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

MASTER THESIS

TITLE: An Open-Source Software-Defined Radio Collection for GNSS receivers

MASTER DEGREE: Master in Aerospace Science and Technology

AUTHORS: Joan M. Bernabeu Frias

ADVISORS: Prof. Dennis M. Akos

SUPERVISOR: Prof. Jaume Sanz Subirana

DATE: 5 de juliol de 2022

Títol: Col·lecció d'implementacións software de codi obert per a receptors GNSS
Autors: Joan M. Bernabeu Frias

Directors: Prof. Dennis M. Akos

Supervisor: Prof. Jaume Sanz Subirana

Data: 5 de juliol de 2022

Resum

La presente tesis tiene como objetivo recopilar el progreso y el esfuerzo dedicado al desarrollo y mejora de una colección de radios definidas por software (SDR) de libre aceso para receptores orientados a los sistemas de navegación por satélite (GNSS). La mayor carga de trabajo se ha centrado en someter a cada una de las implementaciones a una extensa campaña de pruebas, con el fin de determinar las características en cada una de las etapas del proceso de posicionamiento y realizar los ajustes pertinentes para alcanzar el desempeño definido. A lo largo del documento, se dan detalles de la estructura de la colección, los ajustes realizados para mejorar el desempeño en cada una de las implemenaciones, así como de las métricas consideradas para calificar los resultados obtenidos.

Title : An Open-Source Software-Defined Radio Collection for GNSS receivers
Authors: Joan M. Bernabeu Frias

Advisors: Prof. Dennis M. Akos

Supervisor: Prof. Jaume Sanz Subirana

Date: July 5, 2022

Overview

The present master's thesis has as a main objective to encompass the progress and efforts dedicated to the development and improvement of an open-source collection of Software-Defined Radios (SDR) for Global Navigation Satellite Systems (GNSS). Most of the efforts have focused in conducting an extensive test campaign for each implementation of the collection, with the aim of determining their characteristics at every stage in the positioning process, and make the required adjustments to achieve the defined performance. This document details the structure of the collection, the adjustments made to improve each code-base as well as the metrics considered to assess the obtained results.

I want to dedicate this work to Dr. Dennis Akos for having me in his team and giving me support during my stay at CU Boulder.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

GNSS	Global Navigation Satellite Systems
SDR	Software-Defined Radio
ION	Institute of Navigation
ITM	International Technical Meeting
PLL	Phase-Locked Loop
DLL	Delay-Locked Loop
ICD	Interface Control Document
I/Q	In-Phase and Quadra-Phase
PRN	Pseudo-random Number
PVT	Position-Navigation-Time
IDE	Integrated Development Environment

INTRODUCTION

With the text ”A Software-Defined GPS and Galileo Receiver, A Single-Frequency approach” by Borre K., Akos D.M., Bertelsen N., Rinder P., Jensen S., an open source Software-Defined Radio (SDR) receiver for Global Navigation Satellite Systems (GNSS) was developed in Mathwork’s Matlab coding language was made available, together with sample data sets that facilitated the testing process for all readers interested. The first SDR implementation focused on processing GPS L1 C/A legacy signal and served as a starting point for students and researchers at the RF and SATNAV Laboratory from University of Colorado Boulder, where later activities aimed to improve the code-base and add new features as new GNSS signals emerged. In result, the initial code-base evolved into a set of SDRs capable of processing all open-service GNSS signals from every satellite constellation.

CU Boulder’s RF and SATNAV laboratory decided to make this initiative an open-source project in the shape of a collection of SDRs, which could be accessed by users all over the world. Before making the collection available, initial efforts where dedicated to put the code-bases in a common format and test each implementation to give account of their performance. This work generated the paper [1], which was presented at the International Technical Meeting (ITM) conference organized by the Institute of Navigation (ION) in February 2022 in Long Beach, California (USA). After this, more progress was achieved in understanding and in some cases improving the results observed for certain SDRs. This master’s thesis can be interpreted as an expansion of the project’s initial paper. On the one hand, it deals with less format constrains and therefore is able elaborate more in most aspects. On the other hand, is will also report the progress achieved since, which affected the overall collection performance.

The contents of this document are distributed in 4 chapters, a conclusion and an appendix. Chapter 1 gives an overview of the contents of the SDR collection, the coding structure and the tasks the major stages perform. Then, Chapter 2 presents the resources that provide support to the SDR collection, from the SDR code-bases to the sample datasets and the discussion forum. Chapter 3 describes the testing framework designed to assess every code-bases in the collection, covering the equipment used and the approach followed to interpret test results. Next, Chapter 4 describes the debugging process some SDR were submitted to when they did not provide results meeting the metrics defined. The last chapter summarizes the most important points made and extracts conclusions about the final results of the SDR collection. Finally, an Appendix is included at the end which provides complementary information of the structure and processing approach followed by the code-bases in the collection.

CHAPTER 1. GNSS SDR COLLECTION

1.1. An open-Source Collection

The whole SDR collection was developed in Mathwork's Matlab programming language. Thus, users interested are recommended to have an active Matlab's licence, so they can better explore the code leveraging the integrated debugger of Matlab's IDE.

This collection was intended to be an open-source project, that is, all the code involved in the processing of GNSS signals will be publicly available for users via the defined sources. In addition, sample data-sets are provided to accelerate the testing process for those users that do not have GNSS data at hand.

1.2. Structure of the SDR Collection

There are a total of 12 SDR code-bases, each of which is aimed at processing a particular GNSS signal. These classified according to the GNSS they belong to in table 1.1.

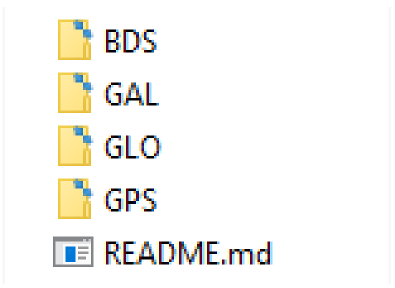
GPS	**Galileo**	**BeiDou**	**GLONASS**
L1CA	E1C	B1C	GL1
L2C	E5a	B1I	GL2
L5C	E5b	B2a	
		B3I	

Table 1.1: This table presents all the GNSS signals that can be processed by the SDR collection. The satellite constellations they belong to are written in bold letters.

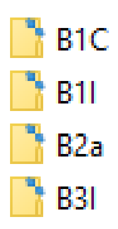
All 12 code-bases of the collection follow a common structure in terms of files and directories. Some count with additional files, i.e. Matlab Scripts, to perform tasks that are specific to a GNSS signal. An example of this is GPS L2C SDR, where *"//include"* contains additional functions aimed at exploiting the pilot component of this GNSS signal, if defined in the corresponding *"initSettings.m"* script.

The common structure is showed in figure 1.1, where it can be seen that SDRs are arranged in constellations, and that for every SDR directory, there are two sub-directories and two scripts. These encompass all the logic required for the SDR code-base to operate. The structure of SDR directories is described next:

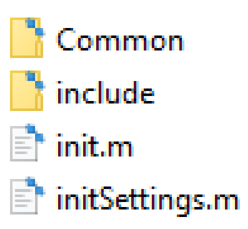
- //**common:** Contains functions that are common across different code-bases, such as those performing coordinate conversions, plot generation, positioning algorithms etc.
- //**include:** Includes functions developed to exploit the characteristics of a particular GNSS signal.



(a) ".//"



(b) ".//BDS"



(c) ".//BDS//B1C"

Figure 1.1: These figures illustrates the directory structure of the SDR collection. Figure 1.1a shows the main directory. Then, figure 1.1b and 1.1c show *".//BDS"* and *".//BDS//B1C"* sub-directories.

- **init.m:** Matlab script used in all SDR implementations to run the code-base in question. When ran, Matlab will load the settings defined and excecute the defined stages of the SDR.
- **initSettings.m:** This script defines the parameters considered in the SDR execution.

1.3. Code-base structure

The goal of this section is to introduce the most important receiver stages implemented in all code-bases from the collection.

While the general SDR receiver structure is similar across code-bases, each comes with a set of different features to adapt the code to the characteristics of a specific signal. The tasks performed by every SDR implementation can be classified in four different major modules, oftentimes refereed to as "stages" in GNSS literature. These are represented in Figure 1.2.

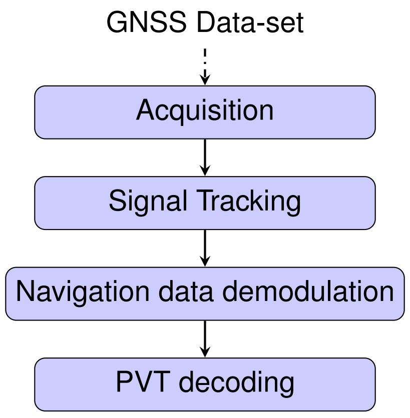


Figure 1.2: Figure showing common GNSS processing stages of the SDR collection.

The next subsections give an overview of the general approach followed by the main stages of all SDRs.

1.3.1. Signal Acquisition

This Signal Acquisition stage is the part of the code-base where more efforts where put during the debugging process to improve the results obtained initially. The design of the acquisition scheme depends on the characteristics of the signal the SDR is aimed to process. There are numerous GNSS signal configurations for each constellation that follow different strategies concerning spreading codes, navigation data and secondary codes. These must be accounted for in the acquisition code-base. A more in-depth description about the acquisition stage can be found in [2], Section 6.

1.3.1.1. Frequency Space

All code-bases follow a FFT-accelerated serial search acquisition approach ([2], Section 6.2) to obtain estimates of the signal's carrier frequency and code-delay, where a number of signal replicas are generated iteratively for each Pseudo-random Number (PRN), separated by a frequency distance. These replicas are correlated to the input signal, aiming at finding a high correlation value, meaning the signal is present in that fraction of data-set.

The frequency offsets are arranged in what are known as frequency bins. This frequency separation will be referred to as frequency step from now on. The frequency step is inversely proportional to the integration time and tells the maximum error allowed in the carrier frequency estimate, which is half the frequency step. In other words, in order to keep the carrier frequency estimate error below a defined threshold, the frequency step must be adjusted so that it is as much as twice the acceptable error. Both the frequency step and the coherent integration time are parameters that have a strong effect in acquisition results, as Section 4.2.1. shows.

1.3.1.2. Correlation process

The duration of the correlation between the local replica and the input signal is the so-called coherent integration time. This parameter can also be referred to as correlation length, if it is interpreted in terms of samples, instead of time. The maximum correlation value from all frequency bins is then divided by the second maximum found. This ratio is called peak metric, and is used in all SDRs to give a measure of the difference in magnitude between the maximum value obtained and the remaining correlation results. If the peak metric is not high enough, it implies that the maximum is close to other cross-correlation products, and so could not correspond to the result obtained after correlating both input and local replica signals with the right code-phase alignment. When the peak metric surpasses the threshold defined in ”initSettings.m”, under ”settings.acqThreshold”, the given PRN is considered to be acquired.

It is important to note that in all SDR implementations the local replica is constructed by concatenating a whole primary code and a block of zeros of the same length. This prevents navigation bit transitions from affecting the correlation results.

1.3.2. Signal Tracking

The tracking stage is oriented at refining and keeping track of the code and carrier estimates provided by the acquisition stage, as well as at demodulating the navigation data. This receiver stage was the one that required less adjustments for this project, as it had been well validated and tested in previous projects and implementations. The information provided next follows [2], section 7, and introduces the main ideas behind the Tracking Stage of the SDR collection.

1.3.2.1. Feedback Loops

GNSS signals are tracked using feedback loops organised in channels, which are typically referred to as tracking channels. The SDR receiver could run a tracking channel for each acquired satellite, but in practice, the number of tracking channels is limited to reduce the computational burden of this stage. This feature is considered in the SDR and can be modifyied in *initSettings.m* under *settings.numberOfChannels* parameter.

Each tracking channel makes controlled adjustments to the carrier-frequency and code-phase estimates delivered from the Acquisition stage, changing the locally generated signal replica for the given satellite, so that it resembles the input signal as much as possible. When the replica is sufficiently accurate, the tracking loop ”locks” to the signal, removes the carrier and spreading code components, and starts registering data bit transitions. The task of every tracking channel is to account for signal variations so that they can keep locked for as long as the satellite is available for use.

1.3.2.2. Tracking parameters

Tracking channels implement two feedback loops, the Delay lock loop (DLL) and the Costas Loop ([2], Section 7.3). The former is focused on the signals' code phase while the latter on the carrier phase . These modules depend on two major parameters that determine the properties of the loop filter: the damping ratio and the noise bandwidth. On the one side, the damping ratio controls how fast the filter reaches the settling stage. On the other side, the noise bandwidth tells the amount of noise allowed in the filter.

