Centre Tecnològic de
Telecomunicacions de Catalunya

WORKSHOP Software defined receivers for satellite navigation

Student Guide

TELECO RENTA

PLAN DE
PROMOCIÓN DE LOS ESTUDIOS
DE TELECOMUNICACIÓN

Financiado por la Unión Europea NextGenerationEU

Plan de Recuperación, Transformación y Resiliencia

UNICO I+D

GOBIERNO DE ESPAÑA

This workshop has been prepared by the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), within the context the project "Plan de promoción de los estudios de Telecomunicación" (UNICO 5G I+D program).

Contact: telecorenta@cttc.cat

Authors: Carles Fernández, Mónica Navarro.

Document history

Revision	Date	Affected sections	Modifications
V1.0	10/10/2024	All	First edition

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

OUTLINE

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

List of Figures

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

Acronyms & abbreviations

DLL	Delay Locked Loop
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
SDR	Software Defined Radio
SW	Software
PLL	Phase Locked Loop
PSD	Power Spectral Density
RF	Radio Frequency
UDP	User Datagram Protocol
USRP	Universal Software Radio Peripheral

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

1. Workshop objectives

Welcome to an extraordinary journey into the fascinating world of Global Navigation Satellite Systems (GNSS). You may take it for granted that your smartphone can pinpoint your location with remarkable precision, but have you ever wondered about the science and technology behind this everyday miracle? The GNSS Workshop is your gateway to unravelling this captivating realm where the invisible dance of radio-frequency signals and a constellation of satellites shape our world.

GNSS, which encompasses systems like GPS, Galileo, GLONASS, and Beidou, is more than just a convenience; it's a technological marvel that continuously evolves and quietly revolutionizes our daily lives. From mapping applications to precision agriculture, emergency services to transportation, the impact of GNSS is immeasurable.

Intriguingly, the GPS receiver inside your smartphone is just the tip of the iceberg. Have you ever wondered how a tiny black box within your device can precisely locate you anywhere on Earth? The answer is a symphony of science and technology involving physics, mathematics, electronics, computer science, and digital signal processing. The GNSS receiver is the magician behind the curtain, weaving together signals from space into your position and time.

But what if we could pull back that curtain and demystify the magic? This workshop offers a unique opportunity to do just that. We'll delve into the world of software-defined GNSS receivers, where complex integrated circuits become lines of code. With open-source technology and Software-Defined Radio (SDR), you'll embark on a hands-on journey to understand, customize, and even build your own GNSS receiver.

The workshop kicks off with a fundamental introduction to GNSS technology, ensuring you grasp the pivotal role it plays in your daily life. From there, it's a captivating dive into the intricate workings of GNSS receivers, where you'll discover the art of computing your position and time with unparalleled precision.

But this isn't just about theory; it's a hands-on experience. You'll learn to set up your own software-defined GNSS receiver experiment. You will connect antennas, configure radio-frequency front-ends, and witness the magic unfold on your personal computer as you receive signals from actual satellites and determine your own position and time.

Summary of workshop Objectives:

- Intuitive Understanding: Gain an intuitive understanding of how Global Navigation Satellite Systems shape our world and underpin our daily experiences.
- Receiver Insights: Develop a comprehensive grasp of how GNSS receivers calculate your precise position and time, demystifying the technology within your devices.
- Hands-On Experience: Dive into the world of software-defined GNSS receivers, creating your experiments and configuring your own receiver to obtain position and time data directly from satellites.

Join us on this adventure, and let's explore the invisible threads connecting us to the skies above. GNSS technology is no longer a black box; it's a window into the art of navigation, and this workshop is your key to unlocking its secrets.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

2. Introduction

Have you ever wondered how your smartphone accurately pinpoints your location on a map? It's all thanks to an incredible technology called Global Navigation Satellite System (GNSS). Let's take a closer look at how it works.

Each GNSS consists of a network of satellites that orbit the Earth, transmitting signals that enable devices like smartphones and other electronic gadgets to determine their precise location anywhere on the planet. The most well-known GNSS is the Global Positioning System (GPS) developed by the United States, but there are others: GLONASS, developed by Russia; Galileo, developed by Europe; and BeiDou, developed by China. Each of these systems comprises approximately 32 satellites orbiting the Earth at an altitude of around 24,000 km from the planet's surface. These satellites complete two full orbits around the Earth in a single day.

Figure 1 - Representation of GNSS satellite constellations (GPS, GLONASS, Galileo, and BeiDou).

GNSS satellites are the modern version of lightphares. Technology has allowed us to place them in the sky instead of the seacoast, for better visibility, and to make them emit radio waves with ingenious waveforms instead of light. In the same way that mariners were able to estimate its 2D position over the sea by looking at two or more lightphares at the same time and the aid of a map, GNSS receivers can compute their 3D position and time by looking at four or more satellites at the same time. The problem that they solve (that is, *knowing where you are*) is in fact quite complex has kept mankind busy since very ancient times.

The art of finding the way from one place to another is called navigation. Until the 20th century, the term referred mainly to guiding ships across the seas. Indeed, the word "navigate" comes from the Latin *navis* (meaning "ship") and *agere* (meaning "to move or direct"). Today, the word also encompasses the guidance of travel on land, in the air, and in inner and outer space.

Navigation has been (and today still *is*) a major scientific and technological challenge. Navigation seems to start very early in time: according to Chinese storytelling, the compass was invented and used in wars during foggy weather before recorded history. Early mariners followed landmarks

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

visible onshore, until other technological breakthroughs came into play: the magnetic compass (c. 13th century), the astrolabe (c. 1484), the sextant (1757), the use of lighthouses and buoys, or the seagoing chronometer (1764). These inventions fueled disciplines such as surveying and geodesy, but also had a dramatic impact in transportation, and thus in economics. Today, the virtuous circle of science, technology, and business around navigation is still spinning and well alive.

The original proposal for a passive one-way ranging system, consisting of 24 satellites with atomic clocks in medium Earth orbits to achieve a 12-hour period, received approval from the U.S. Defense Department in December 1973. Named the "Navstar Global Positioning System," it was initially designed for military purposes. In February 1978, the first Block I developmental Navstar/GPS satellite was launched, followed by three more satellites by the end of that year. Between 1977 and 1979, over 700 tests were conducted with the assistance of Aerospace engineers, confirming the accuracy of the integrated space/control/user system.

In 1983, President Ronald Reagan authorized the use of Navstar (later known as GPS) by civilian commercial airlines to enhance navigation and safety in air travel. This marked the initial step toward authorized civilian usage, leading to the commercial availability of handheld GPS units by 1989. The Magellan NAV 1000, weighing 700 grams, offering limited battery life, and priced at $3,000, was among the first commercially available devices.

GPS technology continued to advance through the 1980s and 1990s, making its debut in cellphones in 1999. In 2000, the U.S. government approved plans to add three additional GPS signals for non-military use, and the "selective availability" program, limiting civilian GPS accuracy, was terminated. Concurrently, Russia, the European Union, and the Republic of China developed their own satellite-based navigation systems: GLONASS (operational since 1993), Galileo (offering Early Operational Capability from December 15, 2016), and BeiDou (operational since December 2011), respectively.

Receiver technology rapidly evolved, featuring smaller size, improved integration with other systems, increased precision, reduced power consumption, and lower costs. Today, GNSS receivers find pervasive use across various domains. These include transportation (land, sea, and air), surveying and mapping, crucial timing and synchronization applications in finance, telecommunications, and scientific research, as well as various scientific fields such as geophysics, meteorology, and seismology. Additionally, GNSS technology finds application in agriculture, military and defence operations, emergency and disaster management, mining and construction, environmental monitoring, and consumer devices like smartphones, cameras, and gaming platforms. From monitoring key infrastructures and tracking tectonic plate movements to enhancing user experiences in electronic games like the popular Pokémon GO, satellite-based navigation technology plays a pivotal role in a diverse array of our daily activities.

This Workshop provides an in-depth exploration of the fascinating realm of GNSS technology. Commercial receivers typically take the form of integrated circuits, commonly known as "chips," housed in small black boxes where the magic of navigation unfolds. While this packaging is convenient for integration into compact devices, it doesn't offer users insight into the internal workings. However, researchers at CTTC have introduced an innovative approach to GNSS receiver implementation: an open-source, software-defined receiver (see http://gnss-sdr.org). This revolutionary concept replaces the specialized integrated circuit with lines of code that can run on a general-purpose microprocessor, such as the one in your computer. This approach empowers users to craft their own GNSS receiver, experiment with tuning parameters, and gain a deeper understanding of the intricacies behind seemingly routine activities, such as checking Google Maps on your smartphone.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

2.1 Fundamentals of satellite-based navigation systems

Picture each satellite as a unique singer, performing a distinct song. Each song has the same duration and repeats continuously, like a musical loop.

Now, imagine you've memorized all these songs. By hearing a snippet of one, you can instantly calculate how much time has passed since it began. Multiplying this time by the speed at which the song travels from the satellite to your ears, you can determine the distance between the satellite and you. If you also know the precise location of that satellite, you can deduce that you're situated on the surface of a sphere, with the satellite at the centre and the distance as the radius. Exciting, right?

Extend this idea to listening to three different songs at once. You can then draw three spheres, each centred on a different satellite and with varying radii. Your exact position becomes the magical point where all three spheres intersect. Voila! You've cracked the code and found your location.

