed, the drone could comfortably fly the payload, and some flights were performed throughout the morning. The drone flew four times, two flights for each system. The campaign seemed to have gone well until the data was analysed. We realized that the positioning data was not correctly captured; therefore, any captured data could be correlated with the ground's footprint, making the data meaningless. This issue should be fixed and tested in another campaign.

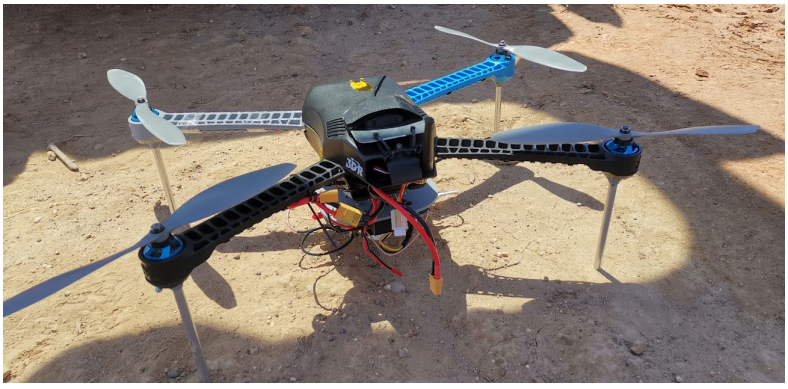


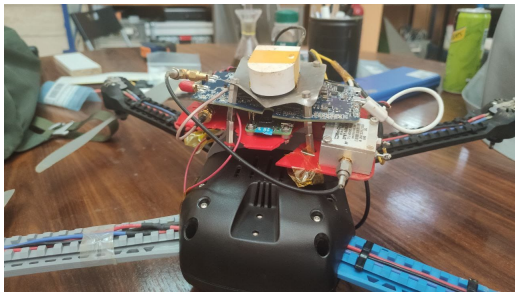
Figure 6.2: First measuring campaign: Iris ready for take-off.

Figure 6.3: First measuring campaign: the team.

2nd campaign

In this campaign, the issue with the positioning data and other minor bugs detected with the data from the first campaign were fixed. This day was extremely hot in the vineyard, and due to that, another issue arose. In the middle of the first flight, the drone started losing power and forcing a landing. After the visual inspection, the aircraft did not have any significant damage, so we decided to fly again. In the subsequent flights, the same happened with no significant damages. On the last flight, the drone overheated on-flight and fell from an unsafe altitude. A rock directly hit the SD card, which cracked and lost all the data (Fig. 6.4(b)). Although the payload seemed damaged (Fig. 6.4(a)), none of the components was harmed.

Even though the drone was not damaged, to prevent further problems, a drone operator company was hired: Mdrone [25]. Mdrone used the Condor Beta drone 3.32. Also, at this point, the second version of the payload structure was developed with the intention of securing the most critical components from future accidents.



(a) State of the payload after the accident.

(b) State of the SD card after the accident.

Figure 6.4: Second measuring campaign.

3rd campaign

In this campaign, the used drone was the Condor Beta, which is much more reliable and precise. Furthermore, it automatically saves the log files of the flights, so the positioning data was already being stored by the drone. Condor Beta permits more flights and longer. Each battery provides 40-minute flights, so we can perform four 20-minute flights in one campaign since we have just two batteries. This time, the campaign was successful. There were no incidents, and the data collected was meaningful. The results presented in the project are from this campaign.

Figure 6.5: The team of the 3rd measuring campaign with the Condor Beta drone with the payload integrated.

The following sections explain the measurement campaign approach for each system and the results.

6.2. L-Band Radiometry

6.2.1. Campaign

As detailed in 2.3., the radiometer measures the radiation emitted by the soil using a patch antenna with a low directivity. The only way to get a good spatial resolution is to keep the drone low so that the antenna beam covers a low surface (low footprint). The beam width of the main low of a patch antenna is around 90 degrees, so an altitude of 14 meters leads to a footprint of 10 meters. MWR can detect the power where the antenna is pointing, so the drone needs to fly over all the interesting areas. Since the main output of the radiometer is soil moisture, it is interesting that the flight covers areas that are known that is more humid than others. For this reason, the flight covers two different vineyards, an abandoned vineyard and a dirt road. It is expected that the road and the abandoned vineyard are drier than the tended vineyards. Fig. 6.6 depicts the path followed by the drone during the campaign.