Both the damping ratio and the noise bandwidth for the Costas and Delay lock loops can be configured in the *initSettings.m* file. The parameters defined to do so are:

- *settings.dllDampingRatio*
- *settings.dllNoiseBandwidth*
- *settings.pllDampingRatio*
- *settings.pllNoiseBandwidth*

Where *”dll”* refers to the delay lock loop and *”pll”* to the Costas phase lock loop.

While all SDRs follow similar tracking loop schemes, some, such as GPS L2C SDR, need some adjustments in the parameters described above so that they provide adequate results, as it is pointed out in Section 4.2.1..

Tracking results are stored in a Matlab's *'.mat'* file, but also can be assessed in the figures generated by the tracking stage after the processing of all the channels is over. Figures 1.3a and 1.3b show an example of two different tracking results plots, each of which include seven figures. These show the In-Phase and Quadra-Phase (I/Q) prompts ([2], Section 1.5), the demodulated navigation data bits, the changes in the Raw/Filtered Phase-Locked Loop (PLL) and Delay-Locked Loop (DLL) discriminator, as well as the Early-Prompt-Late metric. All of these metrics are described in more detail in the reference noted at the beginning of this section. Note that the plots from Figure 1.3a suggest the navigation data bits were demodulated successfully, as bit transitions can be observed and the early-prompt-late correlation results follow the expected trend. In contrast, the set of plots in figure 1.3b show a tracking channel which failed to demodulate the navigation message, and as a result, navigation data bits cannot be distinguished.

1.3.3. Navigation Data Demodulation

This stage is where the navigation data required by the positioning algorithm is extracted from the results delivered by the tracking stage. This information is known as satellite Ephemeris, and contain the parameters required to compute satellite positions within their orbits, so they can be used to solve the positioning equation. The latter outputs I/Q prompt samples representing data bits, containing navigation data encoded to some extent. The navigation data format for each signal can be found in the Interface Control Document (ICD) of every constellation.

The general process each SDR implements to extract navigation data from I/Q prompt samples is summarised bellow:

1. Detect a preamble within data bits (Either pre- or post-decoding)
2. Arrange the bit sequence in the specified structures (e.g. frames, pages etc).
3. Account for the secondary code if present.
4. Decode and (if required) de-interleave
5. Check the bit stream is free of errors.
6. Extract navigation parameters

Once navigation parameters are extracted, they used by the functions involved in the PVT computation stage, to compute satellite positions.

1.3.4. PVT Computation

The Position-Navigation-Time (PVT) stage takes the navigation data decoded, computes satellite positions, and solves a geometrical problem, whose solution is the receiver's location. This procedure is described in detail in [2], Section 8.

As is the case with the other stages, all SDRs follow the same approach and use the least-squares ([3], Appendix A) method to solve for a position estimate, once all the required data is available. Position estimates are delivered in both ECEF ([3], Section 2.2.2) and ENU ([3], Section 7.7) coordinates.

Similarly to the tracking stage, the PVT stage generates a plot showing some PVT statistics to give the user an insight of the positioning performance. Figures 1.4a and 1.4b, show an example of two positioning plots obtained for two different data files. These contain figures showing the evolution of the navigation solution in time and over a 2D plane, as well as a sky-plot illustrating the satellite geometry with respect to the user. In Figure 1.4a, for instance, the evolution of the navigation solution with time is contained within a rather small threshold, in comparison to Figure 1.4a, which denotes irregular changes in the same plot. Note that plots such as Figure 1.4a do not imply the navigation solution is accurate, as they only provide information about the statistics of the results obtained.

1.4. Code-base execution

All of the SDRs in the collection have been developed to work with a baseline configuration, in other words, with a default *initSettings.m* script. The value selected for the parameters defined in this script will have an effect on the obtained results. The *init.m* script runs the SDR by executing the functions contained in *//include* and *//common*. Users are able to follow the progress via the Matlab's command line, as it will prompt informative messages after the execution of the receiver's main stages.

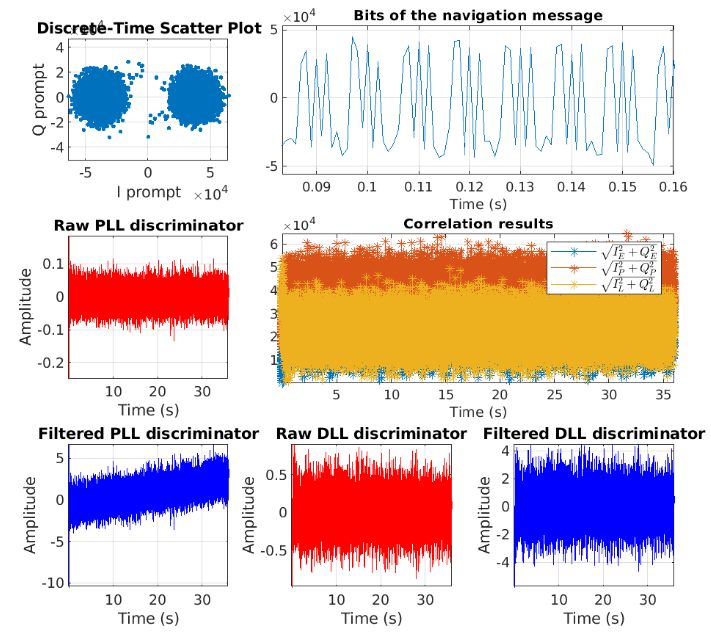
The workflow common to all SDRs in the collection can be summarized in the following steps:

1. Set-up *initSettings.m* with the desired configuration.
2. Run *init.m* to execute all the SDR stages.
3. Results are stored by default in the SDR's main directory.

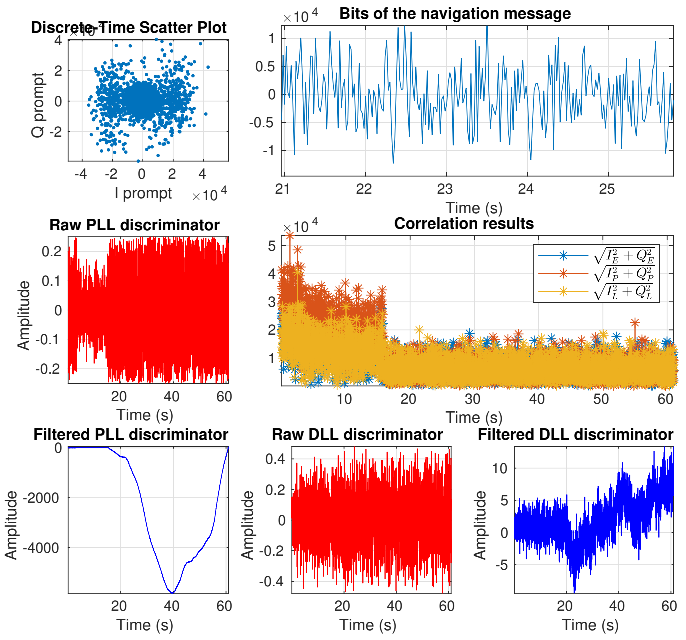
Note that *init.m* is a Matlab function that expects a parameter called *filename*. The latter is the path to the selected I/Q data-set with GNSS data, whose format is described using Institute of Navigation's SDR meta-data standard [4]. The easiest way to call *init.m* function is by typing in Matlab's command prompt showed in the 1.1. Note that Matlab workspace must be pointing at the directory encompassing the SDR functions.

```
>> init(path//to//binary.bin)                    (1.1)
```

To get a hands-on feel of the way SDRs operates, users are encouraged to run a code-base using a sample data-set. These can be accessed via the CU Boulder portal in [5].

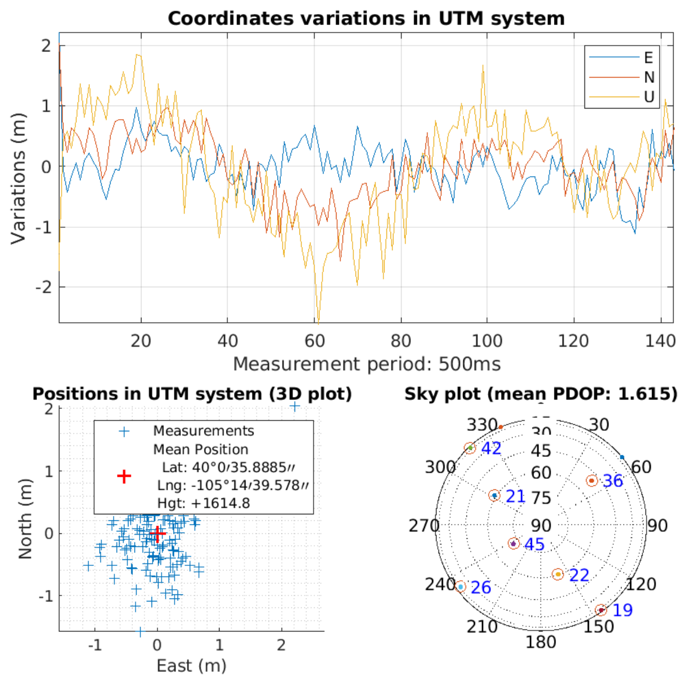


(a)

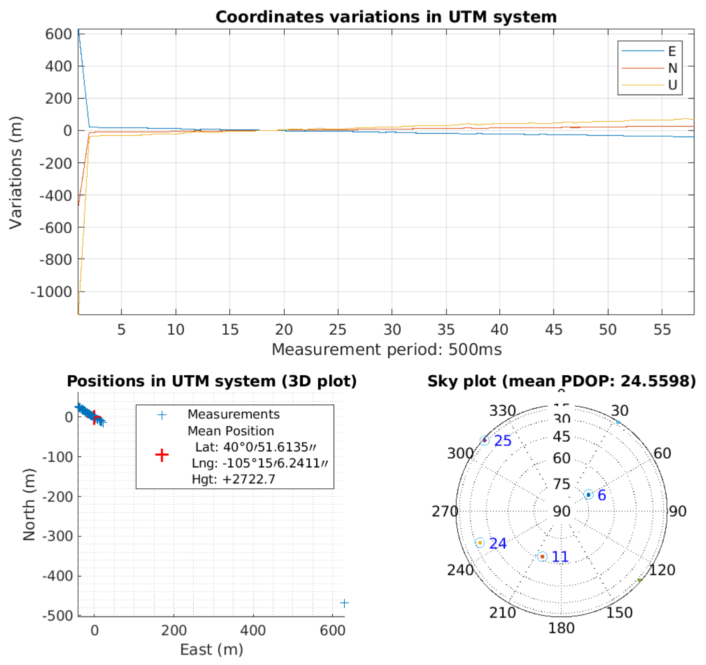


(b)

Figure 1.3: Figure 1.3a shows the plot generated for a successful tracking channel. In contrast, Figure 1.3b, illustrates the results obtained when the tracking loop in question did not lock appropriately to the signal and therefore was not able to demodulate navigation data.



(a)



(b)

Figure 1.4: Figure 1.4a shows the plot of what seem rather good statistics for the navigation solution. Figure 1.4b shows a navigation plot for a file that presented a problem affecting the PVT solution.