This analogy simplifies the core principle of Global Navigation Satellite Systems. While in reality, satellites don't sing but emit periodic electromagnetic waves, the fundamental idea remains as straightforward as this musical analogy.

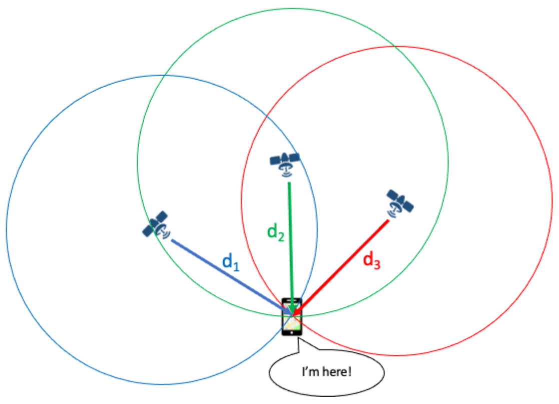


Figure 2 - Oversimplified diagram of how satellite-based navigation systems work

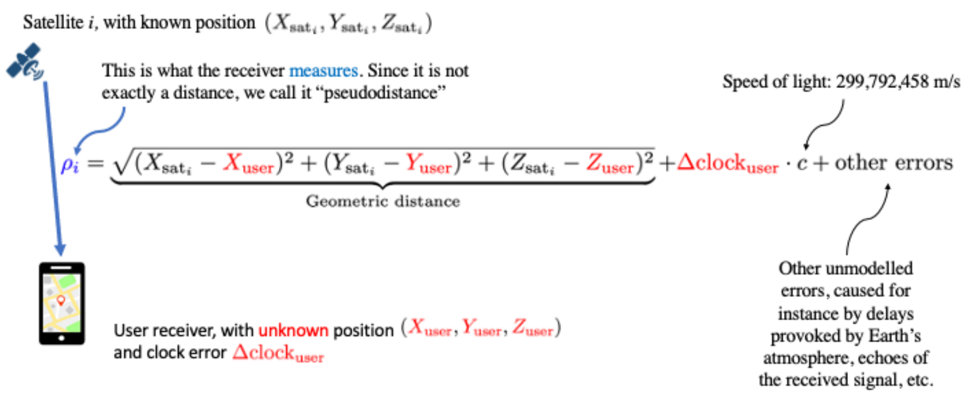
Diving into the intricacies, the analogy mentioned earlier simplifies the process of computing our 3D position based on the distances to three known points. We've explored how these distances can be calculated as delays multiplied by the propagation speed of electromagnetic waves. However, to accurately determine the delay caused by the propagation time from the satellite's antenna to your receiver's antenna (your 'ears' in the analogy), we encounter a crucial requirement – the need for a clock ticking in perfect harmony with the internal clock of the satellite.

Any deviation from synchronization would introduce errors in the delay estimation. Here, precision matters significantly. Given that electromagnetic waves travel at the speed of light, a mere 1-millisecond error can result in an approximate 300-kilometer positional discrepancy! To mitigate this, all GNSS satellites are equipped with extremely precise atomic clocks. These clocks, however, come at a considerable cost.

Yet, the challenge doesn't end there. For an accurate position fix, your receiver would also require an atomic clock to ensure precise timekeeping. Unfortunately, these clocks are exorbitantly expensive, far surpassing the cost of your everyday smartphone. Consequently, relying on atomic clocks in consumer-grade devices is not a practical solution.

Financiado por la Unión Europea NextGenerationEU Plan de Recuperación, Transformación y Resiliencia UNICO I+D GOBIERNO DE ESPAÑA

To understand how we can overcome this hurdle, let's attempt to encapsulate what we've learned in an equation. Consider that we are receiving the signal transmitted by satellite *i*, positioned at ($Xsat_i$, $Ysat_i$, $Zsat_i$), while our receiver is at an unknown location (X_{user}, Y_{user}, Z_{user}). Additionally, the internal clock of our receiver deviates by an unknown amount, $\Delta clock_{user}$ seconds, from the atomic clock of the satellite. We can express this relationship in the following equation:



Satellite i, with known position $(X_{\text{sat}_i}, Y_{\text{sat}_i}, Z_{\text{sat}_i})$
This is what the receiver measures. Since it is not exactly a distance, we call it "pseudodistance"
Speed of light: 299,792,458 m/s
$\rho_i = \underbrace{\sqrt{(X_{\text{sat}_i} - X_{\text{user}})^2 + (Y_{\text{sat}_i} - Y_{\text{user}})^2 + (Z_{\text{sat}_i} - Z_{\text{user}})^2}}_{\text{Geometric distance}} + \Delta\text{clock}_{\text{user}} \cdot c + \text{other errors}$
Other unmodelled errors, caused for instance by delays provoked by Earth's atmosphere, echoes of the received signal, etc.
User receiver, with unknown position $(X_{\text{user}}, Y_{\text{user}}, Z_{\text{user}})$ and clock error $\Delta\text{clock}_{\text{user}}$

Figure 3 - Mathematical model for receiver's measurement

This equation reveals that each delay measured from a satellite (called "pseudodistance") introduces four unknown quantities. Notably, these unknowns are consistent across all satellites, as they are only dependent on the location and clock characteristics of the receiver. As you may anticipate, resolving these four unknowns requires incorporating measurements from at least a fourth satellite. Consequently, we can formulate a system of (at least) four equations, each corresponding to a different satellite, collectively constituting a set of simultaneous equations:

$$
\begin{aligned}
\rho_1 &= \underbrace{\sqrt{(X_{\text{sat}_1} - X_{\text{user}})^2 + (Y_{\text{sat}_1} - Y_{\text{user}})^2 + (Z_{\text{sat}_1} - Z_{\text{user}})^2}}_{d_1} + \Delta\text{clock}_{\text{user}} \cdot c + \text{other errors}_1 \\
\rho_2 &= \underbrace{\sqrt{(X_{\text{sat}_2} - X_{\text{user}})^2 + (Y_{\text{sat}_2} - Y_{\text{user}})^2 + (Z_{\text{sat}_2} - Z_{\text{user}})^2}}_{d_2} + \Delta\text{clock}_{\text{user}} \cdot c + \text{other errors}_2 \\
\rho_3 &= \underbrace{\sqrt{(X_{\text{sat}_3} - X_{\text{user}})^2 + (Y_{\text{sat}_3} - Y_{\text{user}})^2 + (Z_{\text{sat}_3} - Z_{\text{user}})^2}}_{d_3} + \Delta\text{clock}_{\text{user}} \cdot c + \text{other errors}_3 \\
\rho_4 &= \underbrace{\sqrt{(X_{\text{sat}_4} - X_{\text{user}})^2 + (Y_{\text{sat}_4} - Y_{\text{user}})^2 + (Z_{\text{sat}_4} - Z_{\text{user}})^2}}_{d_4} + \Delta\text{clock}_{\text{user}} \cdot c + \text{other errors}_4 \\
\vdots &= \qquad\qquad \vdots
\end{aligned}
$$

Figure 4 - System of equations for the determination of user's position and time.

In this system, each equation represents the delay measured from a specific satellite. Solving this set of equations simultaneously allows us to determine the precise 3D coordinates of the receiver, and the exact time in which those signals have been received. With atomic clock precision!

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

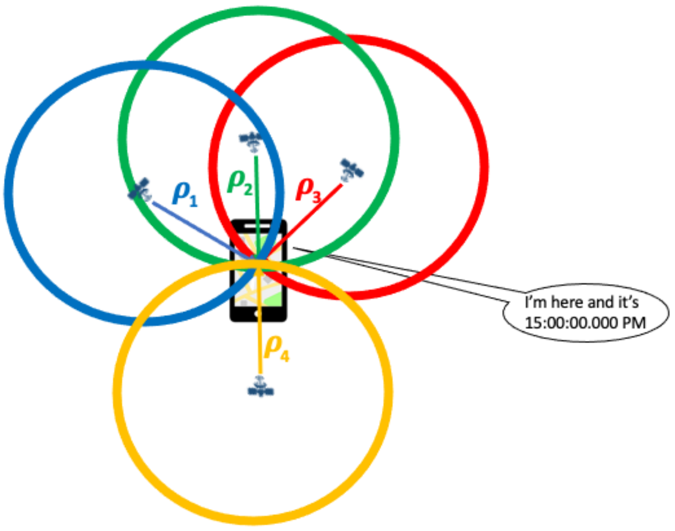


Figure 5 - An improved diagram of how satellite-based navigation systems work.

At this juncture, you might be wondering how the receiver can discern the exact position of each satellite at any given time. Drawing a parallel with our musical analogy, satellites ingeniously alter the lyrics of their 'songs' while steadfastly adhering to the tempo. This alteration involves the transmission of crucial orbital parameters that allow the receiver to compute the satellites' precise positions at any given moment. Amazing, isn't it?

In technical terms, this transmission of orbital parameters is referred to as the *navigation message*. Essentially, it constitutes a set of critical parameters that convey information about the satellite's orbital trajectory, health status, and other pertinent details. It serves as the informational cornerstone, allowing the receiver to calculate the positions of the satellites in real-time and ensuring the accuracy and reliability of the entire Global Navigation Satellite System (GNSS).