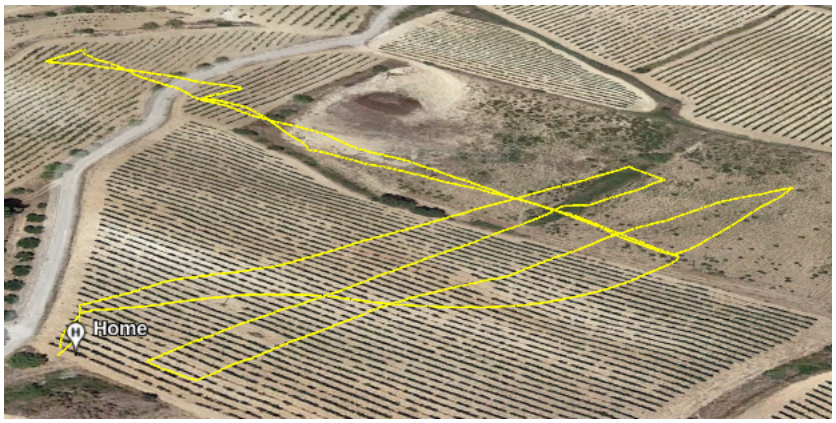
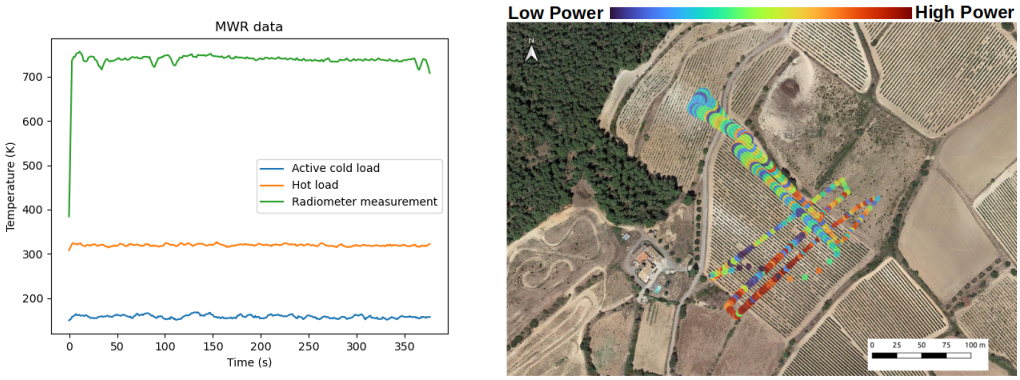


Figure 6.6: Path of the drone in the radiometer flight.

6.2.2. Results

Once the data is acquired and processed, it is time to analyze the results. Fig. 6.7(a) is the plot of the equivalent temperature of the two calibration loads (HL and ACL), and the radiometer measurement as a function of time (in samples). The expected result was that the cold load had low power, the hot load had high power, and the radiometer power was in between. The calibration loads worked as expected; however, the radiometer measurement is greatly out of the expected range, showing a temperature higher than the hot load. If the system works, the reason behind this phenomenon are the RFIs.



(a) Equivalent temperature of the two calibration loads and the radiometer measurement as a function of time.

(b) Received power of the radiometer along the drone's flight.

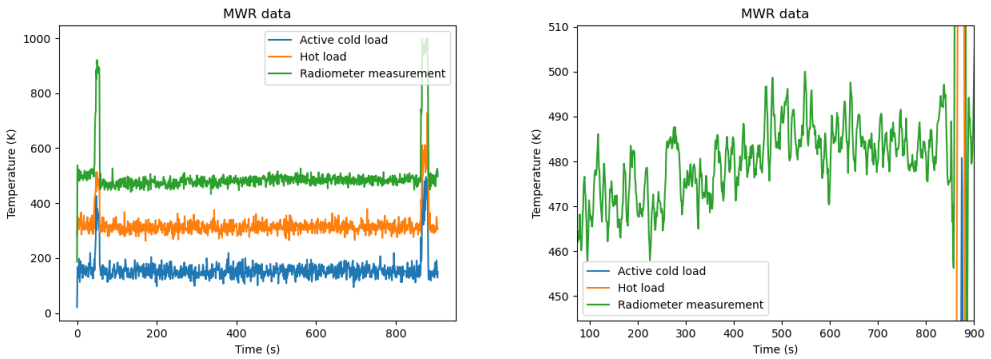
Figure 6.7: MWR results from the first campaign.