CHAPTER 2. PROJECT RESOURCES

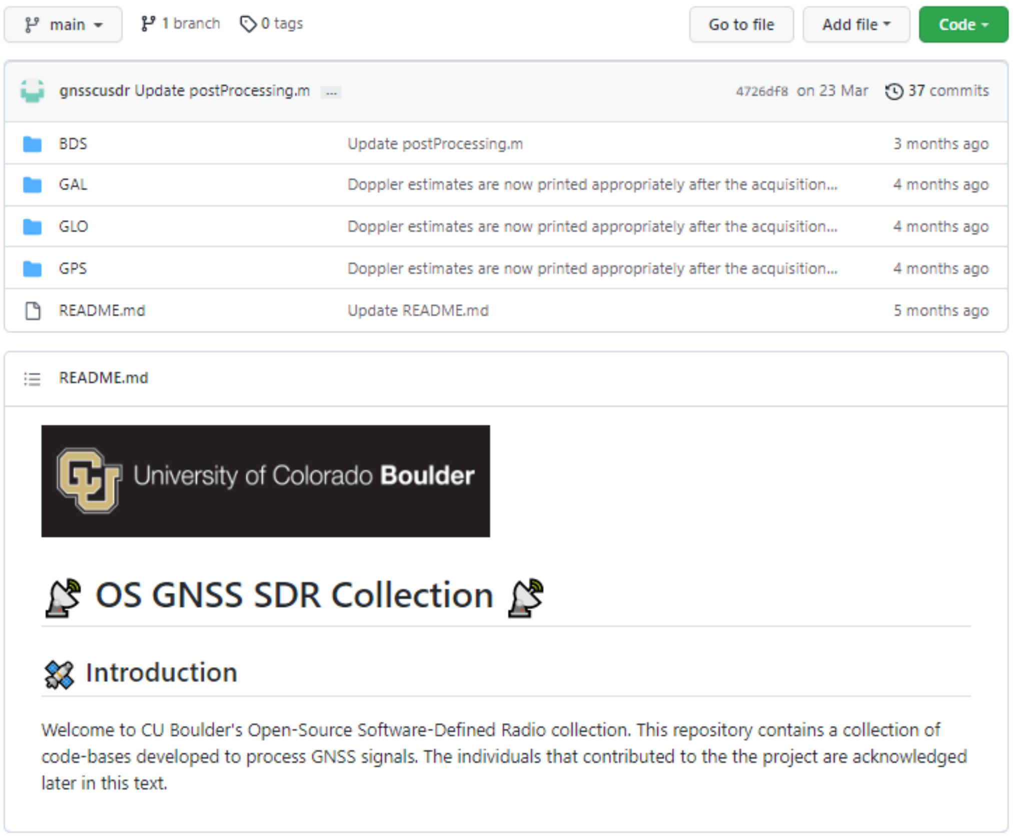
2.1. Overview

The previous chapter presented the contents of the SDR Collection and described how to execute code-bases. This section talks about the resources that support this project as an open-source initiative. Amongst these elements are a GitHub repository with all SDR code-bases, a Google Drive with Sample data-sets, and a Slack Discussion forum. All can be accessed via CU Boulder's portal in [5] and are described in the following sub-sections.

2.1.1. GitHub Project Repository

The project's GitHub repository was intended to provide users the ability to not only download the elements of the SDR collection at will, but also keep track of the changes and progress of the project. This approach was also convenient to allow user contributions in the future via Pull-requests ([6] Section 6.2) with improvements and/or new features.

Figure 2.1 illustrates the front page of the project's repository, which follows the same folder structure introduced in Section 1.2. Users can decide whether to download the whole collection or only the directories of interest.



gnsscusdr Update postProcessing.m … 4726df8 on 23 Mar 37 commits

BDS	Update postProcessing.m	3 months ago
GAL	Doppler estimates are now printed appropriately after the acquisition...	4 months ago
GLO	Doppler estimates are now printed appropriately after the acquisition...	4 months ago
GPS	Doppler estimates are now printed appropriately after the acquisition...	4 months ago
README.md	Update README.md	5 months ago

README.md

University of Colorado Boulder

OS GNSS SDR Collection

Introduction

Welcome to CU Boulder's Open-Source Software-Defined Radio collection. This repository contains a collection of code-bases developed to process GNSS signals. The individuals that contributed to the the project are acknowledged later in this text.

Figure 2.1: Screenshot showing the front page of the project's GitHub repository.

There are several ways to download the code to perform tests. One recommended way is to fork the repository in Github and clone it locally using git [6]. This way the fork, i.e. copy of the repository, is automatically linked to GitHub, which proves useful to track the changes made. Users can clone their fork of the project by using Git and the HTTP link of the repository, which is found in the ”Code” tab. An example of this is:

```
>> git clone https://github.com/gnsscusdr/CU-SDR-Collection.git
```

Another way to download the collection is via the GitHub Web platform. This is done by clicking the ”Code” tab and then selecting the ”Download ZIP” option. This is illustrated in figure 2.2.

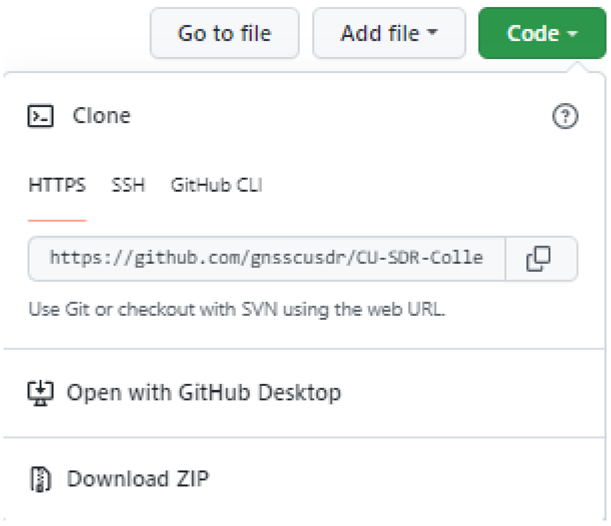

Go to file
Add file
Code
Clone
HTTPS
SSH
GitHub CLI
https://github.com/gnsscusdr/CU-SDR-Colle
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP

Figure 2.2: Screenshot showing how to manually download the project’s repository

Once the SDR Collection is downloaded, its contents can be explored either using Matlab’s IDE or any other source code editor. However, only Matlab will let the user to run the code and set breakpoints of interest.

2.1.2. Google Drive with Data-sets

All SDRs take as input binary files containing GNSS data. Users have the option to use their GNSS files in the format described in Section 3.2., and process them with the SDRs of interest.

Another option to conduct tests is to make use of the sample data-sets this project made publicly available in a Google Drive. As Figure 2.3 shows, 8 different data-sets can be downloaded allowing to run all SDRs in the collection. The reader is referred to Section 3.2. for more information about the SDRs encompassed by each of the selected frequency bands.

2.1.3. Slack Discussion Forum

This open-source initiative was initially meant to be a collaborative project. With this idea in mind, a Slack space was created so that users can keep the project alive themselves by asking/answering questions and sharing their thoughts. It is of interest to all of the

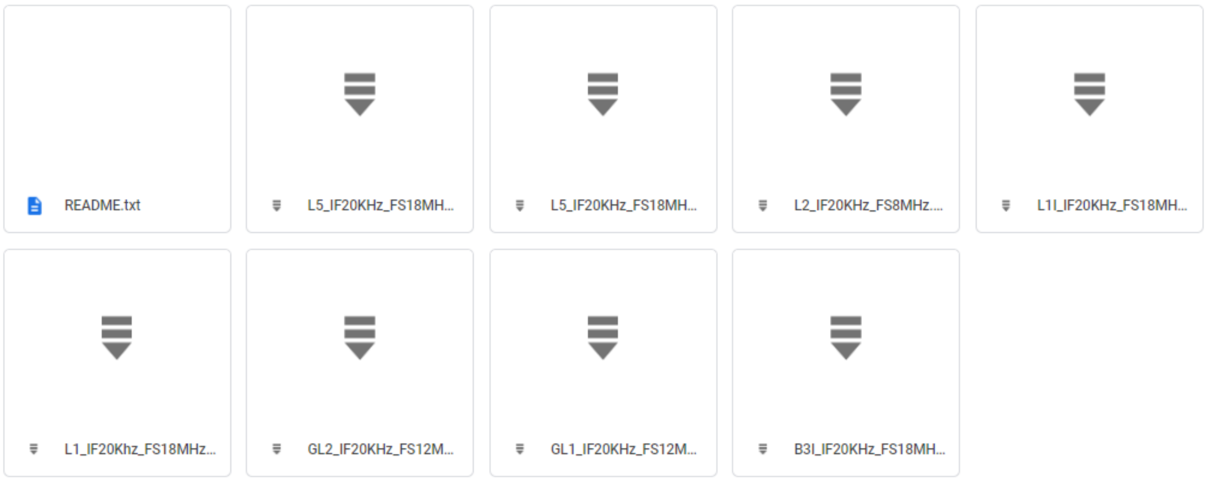
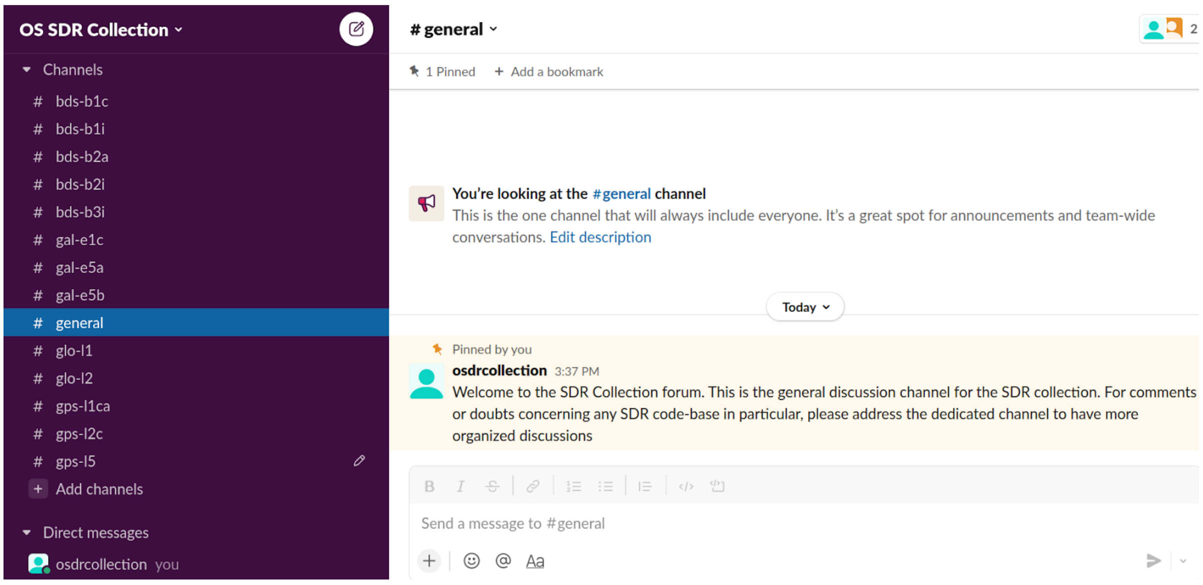


Figure 2.3: Screenshot showing the contents of the Google Drive with the sample datasets

individuals in the GNSS community to have a place like this which they can reach out to at no cost.

The front page of the discussion forum is shown in Figure 2.4. The space is organized in discussion channels, where there is one for every SDR plus a general channel. Specific channels are expected to hold discussions concerning maters of the SDR in question. The general channel is meant to provide a feed with less specific comments and more announcements affecting the SDR collection.

The only requirement for users to access the discussion forum is to have a Slack account, which is free to have. Once this requirement is met, users just need to join the space via the link provided in [5].



You're looking at the #general channel
This is the one channel that will always include everyone. It's a great spot for announcements and team-wide conversations. Edit description

Pinned by you
osdrcollection 3:37 PM
Welcome to the SDR Collection forum. This is the general discussion channel for the SDR collection. For comments or doubts concerning any SDR code-base in particular, please address the dedicated channel to have more organized discussions

Figure 2.4: Screen-capture showing the front page of the project's Slack discussion forum.

CHAPTER 3. TESTING FRAMEWORK

3.1. Experimental setup

This section is aimed at presenting the equipment used and the process followed to collect GNSS data, as well as the testing framework designed to assess the SDR collection. Sections 3.1.2. to 3.1.3. introduce the components involved in the GNSS data collection. These where arranged together as illustrated in Figure 3.1. Section 3.2. describes the data recording process. Finally, Section 3.3. talks about the testing framework designed and how SDRs were evaluated.

3.1.1. RF Antenna

The device used to sense RF GNSS signals was a *Trimble Zephyr 2* antenna, which counts with enhanced capabilities for multi-path minimisation as well as low-elevation satellite tracking properties. It was also rated for the environmental conditions where it was located, and operated under open-sky conditions during the entire data collection campaign. More characteristics can be found on *Trimble's Zephyr 2* data-sheet in [7].

The antenna was installed at the rooftop of the Ann and H.J. Smead Department of Aerospace Engineering Sciences building in University of Colorado Boulder. The location of the antenna was well known and acted as a reference location from which to obtain errors made in the positioning process.

3.1.2. USRP and TXCO devices

An Ettus' Universal Software Radio Peripheral (USRP) B200 SDR connected to an IQD's Temperature compensated crystal oscillator (TCXO) was used to collect digital samples from GNSS analogical signals sensed by the antenna.

The B200 device was controlled by means of the USRP Hardware Driver (UHD) through a Linux operating system. UHD is a software Application Programming Interface (API) that enables the development of code to manage USRP settings and operation. More details describing the characteristics of the USRP B200 can be found on the data-sheet available at [8].

The TXCO used was confined in a hermetically sealed 3.2 x 2.5mm SMD package. It was used as the reference oscillator for the signal sampling operation with improved stability properties, compared to the USRP's internal oscillator. This device is described in further detail in the data-sheet from [9].