In essence, the system of equations above encapsulates the fundamental principle of GNSS navigation. As technology advances, researchers and engineers around the world are continually refining methods for solving it in more accurate and efficient ways. This involves the constant improvement of statistical models for the "other errors" components, such as troposphere and ionosphere characterization, accounting for the potential presence of signal echoes, and the mitigation of electromagnetic interferences, among other factors.

The evolution of GNSS technology relies on the continuous enhancement of these statistical models and digital signal processing methods. Additionally, it involves the creation of innovative technologies that facilitate the practical implementation of these methods. The pursuit of more accurate, reliable, and robust GNSS solutions drives collaboration among experts in various fields, ensuring that the technology not only keeps pace with the demands of today but also anticipates the challenges of tomorrow.

Crucially, this collaborative effort spans beyond mathematicians, physicists, or telecommunication / electrical / computer engineers. It extends to the entire community of users who seek to harness this technological marvel for leisure, business, or exploratory endeavours. From everyday commuters relying on navigation apps to global enterprises optimizing logistics, GNSS has become an indispensable tool, and the ongoing collaboration between developers and users ensures its adaptability and effectiveness in meeting the diverse needs of a global user base.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

2.2 How the signals transmitted by GNSS satellites are?

Up to this point, we've grasped the significance of receiving signals from a minimum of four GNSS satellites simultaneously to compute our position and time accurately. Let's delve deeper into how these signals are characterized.

Two primary approaches are employed for signal description: the time domain and the frequency domain. Although mathematically equivalent, they offer distinct perspectives on the features of the signal.

In the time domain, signals are described through their waveform, a representation moulded by a process known as *modulation*. This transformative process turns a binary stream of 0s and 1s into voltage variations. For instance, positive voltage may be assigned to a bit 1, and negative voltage to a bit 0. In GNSS signals, this includes the combination of their uniquely distinctive code (their "song" in our musical analogy) and the navigation message containing info about their orbit and status (the "lyrics"), so the receiver can compute its exact location. Those two streams of 0s and 1s, each one changing their values at a different rate, are the multiplied by a sinusoid, which acts as the carrier frequency that allocates the signal's energy in the desired frequency band. The resulting product is then transmitted by the satellite's antenna. This process is sketched in Figure 6.

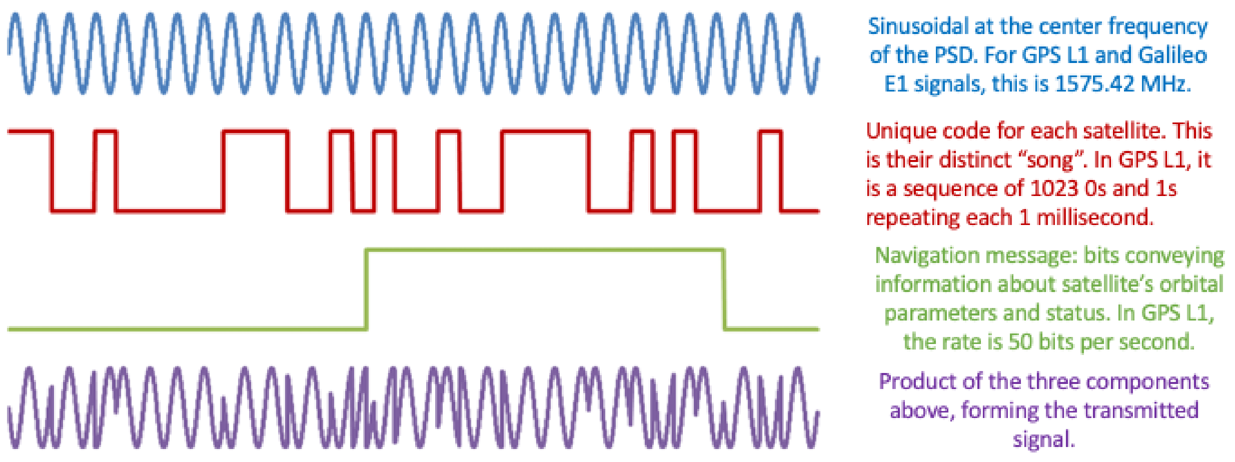

Sinusoidal at the center frequency of the PSD. For GPS L1 and Galileo E1 signals, this is 1575.42 MHz.
Unique code for each satellite. This is their distinct "song". In GPS L1, it is a sequence of 1023 0s and 1s repeating each 1 millisecond.
Navigation message: bits conveying information about satellite's orbital parameters and status. In GPS L1, the rate is 50 bits per second.
Product of the three components above, forming the transmitted signal.

Figure 6 - Signal transmitted by a GNSS satellite expressed in the time domain. This representation is not to scale, it's only for illustration purposes.

Conversely, in the frequency domain, signals find description through their power spectral density (PSD), a function outlining how power is distributed across frequency components within the signal. is commonly expressed in watts per hertz (W/Hz), or its logarithmic scale dBW/Hz (1 watt = 0 dBW, 1 milliwatt = +30 dBW, 1 microwatt = -60 dBW, 1 nanowatt = -90 dBW, and so on).

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

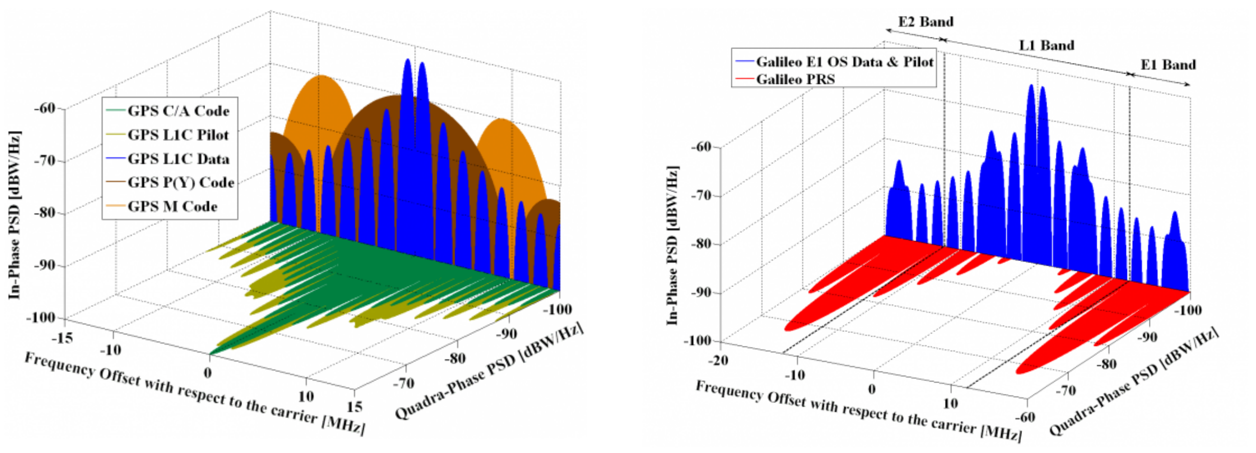


Figure 7 - Signals transmitted by a GNSS satellite expressed in the frequency domain. Various modulations exhibit distinct Power Spectral Density (PSD) functions. Specifically, the figure illustrates the PSD for GPS L1 signals (on the left) and Galileo E1 signals (on the right). Source: Navipedia.

Table 1 presents a comprehensive list of current and planned GNSS signals, each system delivering signals across distinct frequency bands, utilizing diverse modulations, and featuring unique navigation message structures. While most commercial receivers predominantly utilize GPS L1 C/A signals (with some modern devices incorporating Galileo E1b/c), it's noteworthy that only high-end devices, such as those used in surveying, make use of signals from more than one frequency.

GNSS Signal	Center Frequency (MHz)	Modulation type
GPS L5	1176.45	BPSK(10)
Galileo E5a	1176.45	QPSK(10)
BeiDou B2a	1176.45	BPSK(10)
GLONASS L3OC	1202.025	BPSK(10)
Galileo E5b	1207.14	QPSK(10)
BeiDou B2I	1207.14	BPSK(2)
GPS L2C	1227.60	BPSK(1)
GLONASS L2OF	1246.00	BPSK(0.5)
GLONASS L2OC	1248.06	BOC(1,1)
BeiDou B3I	1268.52	BPSK(10)
Galileo E6B	1278.75	BPSK(5)
BeiDou B1I	1561.098	BPSK(2)

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

BeiDou B1C	1575.42	BOC(1,1)
GPS L1 C/A	1575.42	BPSK(1)
GPS L1C	1575.42	BOC(1,1)
Galileo E1b/c	1575.42	CBOC(6,1,1/11)
GLONASS L1OC	1600.995	BOC(1,1)
GLONASS L1OF	1602.00	BPSK(0.5)

Table 1 - Current and planned GNSS signals providing Open Service.

Curious for more in-depth information? Explore the details at https://gnss-sdr.org/docs/tutorials/gnss-signals/. This resource provides a comprehensive description of all available Open Service signals, along with links to the official documentation for further exploration.

2.3 Synchronization of GNSS signals

As already suggested by our musical analogy, it's all about timing: we need to know when the song started. Essentially, this involves determining the locations of the rising edges in the signals illustrated in Figure 6. By doing so, we can sample the signal appropriately, transitioning back to the digital domain, and extract the required information from the received stream of 0s and 1s.