A test was conducted to determine the RFI sources that could produce the rise of power. As previously stated, the selected area is expected to have a low presence of RFIs; nevertheless, the RFI analysis system that will be implemented in future versions of the FMPL-D will characterize them in detail. Several studies on drone-based remote sensing suffer from the same problem, such as [26] and [27]. Based on these studies, in this project, it is assumed that the main RFI sources are the drone and the FMPL-D. The previously cited studies suggest that the main sources of RFIs are the CPUs, the USBs, and other electronic components present in the drone and the payload. One could think that the DC electric motors could also be an interference source, but those studies discard it. For

instance, a USB type 2 cable works at a frequency of 480 Mbps, so the 3rd harmonic falls exactly at 1440 MHz, right into the radiometer's band.

A static test was conducted to test the influence of the CPUs and the electronic devices on the radiometer's measurements. The payload was set in the radiometer configuration and mounted on the Condor Beta drone. Then, the acquisition was started with all the drone's systems turned off. Some seconds later, a laptop was put in the field of view of the radiometer's antenna (under the drone) for 3 seconds to see if the CPU produced a power peak. 5 minutes later, the drone's systems, such as the controller and telemetry, were turned on to check its influence. Finally, after another five minutes, the laptop was put under the drone again to get another power peak.

The result of this experiment is depicted in Fig. 6.8(a), showing two clear power peaks at the beginning and at the end of the measurements that correspond to the interferences produced by the computer's CPU and electronics. As in the previous results, the power of the radiometer's antenna is hotter than the hot load. By zooming in (Fig. 6.8), the increase of power corresponding to the drone's systems can be noticed; however, its influence is lower than expected.



(a) Result of the static test.

(b) Result of the static test, zooming into the radiometer antenna power.

Figure 6.8: Result of the static test, zooming into the radiometer antenna power.

Conclusion

To sum up, even though the RFI sources have different natures, the most common one for the radiometer's system is the actual system's electronics. Having seen the impressive influence of CPUs on the RFI contributions in L-Band, it is safe to conclude that the Raspberry Pi and the ADALM Pluto have the most impact. The high-speed data-transmission cables such as the USB also have an effect. The drone's systems also are non-negligible sources of interference.

To make the radiometer functional, it is imperative to control the RFI sources and minimize them as much as possible. For this reason, all the most influential interference sources must be shielded: the Raspberry Pi, the ADALM-Pluto, the USB connections, and all the system itself (except the antenna) to prevent RFIs from external sources (such as the drone). Due to time limitations, the shielding of the system will be implemented in a future version of the FMPL-D.

6.3. GNSS-R

6.3.1. Campaign

To decide on the flight plan for the GNSS-R system, it is necessary to analyze how it operates. Like for the MWR case, GNSS-R is crucial to minimize the spatial resolution of the measurements. As previously stated, for MWR this is directly proportional to the altitude of the antenna. However, for the GNSS-R case, the spatial resolution also depends on other factors.

When the signal is reflected on the ground, it is scattered following a Rayleigh distribution depending on the surface roughness. The higher the surface roughness, the higher the scatter's power will be. If the Rayleigh parameter of the rough surface is smaller than one, then the specular reflection is dominant over the scatters, and therefore, the spatial resolution will be limited to the first Fresnel zone. [28] approximates the first Fresnel zone with Eq. 6.1 as an ellipse, being the semi-major axis a, and the semi-minor axis b. Where the altitude of the transmitter (h_T) is the altitude of the GNSS satellite (which can not be changed), and the altitude of the receiver (h_R) is the altitude of the drone. The footprint of the GNSS-R system is this ellipse.

$$a = \sqrt{\lambda \frac{h_T \cdot h_R}{h_T + h_R}}; b = \frac{a}{cos(\theta)} \tag{6.1}$$

For a fixed carrier frequency and a fixed transmitter, the spatial resolution only depends on the receiver's altitude (drone) and the incidence angle of the reflected signal. Fig. 6.9 is a graphical representation of equation 6.1 fixing h_T and λ. Notice that the semi-major axis of the ellipse a only depends on the altitude, and it is equal to the semi-minor axis for an incidence angle of zero degrees (normal reflection). The ellipse's eccentricity increases with the incidence angle while keeping the semi-major axis constant, increasing the area of the ellipse, and worsening the spatial resolution.