3.1.3. PC set-up

The PC set-up consisted of a Linux computer counting with all the required Drivers and program dependencies, as well as with Mathworks' Matlab software installed. Matlab was used to program and automate the data recording process.

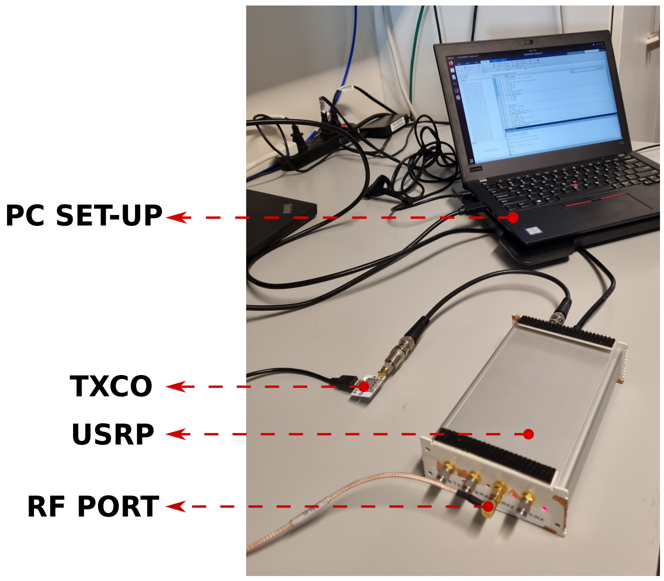


Figure 3.1: The antenna was connected to the RF port of the USRP. The USRP sampled the analogical data delivered by the antenna using the TXCO as the reference oscillator. The resulting sampled data was stored in a Linux-based computer.

3.2. Data recording

The equipment presented in previous sections was used to record data suitable for each SDR code-base. The process to obtain signal data for all 12 code-bases was reduced to eight stages by selecting an adequate frequency band-with for signals which shared the same central carrier frequency. All signals are summarised in Table 1.1 according to the frequency band they belong to and the band-width used to record them.

For each stage, a total of 100 files with 61 seconds of I/Q GNSS data were recorded over an time period of 24h. This was the data structure defined in the testing framework, described in Section 3.3.2.. The I/Q samples recorded by the USRP were formatted as 8 bit sine carriers. All the data-sets recorded are available together with a description file based on ION's metadata standard for GNSS [10].

	Freq. Band (MHz)	**Bandwidth Selected**	**Signals**		
1	1575.42	18	GPS L1CA	GAL E1C	BDS B1C
2	1176.45	18	GPS L5	GAL E5a	BDS B2a
3	1207.14	18	GAL E5b		
4	1561.098	18	BDS B1I		
6	1268.52	18	BDS B3I		
7	1602	12	GLO L1		
8	1246	12	GLO L2		
5	1227.60	8	GPS L2C		

Table 3.1: This table show the eight stages followed to record data for a total of 12 SDR code-bases.

3.3. Testing Framework

The testing framework was designed to assess SDRs to have a knowledge about their performance and act in consequence. The results obtained from this phase uncovered issues and errors affecting the performance and usability of each SDR implementation.

3.3.1. Testing metrics

The first step of the testing framework was to define the error metrics that would classify the major issues found in each SDR implementation. This would tell how well SDRs performed and where they needed to improve. Then, after this information was provided by the tests conducted, a statistic was defined to set a threshold telling whether SDRs required an assessment, based on the issues observed and how many times did they repeat.

The metrics used to classify most of the errors found were defined according to Table 3.2. Result type 1 reported when a code base experienced a code error that prevented it from operating, i.e. a crash in the execution. Type 2, told when the navigation solution not computed nor returned by the SDR code-base. Results type 3 and 4 were differentiated by a threshold concerning the positioning error returned by the SDR. This threshold was set to 30 meters and is further explained in the next subsection.

Despite several SDRs showed initially overall positioning results that met by far this error criteria, the goal was to define an error level which could be met by all SDRs as often as 90% of the files for which they were tested.

Type	**Meaning**
1	Code-base experienced a coding issue
2	No navigation solution obtained
3	Navigation solution with error $>$30 m
4	Navigation solution with error $<$30 m

Table 3.2: Table encompassing the error types defined for the tests and their meaning.

3.3.2. Testing approach

In order to compute statistics reflecting the performance of SDRs, all of them were run 100 times, providing one of the results from Table 3.2 at every execution. Thus, each test returned 100 results for an SDR. From these statistics were extracted to make evaluate the overall performance.

Error types frequently gave hints of what had to be adjusted in SDR code-bases. Type 1 indicated a coding error, even if it was not a repeating error, but just a boundary case. Type 2 showed the code-bases might need an assessment up to the Navigation Data Decoding stage. Type 3 errors were more ambiguous, as they indicated the error of the solution exceeded the defined threshold, which could be caused by numerous factors.

A metric was required to tell when code-bases provided results good enough to form part of the final collection. In this case, the metric selected was an error threshold. This threshold was set to a 30 meter Root-Mean-Squared ([3], Section 7.3.2) positioning error, which was computed from the position estimate delivered by the SDR and the well-known location of the Trimble's Zephire 2 antenna, described in Section 3.1.1., acting as the reference position. Hence, the error measured how far was the SDR position estimate with respect to the antenna's precise location. The 30 meter error value chosen for the threshold was defined so as to count with an error constraint which could be met by most SDRs regardless of their GNSS characteristics (satellites available, power levels) for at least 90% of the cases.

With the idea of putting together a SDR collection robust to code-errors guaranteeing a performance level, each code-base was considered valid only if it met the error metric for at least 90% of the tests. This would bound the errors user can expect in the position solution of their tests for most of the times.

CHAPTER 4. DEBUGGING PROCESS AND RESULTS

4.1. Debugging process

The ultimate goal of the debugging process was to assert the performance of every SDR implementation which did not achieve the desired performance during the testing face, to count in the end with a robust collection of SDRs. Missing pieces and errors were found for several code-bases, which had to be addressed following different approaches.

This work is covered in Sections 4.2.1. to 4.5.2.. Then, Section 4.6. makes a comment about a more general adjustment concerning the filtering of satellites that also improved the results provided by most SDRs.

This section updates and expands the contents of the initial paper [1] this thesis is based on, which presented the first version of the SDR collection. It describes the progress and the most recent achievements made, elaborating more on every case-study thanks to the less limiting format constrains of a thesis document.

4.1.1. Error criteria and Initial results

The first assessment of the initial SDR Collection showed rather disparate results, summarized in Table 4.1. The considerations extracted from this table are going to be pointed out next, organized by the error types observed:

- **Results type 1:** Galileo E5a and E5b as well as BeiDou B1C SDRs presented coding issues. These were specially harmful for E5b and B1C SDRs as they did not operate at all. This was a prime problem to solve before making these SDRs available so they did not block user's tests.
- **Results type 2:** Some SDRs presented serious problems when it came to computing a navigation solution. This was specially frequent in GPS L5, GLONASS L2 and BeiDou's signals. These results were not as surprising in the case of GPS L5 SDR, whose signal is not transmitted by many satellites within the GPS constellation. However, it was for the other SDRs.
- **Results type 3:** GPS L1C/A and L2C as well as Galileo E1C were the SDRs that seemed to struggle more in terms of positioning accuracy. As aforementioned, this issue could be due by numerous factors.
- **Results type 4:** Only Galileo E5a and GLONASS L1 met the performance criteria from 3.3.. The remaining code-bases required a performance assessment to increase the number of files for which the 3D RMS position error was below 30 meters.

The next sub-sections are going to give a more in-depth description of the issues found in all the SDRs which initially did not meet the performance defined, and the process followed to achieve better results when that was an option.

	Results type			
SDR	**1**	**2**	**3**	**4**
GPS L1C/A	0	4	28	68
GPS L2C	0	0	49	51
GPS L5	0	38	0	62
GAL E1C	0	0	12	88
GAL E5a	1	3	0	96
GAL E5b	100	0	0	0
GLO L1	0	0	4	96
GLO L2	0	13	0	87
BDS B1C	0	15	1	84
BDS B1I	100	0	0	0
BDS B3I	0	18	1	81
BDS B2a	0	12	1	87

Table 4.1: This table summarises the results obtained for each SDR code-base following the scheme denoted in table 3.2. GAL stands for Galileo, BDS for BeiDou and GLO for GLONASS.

4.2. GPS

The SDR collection includes GPS L1C/A, L2C and L5 SDRs. As initial tests in Table 4.1 showed, L2C and L5 SDRs did not provide a performance close to that of L1C/A. For L5 signal, these results could be a priori justified by a more acute lack of satellites operative transmitting this signal. L2C however, seemed to underperfom compared to the results obtained for other constellations with a similar number of satellites available.

That being said, L5 as well as L2C SDRs were both assessed with the aim to improve and validate both code-bases.

4.2.1. L2C

The main issue for GPS L2C was that it only provided an acceptable position solution accuracy (result type 4) in 51% of the cases, as illustrated in Table 4.3a. By taking a closer look at the code-base during a test for a single file, it was observed that a certain subset of satellites were acquired with a high acquisition metric, but failed the tracking stage, i.e. navigation data could not be extracted from their transmitted signals. This caused the SDR to count with less satellites available for the PVT computation stage, thus affecting negatively the availability and accuracy of the position solution. This issue was detected in other data-sets, which motivated a more involved assessment to the acquisition stage, to check the properties of the code-phase and carrier-frequency estimates that were delivered to the tracking stage.

4.2.1.1. Signal structure

GPS L2C signal is formed by two signal components, named CM and CL. The CM component is formed by a spreading code that modulates a navigation message. CL component is a pilot (data-less) signal modulated with a longer spreading code, allowing longer coherent integration times and thus better acquisition sensitivity. More information can be found in [11]. Appendix A.1.2. summarises the signal properties and SDR acquisition scheme of GPS L2C signal.

4.2.1.2. Final assessment

As depicted in A.1.2., GPS L2C signal is first acquired via the CM component, by performing a 20ms coherent correlation. The resulting estimates are refined afterwards using the CL pilot component accumulating non-coherently 75 correlations of 20 ms.

L2C CM signal's 20 ms coherent acquisition limits the minimum frequency resolution in the acquisition process to 25Hz, as explained in ([2], Appendix B.5). Table 4.2a shows a summary of settings selected for this SDR in the first place.

Parameter	**Value**
Frequency step	25 Hz
DLL Damping Ratio	0.7
DLL Noise Bandwidth	1.5 Hz
PLL Damping Ratio	0.7
PLL Noise Bandwidth	25 Hz

(a)

Parameter	**Value**
Frequency step	6.25 Hz
DLL Damping Ratio	0.7
DLL Noise Bandwidth	4 Hz
PLL Damping Ratio	0.7
PLL Noise Bandwidth	15 Hz

(b)

Table 4.2: GPS L2C SDR settings before (a) and after the performance assessment

With this configuration, the SDR was executed and the acquisition stage delivered what a-priori looked like promising acquisition results, shown in figure 4.1. From figure it is observed that most of the satellites acquired count with a high acquisition metric, which was introduced in Section 1.3.1.2.. Some of these satellites, such as PRN 10 (shown in the X-axis), are expected to provide sufficiently accurate acquisition estimates for the well operation of the trackig stage, because the low contribution of the noise in the correlation results allow a more sensitive correlation operation. This yields to more accurate code-phase and carrier-frequency estimates.

Taking into account this layout, the acquisition scheme was explored to find unexpected behaviours, taking a look at the 3D code-phase/carrier-frequency plot. This plot represents the correlation vector computed for every frequency bin of size 25Hz, and is illustrated in Figure 4.3a. From this figure, it can be noted that the correlation peak, which is expected to be located at the right carrier-frequency and code-phase, has a rather asymmetrical shape. It was suspected that the carrier-frequency estimate provided by this correlation result could be too far off from the truth, misleading the tracking stage and preventing it from performing as expected.