This process of measuring and tracking the evolution of the received signal's timing is termed *synchronization*. It encompasses the estimation of three crucial parameters:

1. **The received frequency**: Although GNSS signals are transmitted by satellites at known, fixed frequencies (for instance, GPS L1 and Galileo E1b/c are transmitted at 1575.42 MHz), those signals are not received at that exact frequency by the receivers on the ground. This discrepancy is due to the Doppler effect—similar to how an ambulance siren sounds at a higher pitch when approaching than when moving away from you. The received frequency (f_{received}) is influenced by the transmitted frequency ($f_{\text{transmitted}}$) and the relative velocity between the satellite and the receiver's antenna:

$$f_{\text{received}} = \frac{c + v_{\text{receiver}}}{c + v_{\text{transmitter}}} f_{\text{transmitted}}$$

where v_{receiver} is the receiver's velocity, $v_{\text{satellite}}$ is the satellite's velocity, and c=299,792,458 m/s is the speed of light.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

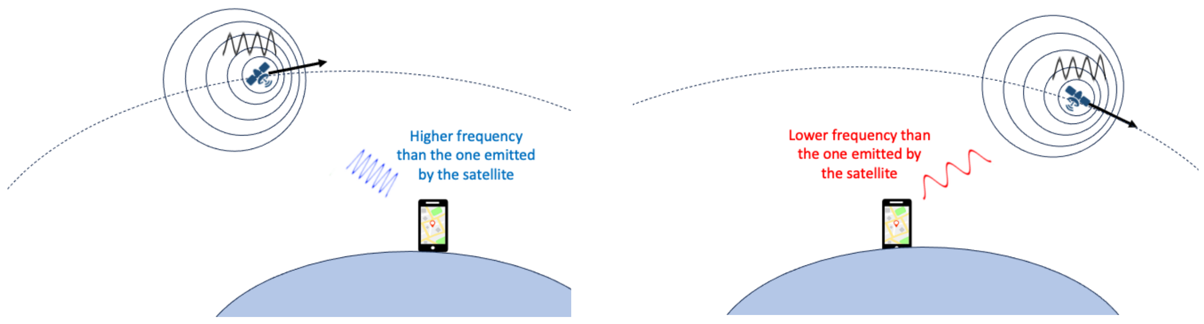

Higher frequency than the one emitted by the satellite
Lower frequency than the one emitted by the satellite

Figure 8 - When a satellite is approaching the receiver, it "sees" a higher frequency than the one originally emitted by the satellite. On the contrary, when the satellite is going away from the receiver, the received frequency is lower. This is called the Doppler effect.

2. **Time delay:** At this point, measuring the propagation delay is not feasible because we don't know how many times the unique code identifying the satellite has been repeated since transmission. However, we can determine when the currently received code started, as we possess knowledge of the full sequence in advance. This time delay is commonly referred to as *code phase*.

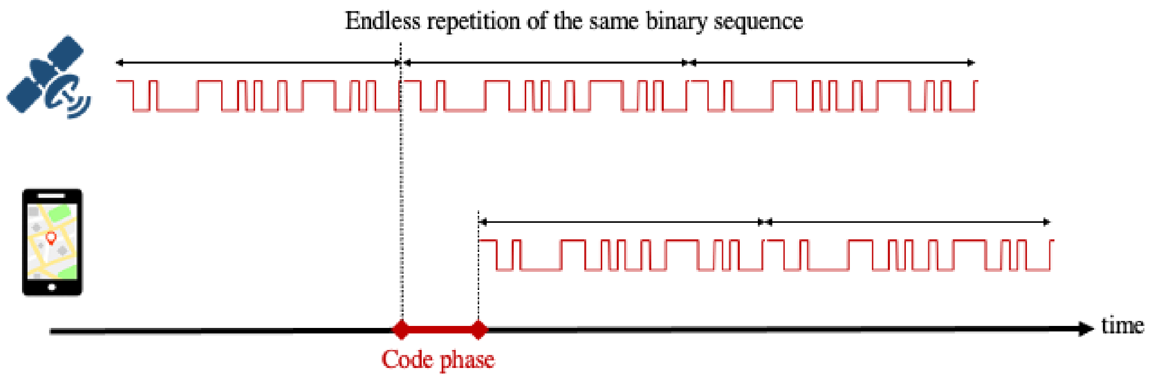


Figure 9 - Concept of code phase.

3. **Carrier phase**: Measurement of the difference between the received signal's carrier frequency and the frequency of the internal oscillator of the receiver.

Determining these three parameters for each of the received signals is the primary task of a GNSS receiver. Once these quantities are accurately estimated and tracked, the receiver can *demodulate* the navigation message, which allows to compute the satellite's position. This step provides all the necessary components to solve the system of equations presented in Figure 4, enabling the determination of the user's receiver position, velocity, and time.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

2.4 How a GNSS receiver looks like from the inside?

If you disassemble a device equipped with a GNSS receiver, you may identify the chip responsible for its implementation. Typically, it appears as a tiny black box with metallic pads. However, understanding its internal workings or modifying its behaviour is challenging. This scenario changes with a *software-defined* receiver, where the traditional black box is replaced by lines of code. In this setup, you gain visibility into the entire processing chain – from the reception of a stream of raw 0s and 1s to the computation of the user's receiver position, velocity, and time.

The description of the structure and functional blocks of a GNSS receiver provided in this section corresponds to the software architecture implemented in the open-source project GNSS-SDR (see https://gnss-sdr.org). Importantly, it is generic enough to describe the essential design of any GNSS receiver currently available on the market.

GNSS-SDR is a software application that runs in a common personal computer. It provides interfaces through USB and Ethernet buses to a variety of either commercially available or custom-made RF front-ends, adapting the processing algorithms to different sampling frequencies, intermediate frequencies, and sample resolutions. It also can process raw data samples stored in a file. The software performs signal acquisition and tracking of the available satellite signals, decodes the navigation message, and computes the observables needed by positioning algorithms, which ultimately compute the navigation solution.

The functional block diagram is as follows:

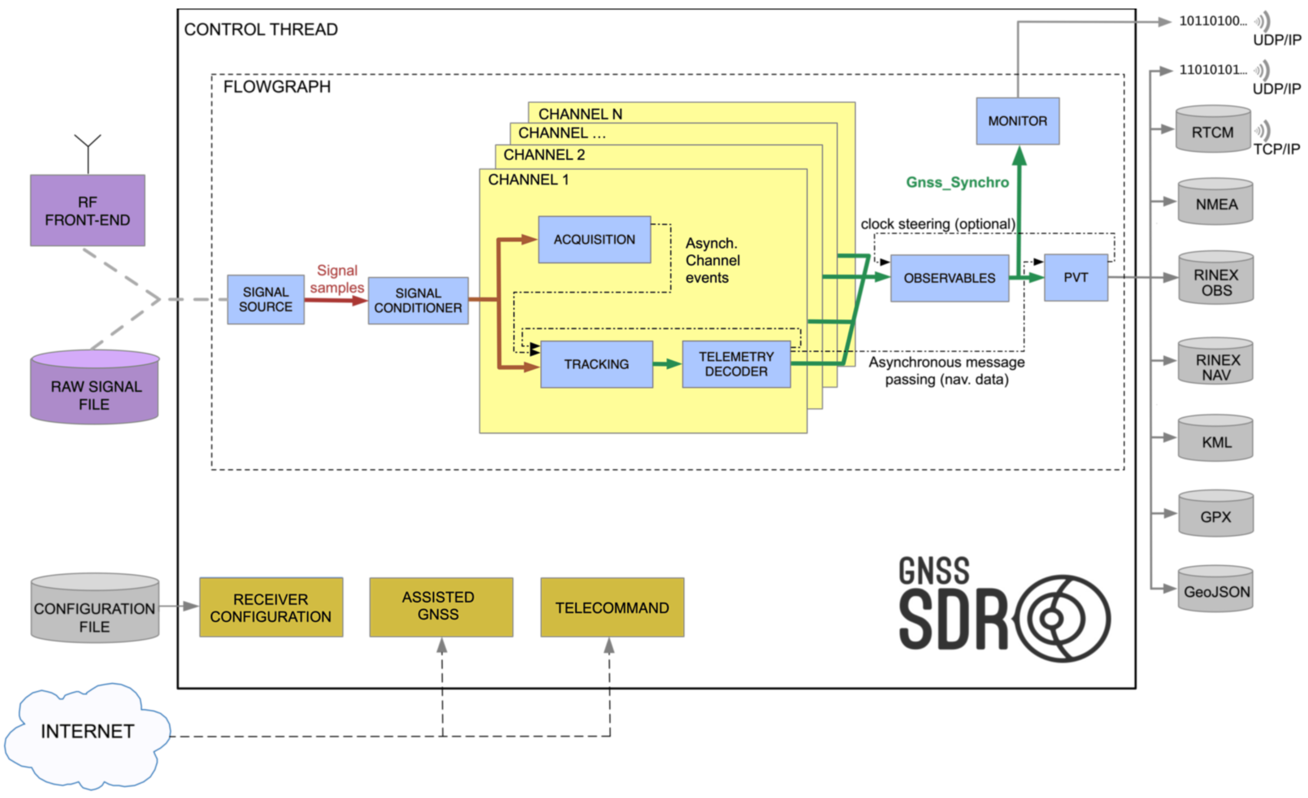


Figure 10 - Basic block diagram of the software-defined GNSS receiver.