The higher the drone flies, the more reflections with low incidence angles will receive, increasing the measuring range of the GNSS-R system. For this reason, flying at the highest possible altitude is desirable; the local regulation imposes 120 meters above take-off. At this altitude, the semi-major axis of the ellipse is 4.78 meters, and the semi-minor axis remains below 7 meters at incidence angles lower than 40 degrees. Which is a very good spatial resolution at such a high altitude.

The flight plan that the drone followed has two parts. First, the drone flew in circles at the highest possible altitude, 120 metres. Then, it followed the same circle but at a lower altitude, 50 meters. In this way, it can be seen the effect of the altitude in the received power and on the measuring range of the system. Fig. 6.10 shows the path that the drone followed.

6.3.2. Results

Once the data is acquired and processed, it is time to analyze the results. Fig. 6.11 depicts the results of the measurements taken in the campaign. On the one hand, Fig. 6.11(a) shows all the specular reflections with an incidence angle lower than 40 degrees

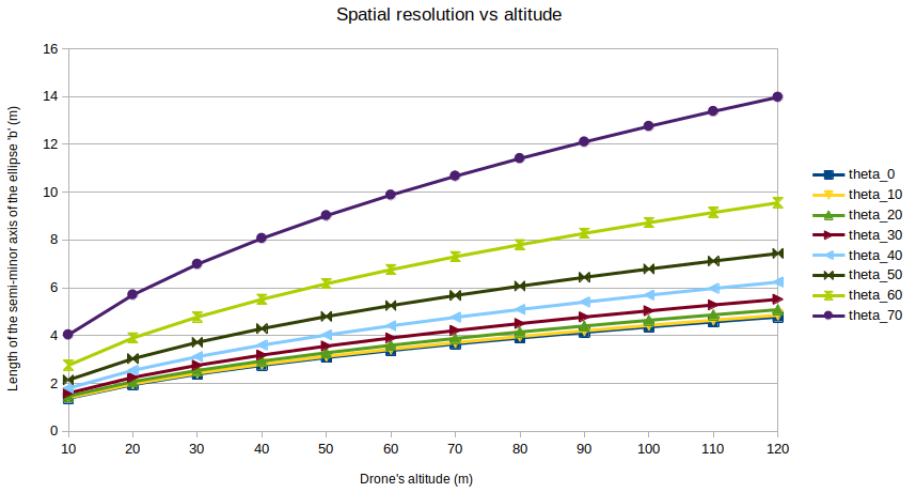


Figure 6.9: Spatial resolution as a function of the altitude and incidence angle.

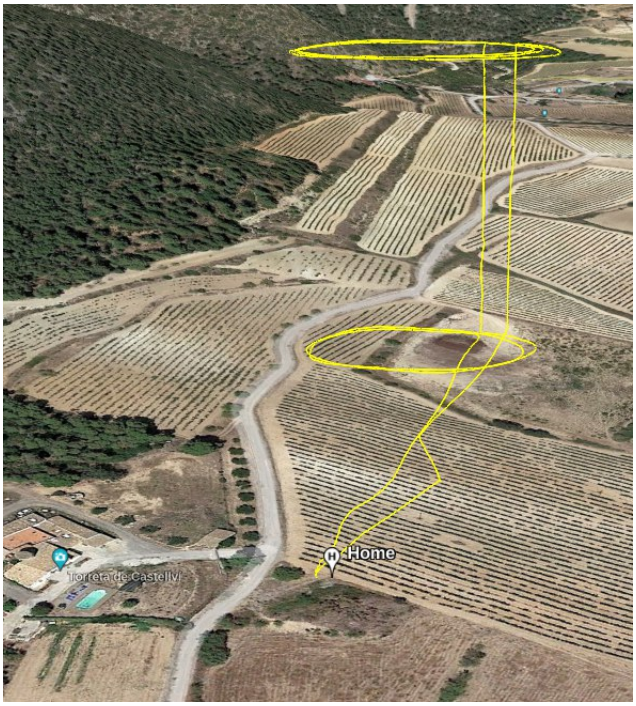


Figure 6.10: Path of the drone in the GNSS-R flight.

that should have been detected by the drone; in black, the path of the drone; in orange, the specular reflections of the flight at 120 meters height; in green, the specular reflections of the flight at 50 meters height. On the other hand, Fig. 6.11(b) depicts all the measurements detected by the system; the colours of the data points represent the power with which they were received. Notice that the data received during the manoeuvres, such as take-off, landing, climbing, and descending, have been discarded.