With the idea of improving the accuracy of this frequency estimate, a smaller frequency bin size was selected for the same test. This would allow for a better frequency resolution, at

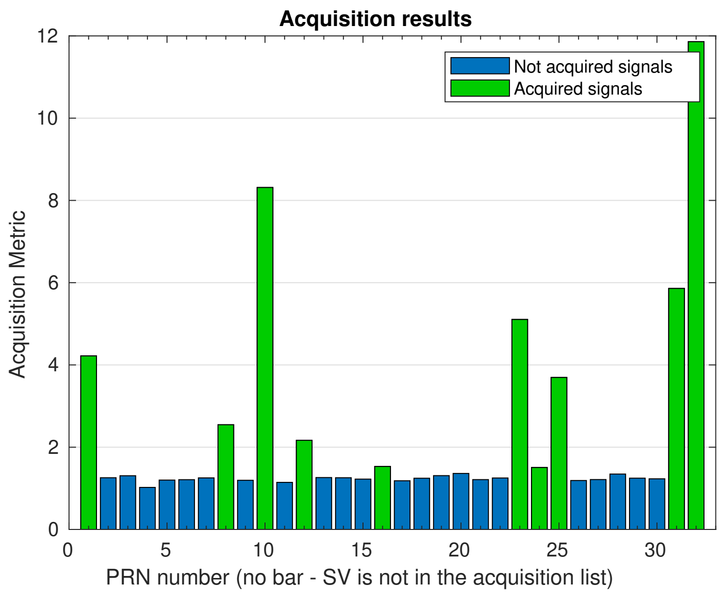


Figure 4.1: The acquisition plot for GPS L2C SDR showed a set of signals acquired at a high acquisition metric, such as PRN 10, which unexpectedly did not pass the tracking stage successfully, not providing the required navigation data bits containing ephemeris data.

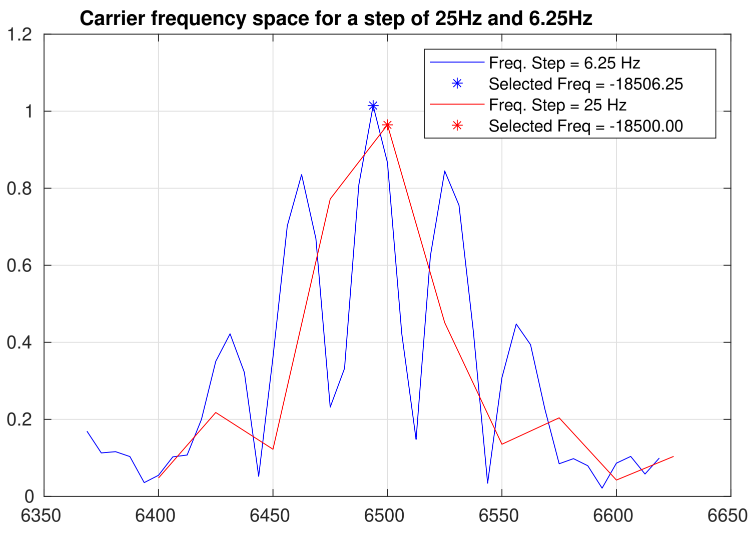
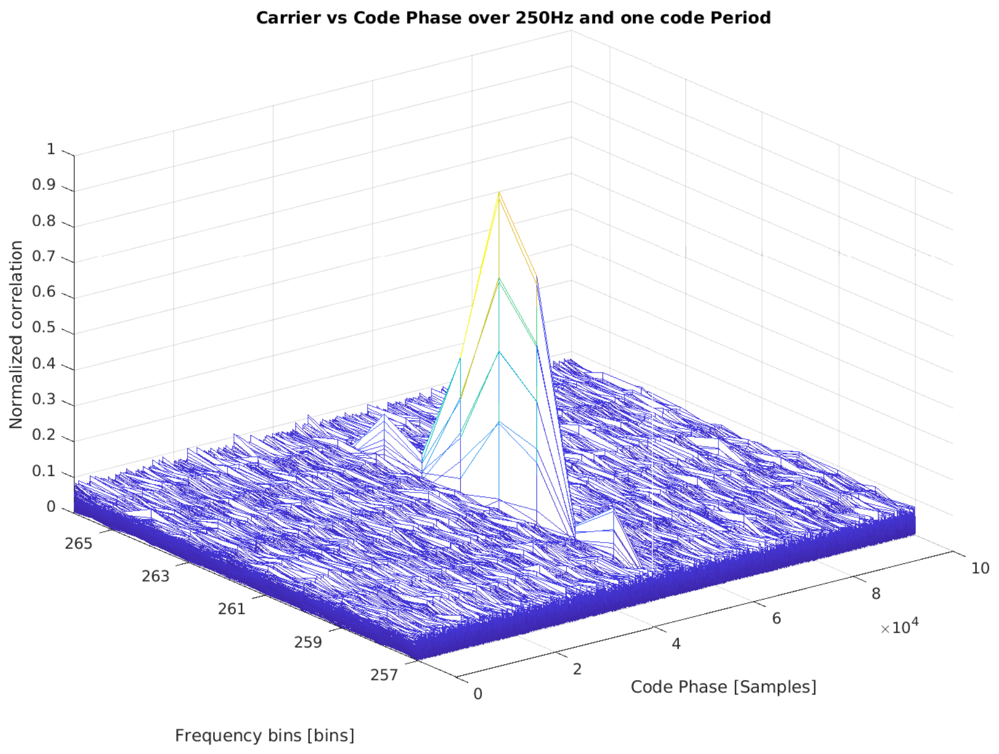


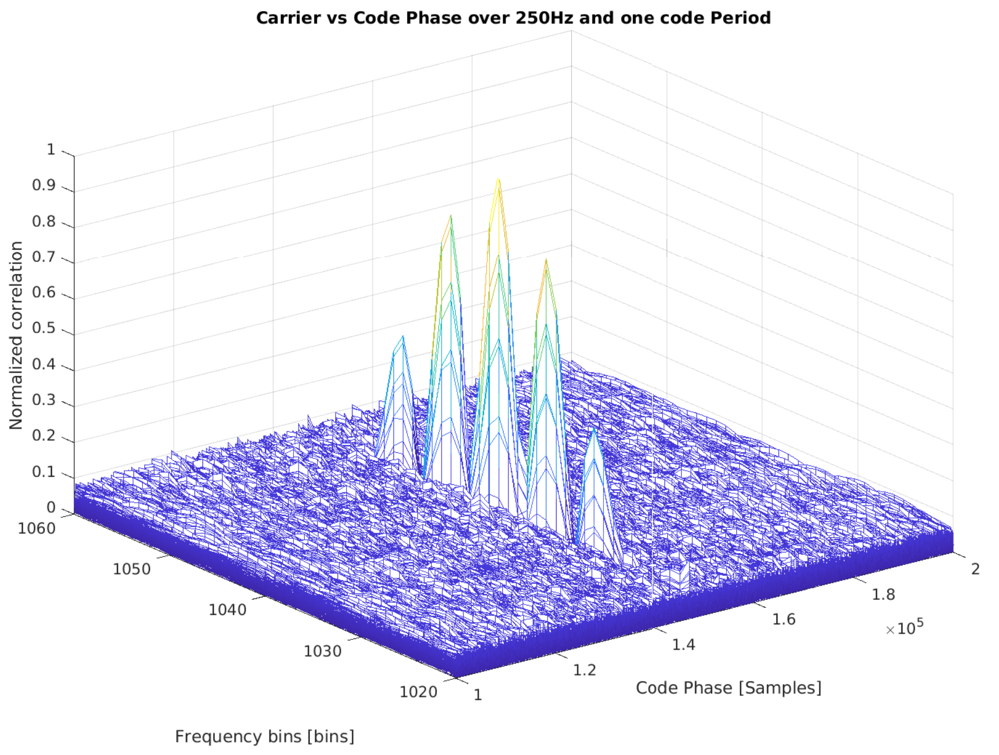
Figure 4.2: Figure comparing the correlation results in the frequency domain for both SDR settings in 4.2a. The configuration using a frequency step of 6.25Hz, points to a different carrier-frequency value, compared to the one using a 25Hz step.

the cost of increasing the computational burden of the correlation, increasing the number of frequency steps for which a correlation was computed. The result obtained for this case is shown in Figure 4.3b. It can be observed that the overall shape in the frequency domain is more symmetrical and centered around a frequency bin. The main lobe is also narrower, which suggests it is pointing to a the frequency bin with an improved accuracy.

The difference in the shape of the correlation plot in the frequency domain for both con-



(a)



(b)

Figure 4.3: This figure shows two correlation result for PRN 10 in both carrier-frequency and code-phase planes using a coarser and a finer frequency resolution. In Figure 4.3a, the selected frequency bin size is of 25Hz, whilst Figure 4.3b narrows this value down to 6.25Hz.

figurations in 4.2a is better reflected in Figure 4.2. Here it can be seen that the carrier-frequency estimates to which the main peaks in both configurations point are different. It is not a substantial difference, but is enough to mislead the tracking stage.

The frequency resolution selected for GPS L2C SDR constituted the most relevant parameter for a proper acquisition that later enabled the tracking of the signal. In addition to this, other minor settings where adjusted to refine the final results, which are summarized in Table 4.2b. As it can be seen in Figure 4.3b, with this configuration the SDR was able to compute an acceptable position solution for most of the files for which no navigation data could be extracted initially.

	Results type			
SDR	**1**	**2**	**3**	**4**
GPS L2C	0	0	49	51

(a)

	Results type			
SDR	**1**	**2**	**3**	**4**
GPS L2C	0	3	6	91

(b)

Table 4.3: GPS L2C SDR test results before 4.3a and after the debugging process 4.3b

4.2.2. L5

Figure 4.4 shows that GPS L5 SDR was performing poorly, as only for some data-sets an acceptable navigation solution was obtained. For 46% of the files, the navigation solution was either unacceptable (type 3) or was not able to be computed (type 2). This performance was mostly due to the lack of satellites in view transmitting GPS L5 signal at every duty cycle.

An example of the lack of satellite availability is demonstrated in Figure 4.5, where only a set of 4 satellites was acquired for a randomly selected data-set. Figure 4.4 supports this premise and was obtained using Trimble's web application in *https://www.gnssplanning.com*. It was configured with the same time-period and reference location as the data-set used for this example. It can be observed that the maximum number of satellites in view is of 7, and just for a short period of time, compared to the time periods where this value is inferior to that.

The absence of satellite redundancy has an impact in the SDR in several ways. It affects negatively the dilution of precision ([3], Section 7.3.1), i.e. DOP, and reduces the number of degrees of freedom from which to compute an accurate position. It also prevents the computation of the navigation solution when the number of satellites is inferior to 5, or one of these PRN is lost in any of the SDR stages. These issues are responsible for the performance obtained for GPS L5C SDR.

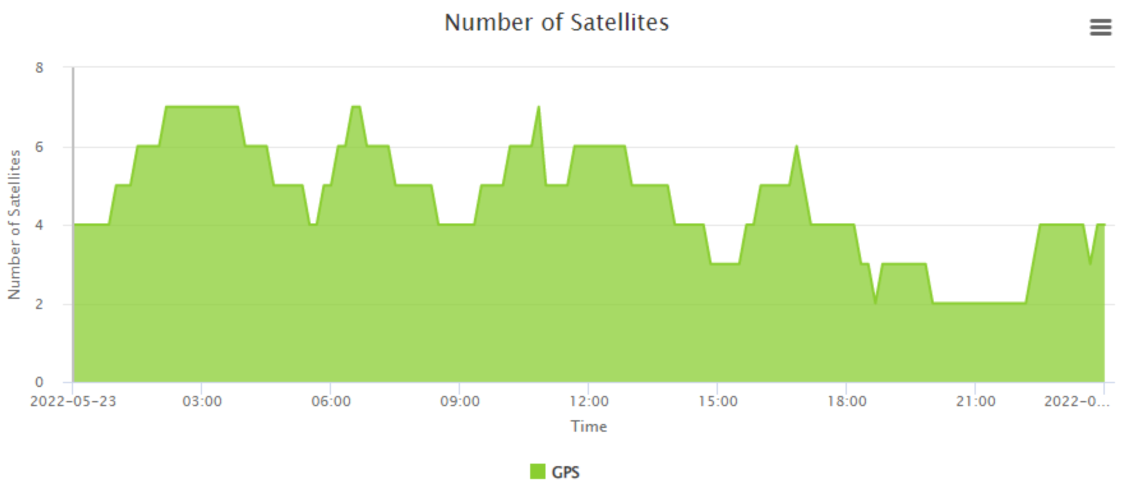


Figure 4.4: Figure extracted from https://www.gnssplanning.com showing the number of satellites transmitting GPS L5 signal for a 24h period.

	Results type			
SDR	**1**	**2**	**3**	**4**
GPS L5	0	14	21	64

Table 4.4: GPS L5C SDR test results. These results did not have much room for improvement compared to other SDRS that counted with more satellites available.