The Signal Source is an abstract wrapper that can take either a file containing raw signal samples or be connected to a radiofrequency front-end delivering real-life signal samples at real-time. Then, the stream of samples goes through all the functional blocks (Signal Conditioner, a Channel consisting of an Acquisition, Tracking, and a Telemetry Decoder block, Observables, and PVT), the

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

last one being in charge of computing the position and time of the receiver with the information computed from signals coming from at least four GNSS satellites.

All the Signal Processing Blocks (the blue boxes in Figure 10) can be seen as empty boxes that we need to fill with algorithms and their corresponding parameters. For each of such boxes, we need to specify in the configuration file what will be their specific implementation, how many Channels (that is, how many satellites) our receiver will be able to detect and track at the same time, etc.

In this Workshop, we will learn how to fill a GNSS-SDR configuration file, fully defining our own GNSS receiver, and then running it to obtain Position, Velocity, and Time solutions. Let's first see what the role of each Processing Block is.

Signal Source

A *Signal Source* is the block that injects a continuous stream of raw samples of GNSS signal to the processing flow graph. This is an abstraction that wraps *all* kinds of sources, from samples stored in files (in a variety of formats) to multiple sample streams delivered in real-time by radiofrequency front-ends.

The input of a software receiver are the raw bits that come out from the front-end's analog-to-digital converter (ADC), as sketched in the figure below. Those bits can be read from a file stored in the hard disk or directly in real-time from a hardware device through USB or Ethernet buses.

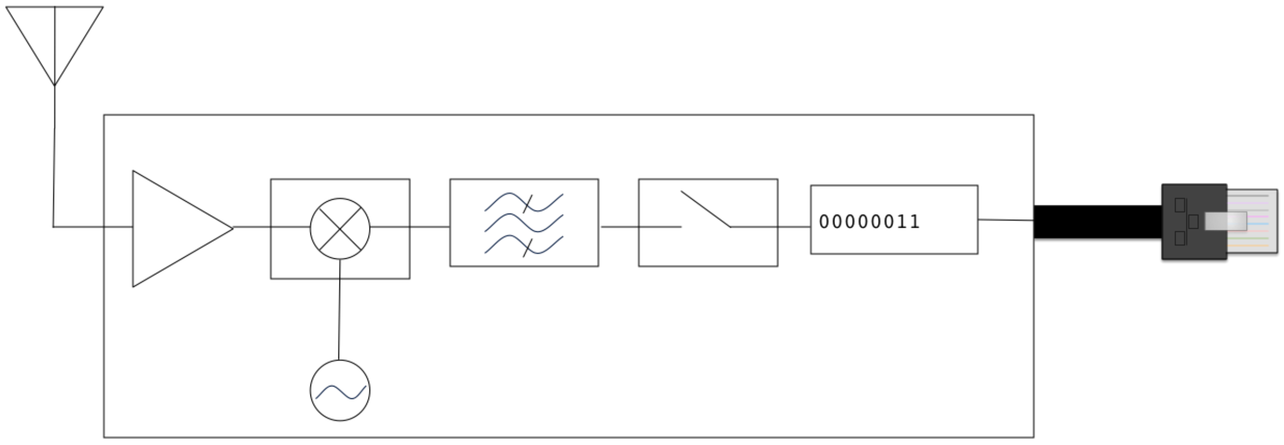


Figure 11 - Simplified block diagram of a generic radio frequency front-end, consisting of an antenna, an amplification stage, downshifting from RF to an intermediate frequency (or baseband), filtering, sampling, and an interface to a host computer for real-time processing mode, or to a storage device for post-processing.

The *Signal Source* block is in charge of implementing the hardware driver, that is, the portion of the code that communicates with the RF front-end and receives the samples coming from the ADC. This communication is usually performed through USB or Ethernet buses. Since real-time processing requires a highly optimized implementation of the whole receiver, this module also allows reading samples from a file stored in a hard disk, and thus processing without time constraints.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

Signal Conditioner

A *Signal Conditioner* block is in charge of adapting the sample bit depth to a data type tractable at the host computer running the software receiver, and optionally intermediate frequency to baseband conversion, resampling, and filtering.

Regardless of the selected signal source features, the *Signal Conditioner* interface delivers in a unified format a sample data stream to the receiver downstream processing channels, acting as a facade between the signal source and the synchronization channels, providing a simplified interface to the input signal to the downstream blocks.

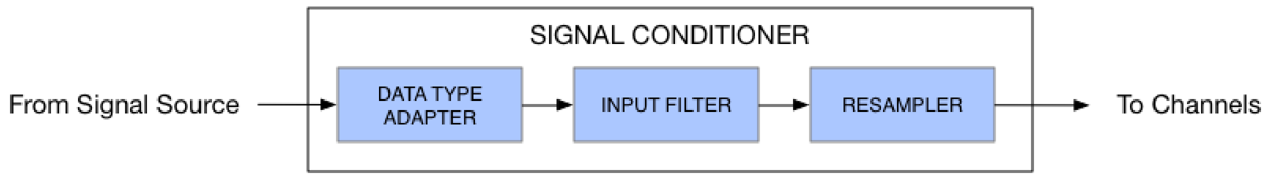


Figure 12 - The Signal Conditioner prepares the input sample stream for further processing.

Channels

Each *Channel* encapsulates blocks for signal acquisition, tracking, and demodulation/decoding of the navigation message for a single satellite. These abstract interfaces can be populated with different algorithms addressing any suitable GNSS signal. The user can define the number of parallel channels *N* to be instantiated by the software receiver.

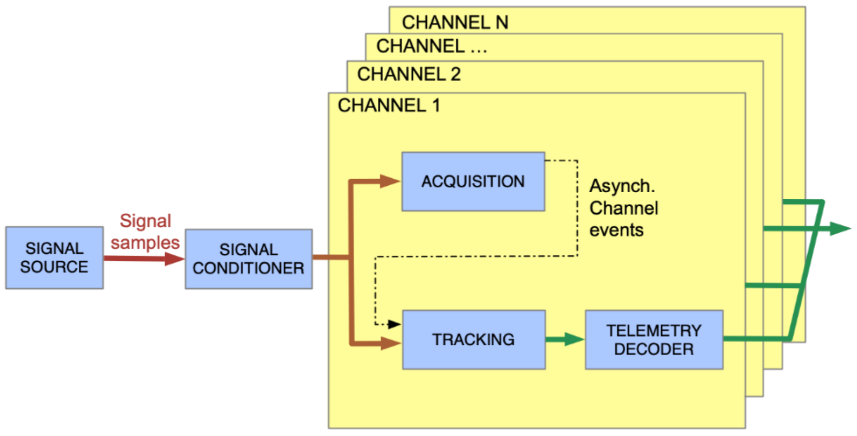


Figure 13 - Each channel can receive one signal band of a single GNSS satellite.

Each channel can receive one signal band of a single GNSS satellite.

Acquisition

The process of identifying the presence of a specific satellite's signal is known as *Signal Acquisition*. Upon positive detection, this stage also necessitates the provision of initial estimates for the time delay and Doppler shift associated with the detected signal.

The search for the existence of a particular satellite signal involves exhaustive testing across all possible combinations of time delay and Doppler shift, resulting in a comprehensive 2D grid search.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

If the peak value surpasses a predefined detection threshold, the signal is confirmed as present. The coordinates of the peak in the search grid then furnish the initial coarse estimation of the signal's time delay and Doppler shift.

Conversely, in the absence of a peak exceeding the detection threshold, the signal is deemed not present, prompting the receiver to initiate the search for another satellite.

It's important to note that the acquisition process stands out as one of the most computationally demanding stages within the entire processing chain.

Tracking

The role of a *Tracking* block is to follow the evolution of the signal synchronization parameters: code phase, Doppler shift, and carrier phase.

This block receives as an input the stream of samples coming from the Signal Conditioner block. In addition, it has an asynchronous input from the Acquisition block, which informs about the ID of a detected satellite, as well as the rough estimations about the Doppler shift and the delay of the incoming signal.