At first glance, it can be seen that the system was not able to get all the reflected signals because of their low power. As detailed in 2.2.0.5., the power reflected is related to the soil's humidity: the lower the water content, the lower the power. When the measuring campaign was carried out, there was a heat wave in the whole of Europe, and the area

(a) Reflections that should have been received. (b) Reflections that were received.

Figure 6.11: Results of the GNSS-R measuring campaign.

was suffering from a drought, so a very low water content was expected.

On the same day of the campaign, some soil samples were taken to measure their water content for reference. One sample for the cared vineyard and another one for the abandoned one. The process of measuring the water content from the samples is very simple. First, four samples of the same spot need to be taken, each one of these samples is weighted with a scaler, and the result is written down. The result of the four measurements is reduced to one number by averaging the median. Then, the samples are put in an oven for one hour to remove any water content. Finally, the samples are weighed again, and from the results, the median average is taken. The difference between the weight of the samples before and after ”baking” is the water content of the samples. It has been found that the water content is between 5 and 9 grams for both locations, which means a 0.071 %; a very low humidity.

The difference between the cared and abandoned vineyards can be clearly seen in the picture, and the difference in the received power is also evident. In the abandoned vineyards, the received power is much lower than for the cared ones because although the soil has almost the same water content, the vegetation is much more humid and provides high specular reflections. Unfortunately, the received power was so low in the drier parts that the system could not get a proper measurement. The reflected power is also lower in the forest on top of the picture than in the vineyard. The modelling of the specular reflections of the vegetation is an unresolved topic for further research. To measure in such dry areas, the system’s sensibility needs to be improved; it will be done in future iterations. Also, measuring over water to have high specular reflections would have been interesting. It will be done in the future.

CONCLUSIONS AND LESSONS LEARNED

In conclusion, the thesis has contributed to designing, building and testing the first version of a drone-based FMPL. Many problems that arise when using drones for measuring purposes have been pointed out. These are now lessons learned for future drone projects.

The requirements defined at the beginning of the project are now evaluated:

- **REQ-F-10. The FMPL-D shall make GNSS-R measurements.**
  - **Achieved.** The functionality of the GNSS-R system has been proven. However, further improvements must be made to have a fully-functional GNSS-R system that can measure soil moisture, for instance, better calibration.
- **REQ-F-20. The FMPL-D shall make L-band MWR measurements.**
  - **Not achieved.** In this case, the measurements of the radiometer were messed by the RFIs introduced by the drone and the components of the FMPL-D itself. Nevertheless, the advancements and the studies performed in this thesis are valuable lessons to fix it in the next version.
- **REQ-F-30. The FMPL-D system may integrate both GNSS-R and MWR in the same payload.**
  - **Not achieved.** During the project development, it was decided to start separating the systems in order to work with two isolated systems and reduce complexity. Regardless, the idea of integrating both systems into one has been kept in mind, and it will be done in the next version of the FMPL-D.
- **REQ-F-40. The FMPL-D structure shall be mounted under a drone.**
  - **Achieved.** For that, two designs were developed. The first one was lighter and simpler, but the components were too exposed to accidents. The second version is more compact and robust and will be the base of the next version of the FMPL-D.
- **REQ-F-50. The result of the flight campaigns shall be a soil moisture map.**
  - **Achieved.** The results from the flights were plotted on a map.
- **REQ-F-60. The FMPL-D shall store all the campaign measurements.**
  - **Achieved.** The system measures and stores all the data to be processed later on the ground.
- **REQ-F-70. The system may be capable of operating for at least 15 minutes.**
  - **Achieved.** The system is capable of measuring for a long time. The constraint is the drone's flight time.