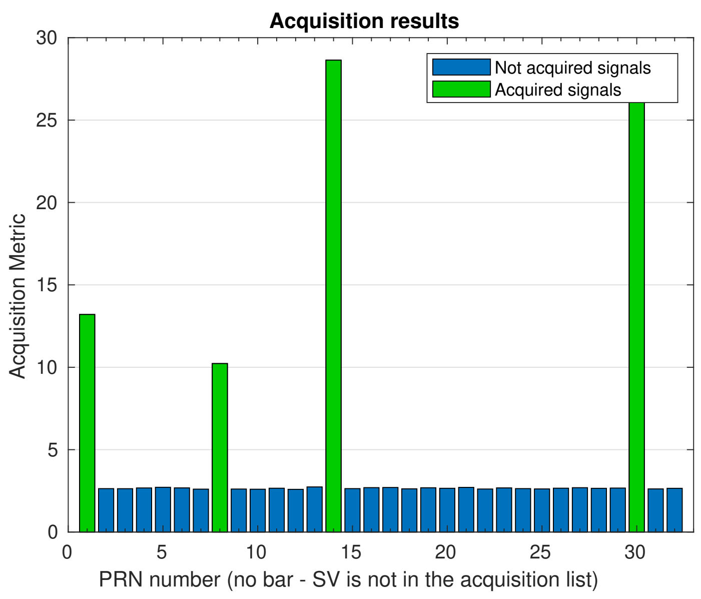


Figure 4.5: This figure was generated in the acquisition stage of GPS L5 SDR using an arbitrary data-set. It demonstrates that for certain time epochs the number of PRNs available compromise the performance of the SDR.

4.3. Galileo

For Galileo, the SDR collection implements E1C, E5a and E5b SDRs. These are the signals that are catalogued as open-service and are available for users. The following section is going to give an overview of the issues found with E5b SDR after the first tests for Galileo signals were carried out.

4.3.1. E5b

While Galileo E5a SDR performed meeting the criteria defined from the beginning, this was not the case for E5b signal. As Figure 4.5a points out, the SDR code-base crashed for 100% of the tests for which it was run. Soon after making a first assessment, it was observed that the source of the error was the navigation data decoding stage. The latter was designed to process GPS E5a signal's F/NAV, instead of E5b's I/NAV ephemeris. Hence, this part of the code needed to be rewritten and tested, following at all times Galileo's ICD [12] for reference. The rest of this sub-section is going to provide an overview of E5b signal's navigation data decoding stage as it was implemented in the SDR collection.

4.3.1.1. Signal Structure

As illustrated in A.6, Galileo E5b signal has two components. One is the component that modulates the navigation data (I) and the other is a pilot signal (Q). These two components are leveraged by the SDR following the scheme in Figure A.6c.

The tracking stage delivers a set of correlation prompts, obtained after performing a correlation operation between the input signal and the dynamically adjusted replica. For Galileo

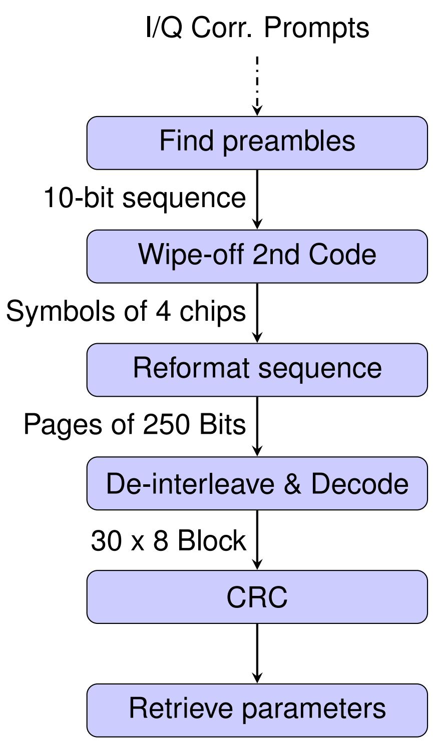


Figure 4.6: Flow diagram showing the scheme followed to decode I/NAV ephemeris from the correlation prompts delivered by the tracking stage.

E5b, this correlation is 1ms long, matching the duration of a primary code (A.6). A primary code constitutes a chip of the 4-chip long secondary code, which in result is 4ms long. Finally, every secondary code symbol of 4ms constitutes a bit from the 250 bps sequence transmitted in I/NAV. The combination of the primary and secondary codes represents the so-called *tiered-code*.

Taking into account this signal structure, the process to retrieve navigation parameters is represented in figure 4.6, and it starts by finding I/NAV preamble within the bit sequence represented by the correlation prompts obtained from the tracking stage. This is done by searching the location of the defined 10-bit preamble within the tiered-code sequence. It is important to note that these 10 bits translate in practice to 40 chips of 1ms, because of every bit is in fact a secondary code symbol of 4 chips. When preambles are found, the secondary code is then wiped-off and the bit sequence is arranged in sub-frames of 250 bits (240 excluding the 10 bit preamble). The sub-frame is then de-interleaved, taking into account the 30 rows by 8 columns scheme denoted in [12], Section 4.1.4.2. Once the bit sequence is arranged properly, it is ready to pass the viterbi decoding stage [13] and then a cyclic-redundancy-check (CRC), as described in [12], Section 5.1.9.4. The parameters from the final sequence are retrieved according to the sub-frame layout denoted in [12], Section 4.4.3.

Additional logic was added to discard cross-correlation products during the preamble search, to count with consecutive sub-frames containing the complete set of navigation parameters. When the navigation data decoding stage for E5b was implemented, the results observed after the tests met the defined performance in terms of position solution accuracy as shown in 4.5a. These results are close to those obtained for Galileo E5a SDR in 4.5b, as it was supposed to be due to their similarities.

	Results type			
SDR	**1**	**2**	**3**	**4**
GAL E5b	100	0	0	0

(a)

	Results type			
SDR	**1**	**2**	**3**	**4**
GAL E5b	0	3	0	97

(b)

Table 4.5: Galileo E5b SDR test results before 4.5a and after debugging 4.5b

4.4. GLONASS

The SDR collection implements both GLONASS L1 and L2 SDRs, which are almost identical. Both GLONASS signals are based on a OFDM media access, and differentiate in the selected frequency band of transmission. For L1 band, PRNs are allocated in the 1598.0625 MHz-1605.375 MHz frequency range. For L2, it goes from 1242.937 5MHz to 1248.625 MHz.

Both SDRs were expected to provide very similar if not identical results. The reason to this is that the processing scheme is the same, and the satellites in view at every epoch were expected to transmit both L1 and L2 signals simultaneously. However, the first assessment did not follow this premise. As Table 4.6 shows, the results in terms of position accuracy better in L1 than in L2. The latter did not meet the initial goal of ensuring most SDRs provided an acceptable position solution accuracy (result type 4) for at least 90% of the data-sets. The issue found for GLONASS L2 SDR is analysed in more detail in the following section.

	Results type			
SDR	**1**	**2**	**3**	**4**
GLO L1	0	0	4	96
GLO L2	0	0	13	87

Table 4.6: Both GLONASS L1 and L2 signals are expected to perform similarly because they count with identical characteristics. As L2 SDR provided worse results in terms of position accuracy it was assessed in more detail.

4.4.1. L2

After obtaining the results from Table 4.6, a dedicated assessment of GLONASS L2 SDR was required to find out the reason why it underpreformed compared to L1 SDR.

When comparing both SDRs with the same input data-sets, it was observed that GLONASS L1 was often able to acquire more satellites than GLONASS L2. This meant, some satellites were not transmitting L1 and L2 signals simultaneously. As no literature was found concerning this issue, initial suspicions were placed under L2 SDR acquisition stage. However, it was soon confirmed that some satellites were actually not transmitting L2 signal.

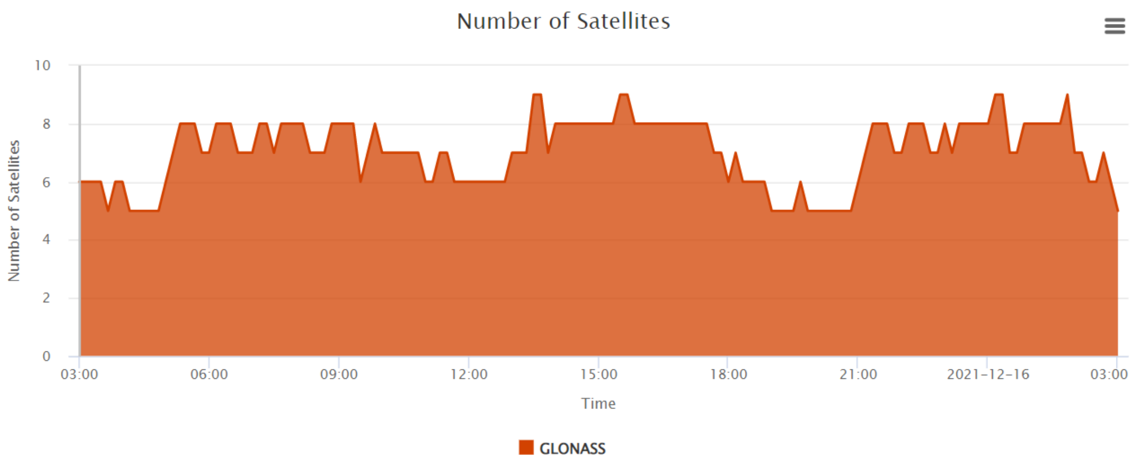
In order to keep track of these problematic satellites, the number of times every PRN was accquired for a 24h period was recorded for both L1 and L2. This showed that PRNs 6, 10 and 23 were never acquired by L2 SDR. A different GNSS receiver in the RF and

Satellite Navigation Lab showed the same results, proving it was not an issue with the SDR code-base.

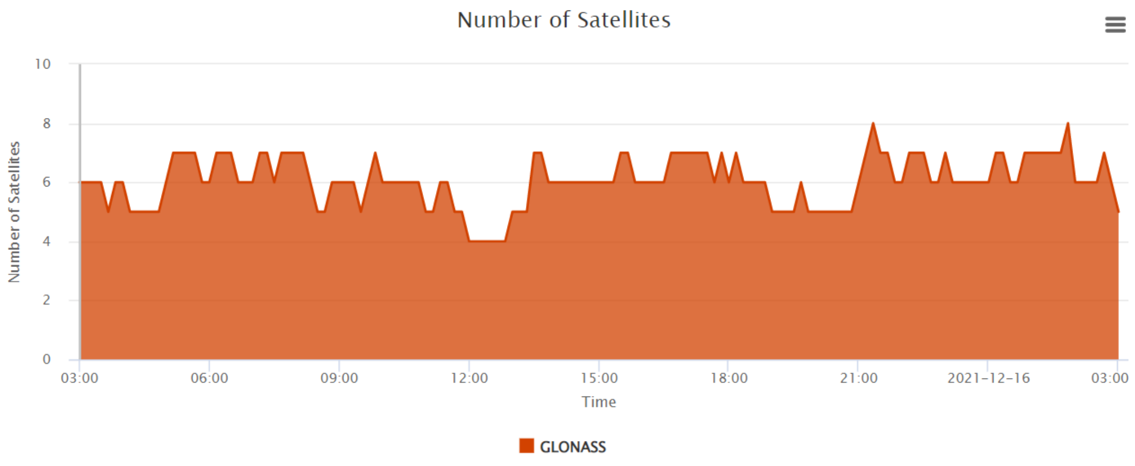
For this, the degraded performance of GLONASS L2 compared to L1 can be explained by the lower number of satellites transmitting L2 signal. For GLONASS L1, there are 22 satellites available, which is not a particularly high number compared to other GNSS signals that are above 32. Subtracting 3 PRNs from this list makes the GLONASS constellation even more vulnerable to satellite failures as it does not count with much redundancy to ensure a suitable geometry globally at all times.

The difference between GLONASS L1 and L2 in terms of satellite availability is illustrated in Figures 4.7a and 4.7b, respectively. It can be seen that L1 signal counts with more satellites in view and ready to use than GLONASS L2, for the same 24h time-period.

As Table 4.6 shows, most of the times, GLONASS L2 SDR was be able to compute a position solution, but the lower satellite redundancy compared to L1 degraded the accuracy of the solution delivered by the positioning algorithm.



(a)



(b)

Figure 4.7: Figure comparing GLONASS L1 and L2 satellites available for a 24h period. No information was found about the difference in the number of satellites observed, affecting negatively GLONASS L2 SDR performance.