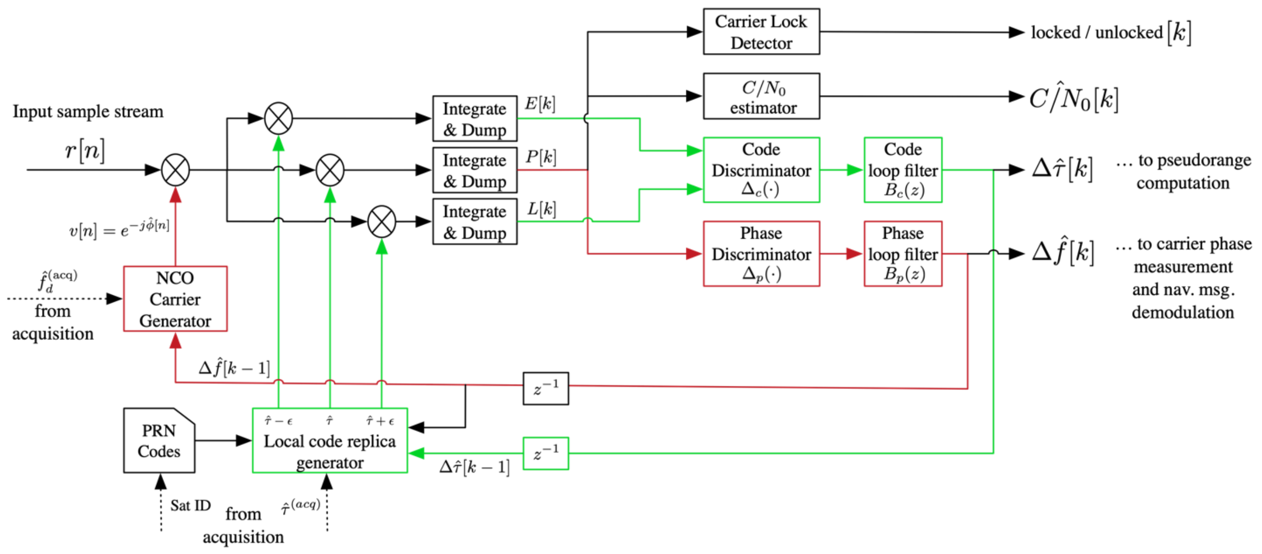


Figure 14 - Architecture of a Delay Lock Loop (DLL), in green, and Phase Lock Loop (PLL), in red, tracking the time evolution of the synchronization parameters.

Telemetry Decoder

The role of a *Telemetry Decoder* block is to obtain the data bits from the navigation data message broadcast by GNSS satellites. More precisely, to demodulate and decode such data from the received signal. Those data bits contain information such as the satellite's position, its clock information, and other system parameters.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

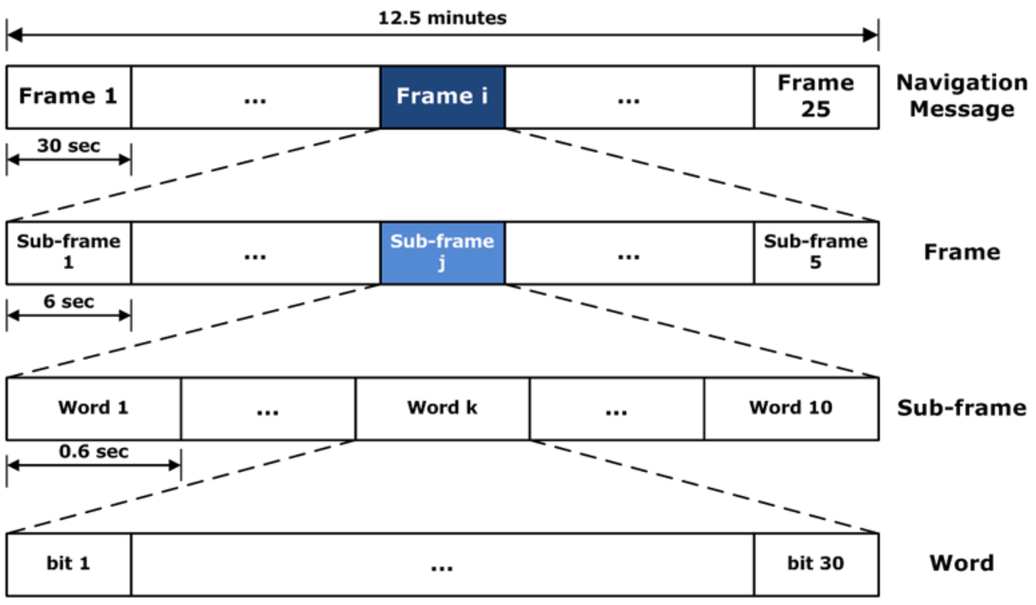


Figure 15 - Structure of the GPS L1 navigation message. Source: Navipedia.

Observables

The role of an *Observables* block is to collect the synchronization data coming from all the processing Channels, and to compute from them the GNSS basic measurements: **pseudorange**, **carrier phase** (or its **phase-range** version), and **Doppler shift** (or its **pseudorange rate** version).

- The **pseudorange measurement** is defined as the difference between the time of reception (expressed in the time frame of the receiver) and the time of transmission (expressed in the time frame of the satellite) of a distinct satellite signal. This corresponds to the distance from the receiver antenna to the satellite antenna, including receiver and satellite clock offsets and other biases, such as atmospheric delays.

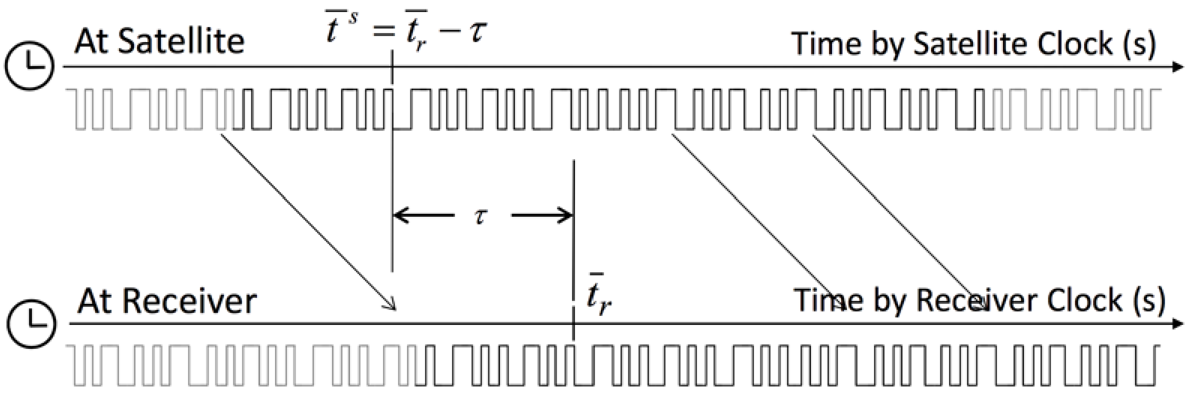

At Satellite
$\bar{t}^s = \bar{t}_r - \tau$
Time by Satellite Clock (s)
τ
At Receiver
$\bar{t}_r$
Time by Receiver Clock (s)

Figure 16 - Concept of pseudorange.

- The **carrier phase measurement** is a measurement of the beat frequency between the received carrier of the satellite signal and a receiver-generated reference frequency.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

- The **Doppler shift** is the change in frequency for an observer (in this case, the GNSS receiver) moving relative to its source (in this case, a given GNSS satellite).

The set of computed Observables are the input that feeds the algorithm that computes Position, Velocity, and Time in the next processing block.

PVT

The role of a *PVT* block is to compute navigation solutions (that is, receiver's Position, Velocity, and Time) from the Observables, and to deliver the information in adequate formats for further processing or data representation.

By processing signals from multiple satellites, the receiver determines your precise position using mathematical techniques like trilateration or multilateration and taking into account possible sources of error due to various factors like atmospheric disturbances or signal reflections.

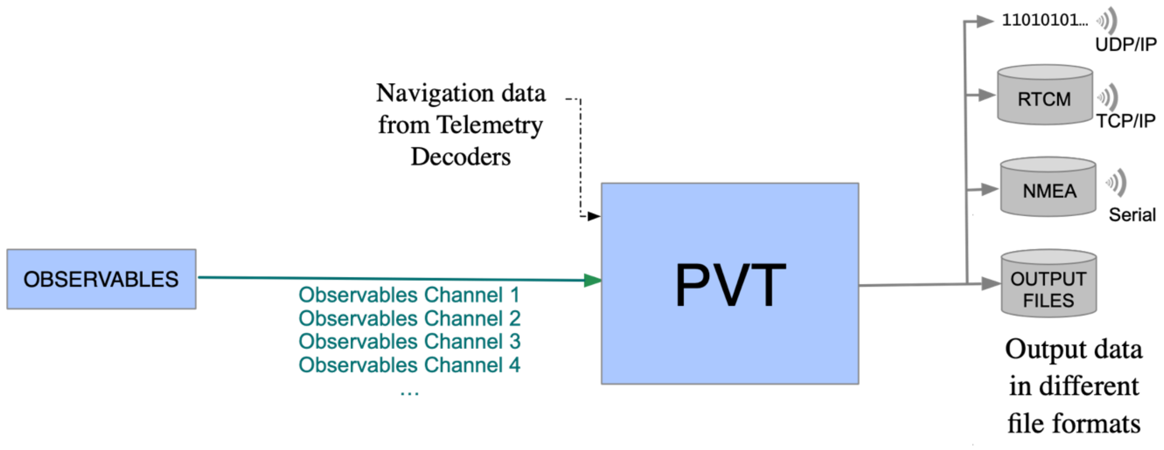


Figure 17 - The PVT block computes the Position, Velocity, and Time solution and prints it in different standard outputs such as KML, GeoJSON, or GPX, among others.

Monitor

The *Monitor* block provides an interface for monitoring the internal status of the receiver in real-time by streaming the receiver's internal data to local or remote clients over UDP. In order to keep things simple, this processing block will not be used in this Workshop.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

3. Preliminaries: Software installation & configuration

GNSS-SDR is free and open-source software, written in C++. Its source code is hosted on GitHub at https://github.com/gnss-sdr/gnss-sdr. While you can download the code, install the necessary dependencies, and build the executable yourself, this process is time-consuming and could take up a significant portion of your workshop time. To simplify and speed up the setup, a Docker image containing GNSS-SDR with all the required drivers, along with GNU Octave for graphical representation, has already been created for you.