On a personal level, this work has been a very deep learning experience from the engineering and personal point of view. I have had the opportunity to work on a multidisciplinary project in all its phases, from the system's concept to the actual tests in the field. From this project, I take home many lessons:

- *Lesson Learned 1*: Do not be too optimistic. If things are likely to fail, they will.
- *Lesson Learned 2*: Do not re-invent the wheel. Before doing anything, do some research about what others have done.
- *Lesson Learned 3*: Do not under-estimate RFIs; they are everywhere.
- *Lesson Learned 4*: Carefully write and plan a procedure before any test or measuring campaign.
- *Lesson Learned 5*: Have a clear plan for the project development.

Although this report marks the end of this thesis, I will keep supporting projects in the NanosatLab, and the new versions of the FMPL-D.

6.4. Future work

This project is the first step for having a full-functional FMPL-D. The errors that could not be fixed in this iteration will be solved in future versions. The two main errors that shall be corrected are:

- The problems with the L-band MWR RFIs.
- The sensibility of the GNSS-R. For instance with better calibration.

In the next version, the L-band MWR and the GNSS-R systems will be joined in one payload and simultaneously capture data.

The software interface to interact with the payload during the measuring campaigns will be improved.

All the processing chain pieces of code will be integrated into one to make the system more ”user-friendly”.

Another interesting feature would be that the payload sends real-time telemetry when the system is working. In that way, the user would be aware of the system’s health during the flights and would not have to wait until the data processing.

BIBLIOGRAPHY

[1] National Aeronautics and Space Administration. More satellites put in low-altitude orbits where they naturally burn up. https://www.esa.int/ESA_Multimedia/Images/2020/10/More_satellites_put_in_low-altitude_orbits_where_they_naturally_burn_up#.YlMa2qLEk2g.link. xiii, 3

[2] National Aeronautics and Space Administration. What are cubesats? https://www.nasa.gov/content/what-are-smallsats-and-cubesats, August 2017. xiii, 4

[3] Erik Kulu. Nanosats database. https://www.nanosats.eu/, 2022. xiii, 4

[4] NanosatLab UPC. 3cat-1. https://nanosatlab.upc.edu/en/missions-and-projects/3cat-1. 5

[5] J.F. Munoz–Martin, L. Fernandez, A. Perez, H. Park, J. Ruiz–de–Azua, and A. Camps. In-orbit validation of the fmpl-2 dual microwave payload onboard the fsscat mission. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 7884–7887, 2021. 6

[6] Balamis. Balamis technology remote sensing. https://www.balamis.com/technology/, 2021. xv, 7

[7] Insider intelligence. Drone market outlook in 2022: industry growth trends, market stats and forecast. https://www.insiderintelligence.com/insights/drone-industry-analysis-market-trends-growth-forecasts/, April 2022. 8

[8] A. Camps, J. F. Marchan-Hernandez, X. Bosch-Lluis, N. Rodriguez-Alvarez, I. Ramos-Perez, E. Valencia, J. M. Tarongi, H. Park, H. Carreno-Luengo, A. Alonso-Arroyo, D. Pascual, R. Onrubia, G. Forte, and J. Querol. Review of gnss-r instruments and tools developed at the universitat politecnica de catalunya-barcelona tech. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 3826–3829, 2014. 8

[9] A. Camps, H. Park, M. Pablos, G. Foti, C. Gommenginger, Pang-Wei Liu, and J. Judge. Soil moisture and vegetation impact in gnss-r techdemosat-1 observations. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1982–1984, 2016. 8

[10] A. Camps, J.F. Munoz-Martin, J.A. Ruiz-de Azua, L. Fernandez, A. Perez-Portero, D. Llaveria, C. Herbert, M. Pablos, A. Golkar, A. Gutierrrez, C. Antonio, J. Bandeiras, J. Andrade, D. Cordeiro, S. Briatore, N. Garzaniti, F. Nichele, R. Mozzillo, A. Piumatti, M. Cardi, M. Esposito, B. Carnicero Dominguez, M. Pastena, G. Filippazzo, and A. Reagan. Fsscat mission description and first scientific results of the fmpl-2 onboard 3cat-5/a. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 1291–1294, 2021. 8, 14

[11] Joan Francesc Muñoz Martin. *Development of novel instruments and techniques for passive microwave remote sensing*. PhD thesis, Universitat Politècnica de Catalunya - BarcelonaTech (UPC), July 2021. 8, 18

[12] J.M. Juan Zornoza J. Sanz Subirana and M. Hernández-Pajares. *GNSS data processing. Volume I: Fundamentals and Algorithms*. ESA Communications, ESTEC, PO Box 299, 2200 AG Noordwijk, the Netherlands, 2013. 11