	Results type			
SDR	**1**	**2**	**3**	**4**
GLO L2	0	13	0	87

(a)

	Results type			
SDR	**1**	**2**	**3**	**4**
GLO L2	0	0	13	87

(b)

Figure 4.8: GLONASS L2 SDR test results before 4.8a and after debugging 4.8b

4.5. BeiDou

BeiDou GNSS is supported in the SDR collection for B1I, B2a, B3I and B1C open-service signals. B2I signal processing was not included because it is considered deprecated with respect to the more recent B2a ([14], Section 2). The latter follows an identical scheme and differentiates only in the centre frequency of transmission ([14], [15]) so it can anyway be used to process B2I by the modifying the settings in the SDR implementation.

The first tests conducted for BeiDou's signals showed B2a and B1I SDRs were presenting issues concerning code-errors and navigation solution accuracy. The assessment performed in both cases is described in the following sections.

4.5.1. B1I

B1I SDR results showed the SDR crashed every time it was executed. A preliminary assessment showed the acquisition stage required fixes in the acquisition scheme as well as in the signal replica generation. The approach followed in both cases is described in more detail in the next sub-sections.

4.5.1.1. Acquisition scheme

The acquisition scheme for B1I did not follow the steps as specified in [16]. It needed to be adjusted so that carrier-phase and code-phase measurements could be computed and delivered to the tracking stage.

As shown in A.9, Medium-Earth-Orbit Satellites transmitting B1I D1 navigation message have a 1ms primary code, and a secondary code of 20ms. This two codes compose the tiered code and constitute a bit of the navigation message.

B1I has bit in the secondary code and the D1 navigation message that could affect the correlation process performed in the acquisition stage. Due to this, the baseline acquisition scheme is to correlate the input signal with a local replica composed by a 1ms signal block, followed by a 1ms block of zeros. This scheme is represented in Figure A.9c.

Rewriting the acquisition scheme so that B1I SDR followed the approach described in [16], Section 5.2, contributed drastically to remove the code-errors observed and obtain the results from Table 4.7b.

4.5.1.2. Signal Replica generation

Another issue found in this SDR was the way the local replica was being generated in the acquisition stage. The latter must be generated accordingly to ensure its samples are aligned with the input signal. If this is not the case, it will affect the correlation value obtained as well as the code-phase and carrier-frequency estimates delivered by the correlation.

The proper way to generate a local replica for a selected PRN is by concatenating blocks with the corresponding primary code, in this case, as shown in A.9, B1I SDR uses 2 consecutive blocks of 1ms in the acquisition. Initially, the code-base defined a 2ms 4ms long array of samples, where the first 2ms constituted B1I's primary code extended to 2ms. The signal replica was generated by filling the initialised local replica picking values of the primary code.

When the number of samples representing a primary code in a SDR is even, this methodology is not a problem. However, when it is odd, the number of samples assigned to each primary code block from the signal replica will differ, causing a misalignment in the correlation operation. This effect is specially unfavourable when the signal replica is shifted to the closest code-phase, but as samples are not properly aligned, the obtained correlation value is smaller in magnitude and thus more difficult to tell apart from other cross-correlation products.

This part was changed so that the signal replica was generated by defining a block of N samples for a period of the primary code, and then concatenating it with another block of N samples followed by two blocks of N samples set to zero. Fixing this issue also contributed to obtaining the results from Table 4.7b.

	Results type			
SDR	**1**	**2**	**3**	**4**
BDS B1I	100	0	0	0

(a)

	Results type			
SDR	**1**	**2**	**3**	**4**
BDS B1I	0	0	2	98

(b)

Table 4.7: BeiDou SDR test results before 4.7a and after debugging 4.7b

4.5.2. B2a

The first test results for BeiDou's B2a SDR (Table 4.8a) showed a 16% of code-related issues, and a total 6% of files from which a navigation solution was not able to be computed or its accuracy was below the defined error threshold.

On the one hand, the 16% of code-errors happened in the PVT stage, right before computing satellite positions, due to the absence of a correction factor. On the other hand, for %6 of the files, only a reduced sub-set of satellites were considered for the PVT computation stage, affecting the accuracy and the availability of the solution. Both issues are related to the navigation decoding stage and are adressed in the following sub-sections.

4.5.2.1. Error due to the absence of a correction factor

The Matlab prompt for files presenting a code-error pointed to the script were satellite positions are obtained, which is *satpos.m*. In particular, at the line the were satellite's ranging code phase offset is corrected using B2a's C-CNAV2 clock offset parameters.

In order to compute this correction factor, BeiDou's ICD [15], Section 7.6.2, provides the following formula:

$$(\Delta_{SV}) = \Delta t_{SV} - T_{GDB2ap} - ISC_{B2ad} \tag{4.1}$$

Where Δ_{SV} is the satellite ranging code phase offset, T_{GDB2ap} the Group delay differential of the B2a pilot component, and ISC_{B2ad} the Group delay differential between the B2a data and pilot components. All these parameters are obtained from B-CNAV2 ephemeris described in [15] Section 7.5 and 7.6.

Equation 4.5.2.1. needs to be applied to adjust satellite's ranging code phase offset according to B2a signal properties. In this case, B2a SDR implementation was reusing code from B1C code-base. This way, it was not retrieving the right correction factors as the ephemeris parameters from B1C's B-CNAV1 and B2a's B-CNAV2 are not delivered in the same format. Each ephemeris count with its own data structure. For this, when the SDR tried to access these ephemeris parameter it sometimes found there was not any data available in these fields.

Δ_{SV} correction factor is sufficiently small to not be easily detected if it is not applied correctly. However, every time the code-base tried to access these parameters and one of the fields was empty, it raised an error.

This issue was finally solved by adapting *satpos.m* script to B2a's B-CNAV2 ephemeris format following the B2a's ICD, and contributed to the improvement in the results shown in Table 4.8b.

4.5.2.2. Solving ephemeris decoding

The other problem found in B2a's SDR was that only a small sub-set of satellites made it to the PVT computation stage. It was observed that some satellites were rejected in the navigation decoding stage because they did not count with a full set of ephemeris data from which to obtain the required parameters in successive stages.

After a more detailed assessment, it was observed that some PRNs did not manage to retrieve certain pages from B-CNAV2 navigation frame. The latter is constituted by 8 sub-frames, 3 seconds long each ([15], Section 6.2.1 and Section 7.2). However, they are arranged in a particular way. Two of these sub-frames (10 and 11) are transmitted consecutively every 3 seconds. An additional frame, either 30, 31, 32, 33, 34 or 40 are alternated after frames 10 and 11, with a period of 6 seconds. This means, in order to fully count with a complete B-CNAV 2 frame, at least 54 seconds are required, plus additional time for transients to ensure all ephemeris frames are processed well.

B2a SDR initially did not account for the required processing time described above. Hence, some sub-frames were not present in the selected portion of GNSS data for a given satellite, making successive stages discard the satellite for not counting with a full set of ephemeris.

Once the amount of data to be processed was increased, this fixed the 6% of files that provided an inaccurate or no navigation solution as Table 4.8b shows.

	Results type			
SDR	**1**	**2**	**3**	**4**
BDS B2a	16	4	2	78

(a)

	Results type			
SDR	**1**	**2**	**3**	**4**
BDS B2a	0	0	0	100

(b)

Table 4.8: BeiDou B2a SDR test results before 4.8a and after debugging 4.8b

4.6. Satellite Filtering

This section introduces a less involved fix which was initially absent in most SDRs code-bases, but after which most SDRs improved considerably the results obtained, providing more accurate position solutions for a greater percentage of files processed. The latter concerns the convenient selection of satellites before the positioning algorithm to opt for a more accurate solution.

This satellite filtering is typically referred-to as satellite mask, and can be based on different metrics which tend to be interconnected amongst each other. In this case study, the filtering of satellites was based on both their elevation with respect to a reference position and their signal health flag.

Some sources ([3], 2.3.2.4) recommend choosing a 5° angle constrain, taking GPS GNSS as an example, below which satellites are filtered out. This filtering is done with the aim to receive satellite signals with better properties in terms of power ([3], 4.3.2) and multipath ([3], 6.3.2).

In addition to this, all GNSS ICDs tell satellite signals health flags should be checked before using the demodulated parameters in later stages of the SDR receiver. All signals include a parameter informing the user if the signal is ready to be used, i.e. ”healthy”. Some SDRs in the collection presented the required logic aimted at retrieving the health flag from the corresponding ephemeris. However, this was not always the case and in certain cases this logic had to be added.

After implementing both satellite filtering constrains in all SDRs, the results improved in most cases as Table 4.9b shows. This was specially notorious in signals such as GPS L1C/A, Galileo E1C and BeiDou B1C, whose results suffered almost exclusively from the absence of a satellite selection criteria in the positioning process. GPS L1C/A, Galileo E1C and BeiDou B1C went from 60%, 88% and 84% of type 4 results respectively, to 100%, 97% and 100% in the same order, after the satellite filtering feature was added.

CONCLUSIONS

Code-base validation results

The foremost premise of this document was to report the results and issues found in the process of collecting 12 SDRs developed by various individuals and assert their robustness and performance. Most of the workload of this project focused on assessing the SDRs presenting code and/or accuracy issues. Each assessment was conducted individually to ensure all implementations were robust and met the defined performance metrics.

In order to obtain statistics reflecting the performance of each SDR, the testing framework described in Chapter 3 was designed to layout the characteristics of the tests, and the metrics to evaluate the results obtained. The test phase began by collecting a total of 8 data-sets, one per frequency band, of 100 files with GNSS data covering a 24h period. The settings selected for each data-set were defined to fit the SDR they were aimed to. Each SDR was then run for 100 files, and the results returned were stored and classified.

The goal set for test results was defined in Section 3.3.2.. This was a 3D RMS positioning error under 30 meters for more than 90% of the times. When SDRs did not operate under this conditions, they were assessed to try to find out the cause and fix it. Chapter 4 showed most SDRs did not meet the defined error metrics at first. GPS L1C/A, Galileo E1C and BeiDou B1C SDRs achieved the defined performance after implementing the satellite filtering strategy from 4.6..This fix proved not enough for other SDRs, which required a more dedicated study in order to improve their results. These assessments where covered in Sections 4.2.1. to 4.5.2..

Tables 4.9a and 4.9b show the initial and final tests results of the SDR collection. Table 4.9b encompasses the results after the assessments from sections 4.2.1. to 4.5.2., taking also into account those SDRs that benefited from the satellite filtering of Section 4.6.. All SDRs but two improved their performance and achieved the error criteria defined. The exceptions were GPS L5 and GLONASS L2 SDRs. However, their respective assessments showed that their major limitation was due to a lack of satellites transmitting both signals.

	Results type			
SDR	**1**	**2**	**3**	**4**
GPS L1C/A	0	4	28	68
GPS L2C	0	0	49	51
GPS L5	0	38	0	62
GAL E1C	0	0	12	88
GAL E5a	1	3	0	96
GAL E5b	100	0	0	0
GLO L1	0	0	4	96
GLO L2	0	13	0	87
BDS B1C	0	15	1	84
BDS B1I	100	0	0	0
BDS B3I	0	18	1	81
BDS B2a	0	12	1	87

(a)

	Results type			
SDR	**1**	**2**	**3**	**4**
GPS L1C/A	0	0	0	100
GPS L2C	0	3	6	91
GPS L5	0	14	21	64
GAL E1C	0	0	3	97
GAL E5a	0	2	0	98
GAL E5b	0	3	0	97
GLO L1	0	0	4	96
GLO L2	0	13	0	87
BDS B1C	0	0	0	100
BDS B1I	0	2	0	98
BDS B3I	0	0	1	99
BDS B2a	0	0	0	100

(b)

Table 4.9: Table 4.9a shows the results of the tests performed to the initial version of the SDR Collection. Table 4.9b presents the results obtained for the same tests after implementing the fixes described in Chapter 4.

SDR Collection summary

As Chapter 1 revealed, the SDR Collection is in essence a directory containing 12 SDR code-bases arranged 4 in satellite constellations. Each code-base is formed by various scripts distributed in two sub-directories named ”include” and ”common”. The only scripts outside these sub-directories are ”initSettings.m” and ”init.m” as they set the configuration and run the SDR, respectively. The SDR Collection is was developed and designed to work in Mathwork’s Matlab platform. Thus, in order to perform tests users need an active MathWorks licence.

Chapter 2 showed this project made available a set of resources supporting the SDR Collection. These are a GitHub repository with all SDR collection code-bases, a Google Drive with Sample data-sets, and a Slack Discussion Forum for comments and suggestion. All of them can be accessed through the portal hosted by University of Colorado Boulder via the URL in [5].