Thus, the first step is to install the Docker engine on your computer.

Please refer to the different sections below for specific instructions based on your operating system. That is, Windows OS or Linux OS.

Once the required SW and drivers are installed and running, you can download the corresponding Docker image and run it to operate the software-defined GNSS receiver.

The workshop describes two versions: one if a suitable radio-frequency front-end and a GPS antenna are available (denoted as exercise #2 or RLT version) and a second version if the hardware is not available and the workshop is implemented using a pre-recorded signal (denoted as exercise #1 or FILE version).

Each version requires a specific configuration file.

Setup of the radio-frequency front-end

Any software-defined receiver requires two pieces of hardware to convert the received electromagnetic waves into a stream of 0s and 1s. This process involves:

- An antenna, which converts the received electromagnetic waves into voltage variations, and

Figure 18 - GPS antenna

- A device that amplifies, filters, and downconverts those voltage variations to baseband. This is followed by sampling and quantifying the signal, ultimately delivering a stream of 0s and 1s that a computer can process. This component is known as the *radio-frequency front-end*, which is precisely what the RTL-SDR v4 USB dongle performs.

Figure 19 - RTL-SDR v4 radio-frequency frontend

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

To proceed, you will need an RTL-SDR v4 dongle and an **active** GPS antenna. An *active* antenna contains a built-in Low Noise Amplifier (LNA) that requires a DC power supply delivered through the coaxial cable. Fortunately, the RTL-SDR v4 can supply this power. Simply connect your GPS to the USB dongle, and the software configuration will handle the rest. When connecting the SMA adapters, please make sure the male and female connectors are correctly aligned before **gently** tightening them. If possible, use a torque wrench to tighten the connectors. Be careful not to overload the torque lever to avoid damaging the connector.

When running the receiver, ensure that the antenna is placed where it has a clear line of sight to a significant portion of the sky.

Please note that GNSS-SDR processing requires substantial computational power, and not all machines may be capable of sustaining real-time operation.

Setup for Microsoft Windows

To run the following commands, you must be using Windows 10 version 2004 or later (Build 19041 and higher) or Windows 11. You will need to install:

1. the Windows Subsystem for Linux (WSL)
2. the ubuntu application
3. the Docker Desktop

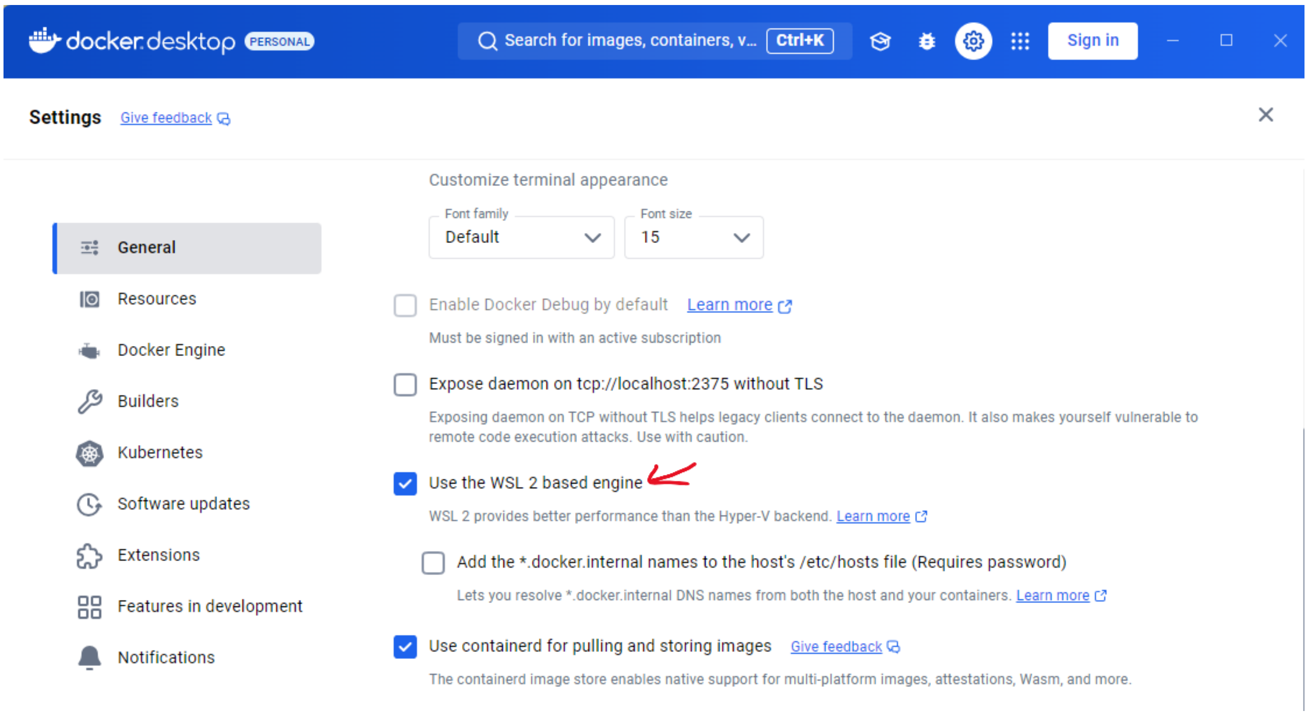
Detailed steps:

1. **Install the Windows Subsystem for Linux (WSL):** Open PowerShell or the Windows Command Prompt in administrator mode by right-clicking and selecting "Run as administrator." Enter the following command and restart your computer:

```
wsl --install
```

2. **Install Docker Desktop** from here: https://docs.docker.com/desktop/install/windows-install and configure it to use WSL (this is the default option). Make sure the “Use the WSL 2 engine” is selected:

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA



Customize terminal appearance

Font family: Default

Font size: 15

[ ] Enable Docker Debug by default Learn more

Must be signed in with an active subscription

[ ] Expose daemon on tcp://localhost:2375 without TLS

Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.

[✓] Use the WSL 2 based engine

WSL 2 provides better performance than the Hyper-V backend. Learn more

[ ] Add the *.docker.internal names to the host's /etc/hosts file (Requires password)

Lets you resolve *.docker.internal DNS names from both the host and your containers. Learn more

[✓] Use containerd for pulling and storing images Give feedback

The containerd image store enables native support for multi-platform images, attestations, Wasm, and more.

3. For the edition of GNSS-SDR configuration files, you will need a text editor. Suggested ones are Visual Studio Code https://code.visualstudio.com/, Notepad++ https://notepad-plus-plus.org/, or EditPad Lite https://www.editpadlite.com/.

4. **Download and install the USBIPD-WIN project** by obtaining the .msi file from https://github.com/dorssel/usbipd-win/releases and executing it.

5. Plug in your RTL-SDR v4 USB dongle.

6. List all USB devices connected to your system by **opening PowerShell in administrator mode** and running the following command:

```
usbipd list
```

Identify your dongle, which will appear as “Realtek Semiconductor Corp. RTL2838 DVB-T”. Make a note of its Bus and Device IDs.

7. Use the `usbipd bind` command to share the device, allowing it to be attached to WSL. Administrator privileges are required for this step. Replace `3-4` with the actual Bus and Device ID of your device:

```
usbipd bind --busid 3-4
```

8. Attach the USB device to WSL (this step does not require an administrator prompt):

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

```
usbipd attach --wsl --busid <busid>
```

Replace `` `<busid>` `` with the same Bus number - Device ID you recorded in the previous step.

9. Verify the attached USB devices from the linux WSL (ubuntu) command line by running

```
lsusb
```

in your terminal. You will get something like:

```
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 004: ID 0b05:193b ASUSTek Computer, Inc. ITE Device(8295)
Bus 003 Device 002: ID 0b05:19b6 ASUSTek Computer, Inc. N-KEY Device
Bus 003 Device 004: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
```

Your dongle should appear as "Realtek Semiconductor Corp. RTL2838 DVB-T". Record its bus number and device ID. For example, if your device is listed as `Bus 003 Device 004: ...`, then the path to your device will be `/dev/bus/usb/003/004`. **This is the path you need to use when running the Docker image, as shown in Exercise #2.**

Setup for GNU/Linux

The commands below have been tested in Ubuntu but should be similar in other GNU/Linux distributions.