[13] Bartosz Ciechanowski. Gps, 2022. [Online; accessed 18-September-2022]. 11

[14] Nereida Rodriguez-Alvarez, Adriano Camps, Mercè Vall-llossera, Xavier Bosch-Lluis, Alessandra Monerris, Isaac Ramos-Perez, Enric Valencia, Juan Fernando Marchan-Hernandez, Jose Martinez-Fernandez, Guido Baroncini-Turricchia, Carlos Pérez-Gutiérrez, and Nilda Sánchez. Land geophysical parameters retrieval using the interference pattern gnss-r technique. *IEEE Transactions on Geoscience and Remote Sensing*, 49(1):71–84, 2011. 14

[15] Carlos Molina. Gnss reflectometry. https://en.wikipedia.org/wiki/GNSS_reflectometry, 2022. xiii, 15

[16] Valery U. Zavorotny, Scott Gleason, Estel Cardellach, and Adriano Camps. Tutorial on remote sensing using gnss bistatic radar of opportunity. *IEEE Geoscience and Remote Sensing Magazine*, 2(4):8–45, 2014. 15

[17] GNU Radio Website. Gnuradio. https://www.gnuradio.org/, 2022. 26

[18] Hasini Viranga Abeywickrama, Beeshanga Abewardana Jayawickrama, Ying He, and Eryk Dutkiewicz. Empirical power consumption model for uavs. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018. 31

[19] rcplanes.online. Estimate propeller's static thrust. https://rcplanes.online/calc_thrust.htm, December 2018. 36

[20] Osmocom. rtl-sdr. https://github.com/osmocom/rtl-sdr, January 2019. 44

[21] Dominic Ford. Live world map of satellite positions. https://in-the-sky.org/satmap_radar.php?year=2022&month=7&day=28. 53

[22] Dr. T.S. Kelso. Celestrak. https://celestrak.org/, September 2022. 61

[23] Which of egm96 geoid or wgs84 ellipsoid fits the earth better? https://gis.stackexchange.com/questions/80533/which-of-egm96-geoid-or-wgs84-ellipsoid-fits-the-earth-better, 12 2013. xiv, 61

[24] Instituto Geográfico Nacional, Esri, HERE, Garmin, Foursquare, METI/NASA, and USGS. Enaire insignia. https://insignia.enaire.es/, 2022. xiv, 66

[25] Mdrone. Mdrone. https://mdrone.com/en/drone-based-technical-services-home/, 2022. 67

[26] RECENT RESULTS FROM P-BAND SIGNALS OF OPPORTUNITY RECEIVER DEPLOYED ON A MULTI-COPTER UAS PLATFORM. Mehmet kurum, preston peranich, md abdus shahid rafi, md mehedi farhad, and dylan boyd. *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2022. 69

[27] Marc Jou Barrancos. Development and testing of processing software for an airborne soil moisture mapper. Master's thesis, Universitat Politècnica de Catalunya (UPC), 2018. 69

[28] Adriano Camps and Joan Francesc Munoz-Martin. Analytical computation of the spatial resolution in gnss-r and experimental validation at l1 and l5. *Remote Sensing*, 12(23), 2020. 71

APPENDICES

APPENDIX A. SCHEMATICS OF THE NADS BOTTOM PCB

Although the visualization of these schematics is done in *KiCad*, the design was done in *Altium*.