BIBLIOGRAPHY

[1] Yafeng Li Dennis M. Akos Joan Bernabeu Frias, Fernando Palafox. A Collection of SDRs for Global Navigation Satellite Systems (GNSS). *Proceedings of the 2022 International Technical Meeting of The Institute of Navigation*, jan 2022. 3, 21

[2] Kai Borre and Dennis Akos. A software-defined GPS and galileo receiver: Single-frequency approach. In *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation, ION GNSS 2005*, volume 2005, 2005. 7, 8, 9, 23

[3] Michael J. Rycroft. Understanding GPS. Principles and applications. *Journal of Atmospheric and Solar-Terrestrial Physics*, 59(5), 1997. 9, 10, 20, 27, 35

[4] Sanjeev Gunawardena, Thomas Pany, and James Curran. ION GNSS software-defined radio metadata standard. *Navigation, Journal of the Institute of Navigation*, 68(1), 2021. 10

[5] Radio Frequency & Satellite Navigation Laboratory University of Colorado Boulder. Open-Source GNSS SDR Collection, 2021. 10, 13, 15, 38

[6] Scott Chacon and Ben Straub. Pro git, 2014. 13, 14

[7] Trimble Inc. *Trimble Zephyr 2 Antennas*, 2017. 17

[8] Ettus Research. *USRP B200/B210 Bus Series.* 17

[9] IQD frequency products. *Temperature compensated crystal oscillator (TCXO) in a hermetically sealed 3.2x2.5mm SMD package.*, 2021. 17

[10] ION GNSS SDR standard working group. *GNSS SDR Sampled Data Metadata Standard*, 2020. 18

[11] NAVSTAR. Navstar GPS space Segment/Navigation User Interface. Technical report, 2013. 23

[12] European Space Agency. European GNSS (Galileo) Open Service Signal In Space Interface Control Document. *European Union*, (1), 2010. 28, 29

[13] G. David Forney. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3), 1973. 29

[14] China Satellite Navigation Office. BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal. (Version 2.1), nov 2016. 32

[15] China Satellite Navigation Office. BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B2a. Technical Report Version 1.0, dec 2017. 32, 34

[16] China Satellite Navigation Office. BeiDou Navigation Satellite System Signal In Space Interface Control Document Open Service Signal B1I. Technical Report Version 3, jan 2019. 32

APPENDICES

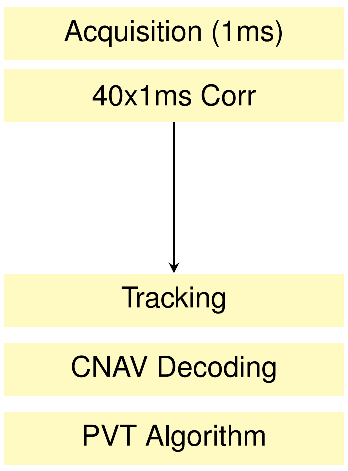
APPENDIX A. GNSS SIGNAL CHARACTERISTICS AND PROCESSING SCHEME

A.1. GPS

A.1.1. GPS L1CA

Signal component	C/A
Modulation	BPSK
freq band [MHz]	1575.42
Code rate [Mcps]	1.023
code length [chips]	1023
nav data [sps]	50
2nd code [bits]	N/A

(a)



(b)

Figure A.1: GPS L1C/A signal characteristics and signal processing flow diagram.

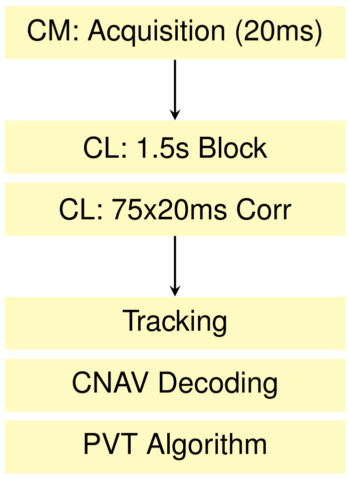
A.1.2. GPS L2C

Signal component	CM
Modulation	BPSK
freq band [MHz]	1227.60
code rate [kcps]	511.5
code length [chips]	10230
nav data [sps]	50
2nd code [bits]	N/A

(a)

Signal component	CL
Modulation	BPSK
freq band [MHz]	1227.60
code rate [kcps]	511.5
code length [chips]	767250
nav data [sps]	N/A
2nd code [bits]	N/A

(b)



(c)

Figure A.2: Flow diagram showing the scheme followed to decode I/NAV ephemeris from the correlation prompts delivered by the tracking stage.

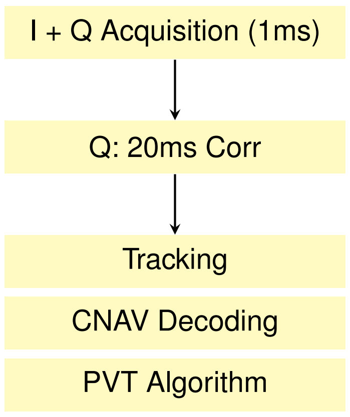
A.1.3. GPS L5

Signal component	I
Modulation	BPSK(10)
freq band [MHz]	1176.45
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	100
2nd code [bits]	10

(a)

Signal component	Q
Modulation	BPSK(10)
freq band [MHz]	1176.45
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	N/A
2nd code [bits]	20

(b)



(c)

Figure A.3: GPS L5 I and Q signal characteristics and signal processing scheme

A.2. Galileo

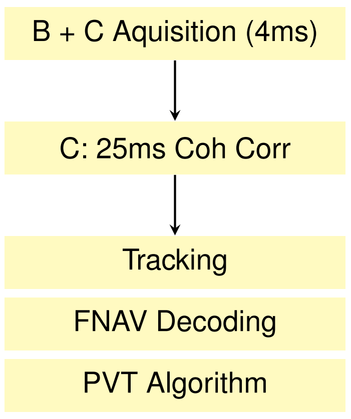
A.2.1. E1C

Signal component	B
Modulation	CBOC
freq band [MHz]	1575.42
Code rate [Mcps]	1.023
code length [chips]	4092
nav data [sps]	250
2nd code [bits]	N/A

(a)

Signal component	C
Modulation	CBOC
freq band [MHz]	1575.42
Code rate [Mcps]	1.023
code length [chips]	4092
nav data [sps]	N/A
2nd code [bits]	25

(b)



(c)

Figure A.4: Galileo E1C L5 B and C signal characteristics and signal processing scheme

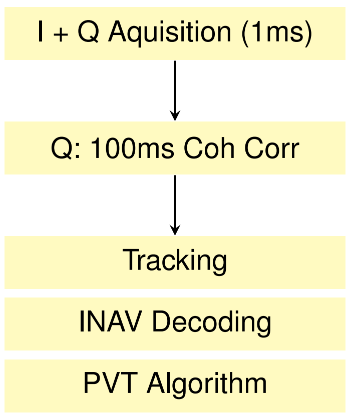
A.2.2. E5a

Signal component	I
Modulation	AltBOC
freq band [MHz]	1192.795
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	50
2nd code [bits]	20

(a)

Signal component	Q
Modulation	AltBOC
freq band [MHz]	1192.795
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	N/A
2nd code [bits]	100

(b)



(c)

Figure A.5: Galileo E5a I and Q signal characteristics and signal processing scheme.

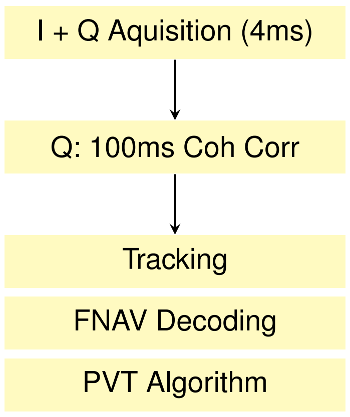
A.2.3. E5b

Signal component	I
Modulation	AltBOC
freq band [MHz]	1207.14
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	250
2nd code [bits]	4

(a)

Signal component	Q
Modulation	AltBOC
freq band [MHz]	1207.14
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	N/A
2nd code [bits]	100

(b)



(c)

Figure A.6: Galileo E5b I and Q signal characteristics and signal processing scheme

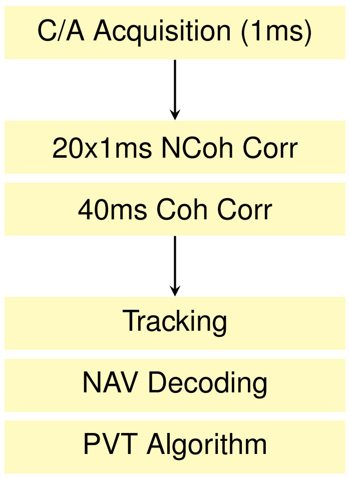
A.3. GLONASS

Signal	L1
Modulation	BPSK(0.511)
freq band [MHz]	1592.9525 - 1610.485
Code rate [Kcps]	511
code length [chips]	511
nav data [sps]	50
2nd code [bits]	N/A

(a)

Signal	L2
Modulation	BPSK(0.511)
freq band [MHz]	1242.9375 - 1248.625
Code rate [Kcps]	511
code length [chips]	511
nav data [sps]	50
2nd code [bits]	N/A

(b)



(c)

Figure A.7: GLONASS L1 and L2 signal characteristics and processing scheme

A.4. BeiDou

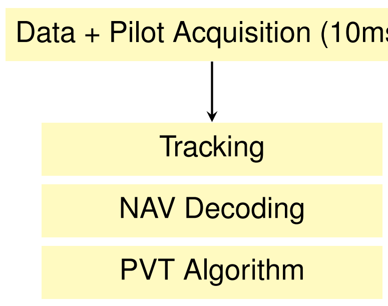
A.4.1. B1C

Signal component	Data
Modulation	BOC
freq band [MHz]	1575.42
Code rate [Mcps]	1.023
code length [chips]	10230
nav data [sps]	200
2nd code [bits]	N/A

(a)

Signal component	Pilot
Modulation	QMBOC
freq band [MHz]	1575.42
Code rate [Mcps]	1.023
code length [chips]	10230
nav data [sps]	N/A
2nd code [bits]	1800

(b)



(c)

Figure A.8: BeiDou B1C data and pilot signal characteristics and processing scheme

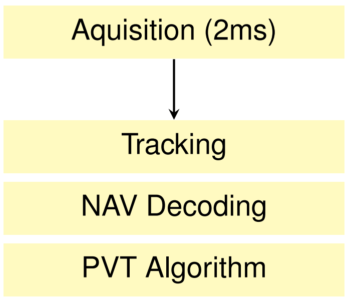
A.4.2. B1I - B2I

Signal	B1I
Modulation	BPSK
freq band [MHz]	1561.098
Code rate [Mcps]	2.046
code length [chips]	2046
nav data [sps]	50
2nd code [bits]	20

(a)

Signal	B2I
Modulation	BPSK
freq band [MHz]	1207.140
Code rate [Mcps]	2.046
code length [chips]	2046
nav data [sps]	50
2nd code [bits]	20

(b)



(c)

Figure A.9: BeiDou B1I signal characteritics and processing scheme. The latter is identical to B2I processing scheme.

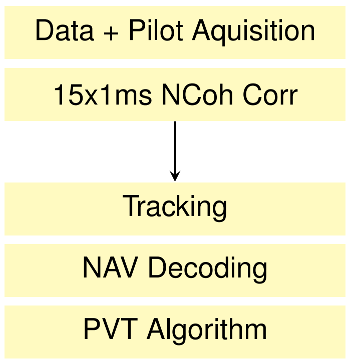
A.4.3. B2a

Signal	Data
Modulation	BPSK(10)
freq band [MHz]	1176.45
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	200
2nd code [bits]	5

(a)

Signal	Pilot
Modulation	BPSK(10)
freq band [MHz]	1176.45
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	N/A
2nd code [bits]	100

(b)



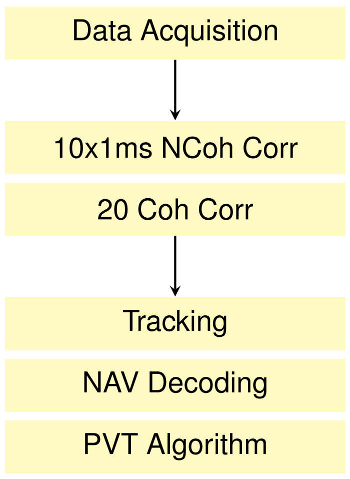
(c)

Figure A.10: BeiDou B2a signal characteristics and processing scheme

A.4.4. B3I

Signal	Data
Modulation	BPSK
freq band [MHz]	1268.53
Code rate [Mcps]	10.23
code length [chips]	10230
nav data [sps]	50
2nd code [bits]	20

(a)



(b)

Figure A.11: BeiDou B3I signal characteristics and processing scheme.