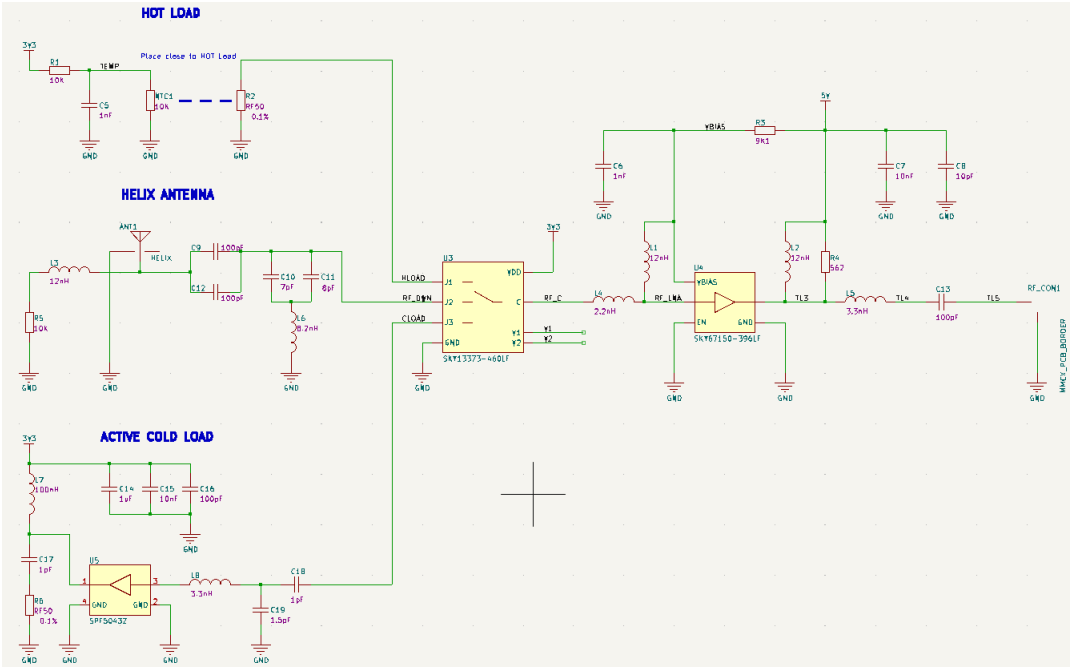


Figure A.1: Schematic of the NADS Bottom PCB.

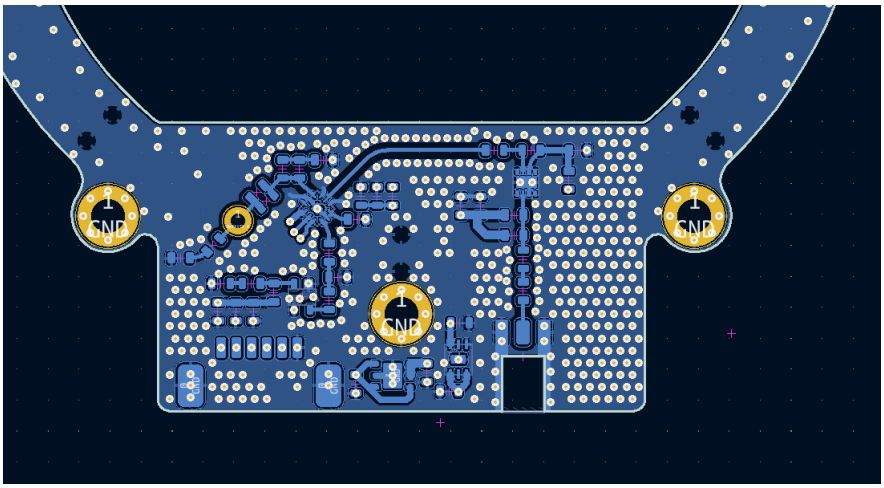


Figure A.2: PCB design of the NADS Bottom.

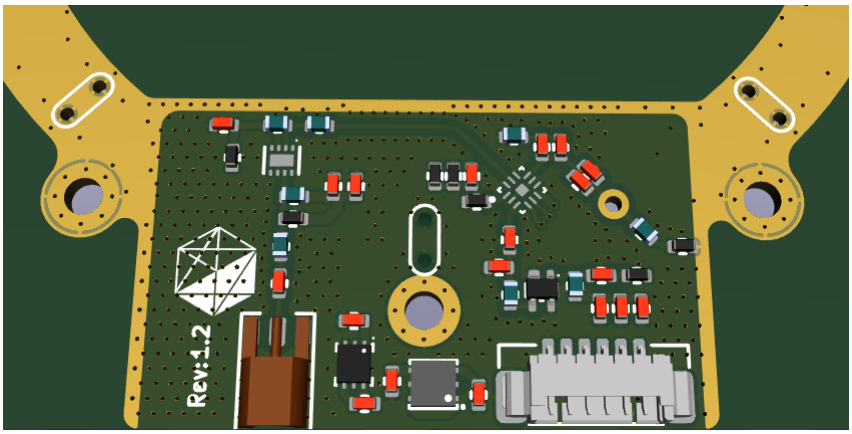


Figure A.3: 3D view of the NADS Bottom PCB.