1. Install and run Docker. Check https://docs.docker.com/desktop/install/linux/ for instructions.

2. Plug in your RTL-SDR v4 USB dongle.

3. Use `lsusb` to get the address of your device:

```
Bus 002 Device 002: ID 8087:8002 Intel Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 8087:800a Intel Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 004: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

The dongle is identified as "Realtek Semiconductor Corp. RTL2838 DVB-T". Take note of the bus number and the device ID. For instance, if your dongle appears at `Bus 003 Device 004: ...`

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

then your device can be found at `/dev/bus/usb/003/004`. **This is the path you need to use when running the Docker image, as shown in Exercise #2.**

Note for Raspberry Pi 5 users: You may need to blacklist the kernel module `dvb_usb_rtl28xxu`. To do this, run:

```
sudo nano /etc/modprobe.d/blacklist-rtl.conf
```

Add the following lines to the file:

```
blacklist dvb_usb_rtl28xxu

blacklist rtl2832

blacklist rtl2830
```

After saving the file and exiting the editor, reboot the system using:

```
sudo shutdown -r now
```

Once the system restarts, the RTL-SDR v4 USB dongle will be ready for use.

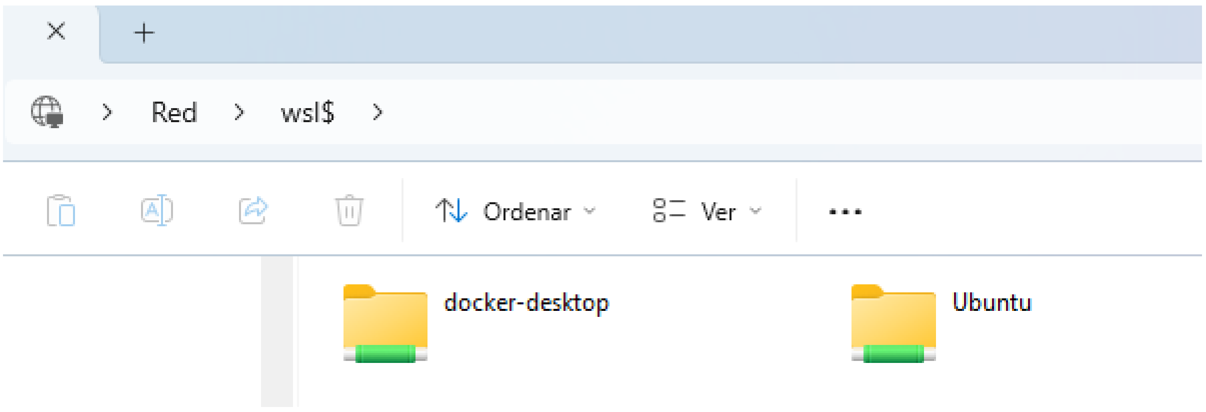
Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

4. Exercise #1: Run your software-defined GNSS receiver with a file and start getting positions

You need to download the file containing GNSS raw signal samples and copied into the selected directory (you need to create the directory "work").

Note: if your operating system is Windows you may copy the file using \\wsl$

and dragging the data files into Ubuntu/home/your_user_name/work (\\wsl.localhost\Ubuntu\home\your_user_name\work).



You can navigate and copy the configuration file as in a regular Windows' directory.

To download a file containing GNSS raw signal samples you can either do it directly from the terminal:

(assuming you are in /home/your_user_name)

```
pwd

/home/your_user_name

mkdir work

cd work

docker run -it --rm -v $PWD:/home \
  carlesfernandez/gnsssdr-telecorenta \
  wget https://sourceforge.net/projects/gnss-
sdr/files/data/2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.tar.gz


tar -zxvf 2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.tar.gz
```

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

or alternative by opening directly the link: https://sourceforge.net/projects/gnss-sdr/files/data/2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.tar.gz/download

in your browser, downloading the file, and unpacking it. This will take several minutes.

This will get you the file `2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.dat`, which contains 100 seconds of raw GNSS signal samples collected by an RF front-end centred at 1,575.42 MHz, that was delivering baseband samples at 4 MS/s, in an interleaved I&Q 16-bit integer format.

In the next step, it is provided the commands for the actual GNSS-SDR configuration. Copy the text shown in Annex I and paste it into your favourite plain text editor.

NOTE: Check that the parameter `SignalSource.filename` actually points to the name and path of your raw data file.

For more details about the configuration options for each block, check out Annex III.

Ok, let's recap. You should have:

- A Docker image containing GNSS-SDR available in your system.
- A signal source: A file named 2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.dat containing 100 seconds of raw GPS signal samples, that were grabbed by a radio frequency front-end.
- A configuration file for a GPS L1 C/A receiver that will take the file 2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.dat as its signal source.

So, we are ready to run our software-defined GPS receiver. In a linux terminal, type:

```
$ docker run -it --rm -v $PWD:/home \
  -e TZ=Europe/Madrid \
  carlesfernandez/gnsssdr-telecorenta \
  gnss-sdr --c=./my-first-GNSS-SDR-receiver.conf
```

NOTE: if you experience issues with "$PWD:/home", alternative you can use "${PWD}/home".

NOTE: Change `=./my-first-GNSS-SDR-receiver.conf` by the actual name and path of your recently created configuration file.

You should see something similar to:

```
$ gnss-sdr --config_file=./my-first-GNSS-SDR-receiver.conf
```

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

```
Initializing GNSS-SDR v0.0.18 ... Please wait.

Logging will be done at "/tmp"

Use gnss-sdr --log_dir=/path/to/log to change that.

Processing file /home/your-
username/work/2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.dat, which contains
1600000000 [bytes]

GNSS signal recorded time to be processed: 99.999 [s]

Starting a TCP/IP server of RTCM messages on port 2101

The TCP/IP server of RTCM messages is up and running. Accepting
connections ...

...
```

Then, after some seconds detecting GPS signals and decoding some frames of their navigation messages (at least, subframes 1, 2, and 3 from four satellites) ...

```
...

Current receiver time: 14 s

New GPS NAV message received in channel 3: subframe 1 from satellite GPS
PRN 20 (Block IIR)

New GPS NAV message received in channel 0: subframe 1 from satellite GPS
PRN 01 (Block IIF)

New GPS NAV message received in channel 4: subframe 1 from satellite GPS
PRN 32 (Block IIF)

New GPS NAV message received in channel 2: subframe 1 from satellite GPS
PRN 17 (Block IIR-M)

Current receiver time: 15 s

Current receiver time: 16 s

Current receiver time: 17 s

Current receiver time: 18 s

Current receiver time: 19 s

Current receiver time: 20 s

New GPS NAV message received in channel 3: subframe 2 from satellite GPS
PRN 02 (Block IIR)

New GPS NAV message received in channel 0: subframe 2 from satellite GPS
PRN 01 (Block IIF)
```

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

```
New GPS NAV message received in channel 4: subframe 2 from satellite GPS
PRN 32 (Block IIF)

New GPS NAV message received in channel 1: subframe 2 from satellite GPS
PRN 11 (Block IIR)

New GPS NAV message received in channel 2: subframe 2 from satellite GPS
PRN 17 (Block IIR-M)

Current receiver time: 21 s

Current receiver time: 22 s

Current receiver time: 23 s

Current receiver time: 24 s

Current receiver time: 25 s

Current receiver time: 26 s

New GPS NAV message received in channel 3: subframe 3 from satellite GPS
PRN 20 (Block IIR)

New GPS NAV message received in channel 0: subframe 3 from satellite GPS
PRN 01 (Block IIF)

New GPS NAV message received in channel 4: subframe 3 from satellite GPS
PRN 32 (Block IIF)

New GPS NAV message received in channel 2: subframe 3 from satellite GPS
PRN 17 (Block IIR-M)

New GPS NAV message received in channel 1: subframe 3 from satellite GPS
PRN 11 (Block IIR)

First position fix at 2013-Apr-04 06:23:31.740000 UTC is Lat = 41.2749
[deg], Long = 1.98754 [deg], Height= 100.795 [m]

Position at 2013-Apr-04 06:23:32.000000 UTC using 4 observations is Lat =
41.274888307 [deg], Long = 1.987581872 [deg], Height = 86.928 [m]

Position at 2013-Apr-04 06:23:32.500000 UTC using 4 observations is Lat =
41.274964746 [deg], Long = 1.987510141 [deg], Height = 90.557 [m]

Current receiver time: 27 s

Position at 2013-Apr-04 06:23:33.000000 UTC using 4 observations is Lat =
41.274921885 [deg], Long = 1.987605767 [deg], Height = 73.365 [m]

Position at 2013-Apr-04 06:23:33.500000 UTC using 4 observations is Lat =
41.274866502 [deg], Long = 1.987553835 [deg], Height = 83.313 [m]

Current receiver time: 28 s

Position at 2013-Apr-04 06:23:34.000000 UTC using 4 observations is Lat =
41.274904024 [deg], Long = 1.987612510 [deg], Height = 87.615 [m]

…
```

Financiado por la Unión Europea NextGenerationEU Plan de Recuperación, Transformación y Resiliencia UNICO I+D GOBIERNO DE ESPAÑA

If you see something similar to this... Yay! You are getting position fixes with your open-source software-defined GPS receiver!

```
...

Current receiver time: 1 min 40 s

Position at 2013-Apr-04 06:24:45.500000 UTC using 5 observations is Lat =
41.274821613 [deg], Long = 1.987629659 [deg], Height = 69.292 [m]

Position at 2013-Apr-04 06:24:46.000000 UTC using 5 observations is Lat =
41.274817101 [deg], Long = 1.987576895 [deg], Height = 43.517 [m]

Position at 2013-Apr-04 06:24:46.500000 UTC using 5 observations is Lat =
41.274830209 [deg], Long = 1.987583859 [deg], Height = 54.475 [m]

Stopping GNSS-SDR, please wait!

Total GNSS-SDR run time: 37.106698 [seconds]

GNSS-SDR program ended.

Stopping TCP/IP server on port 2101

$
```

Now you can examine the processing outputs in the folder from which you invoked GNSS-SDR:

- A .kml file.
- A .geojson file.
- A .gpx file.
- A .nmea file.
- Observation and Navigation RINEX files.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

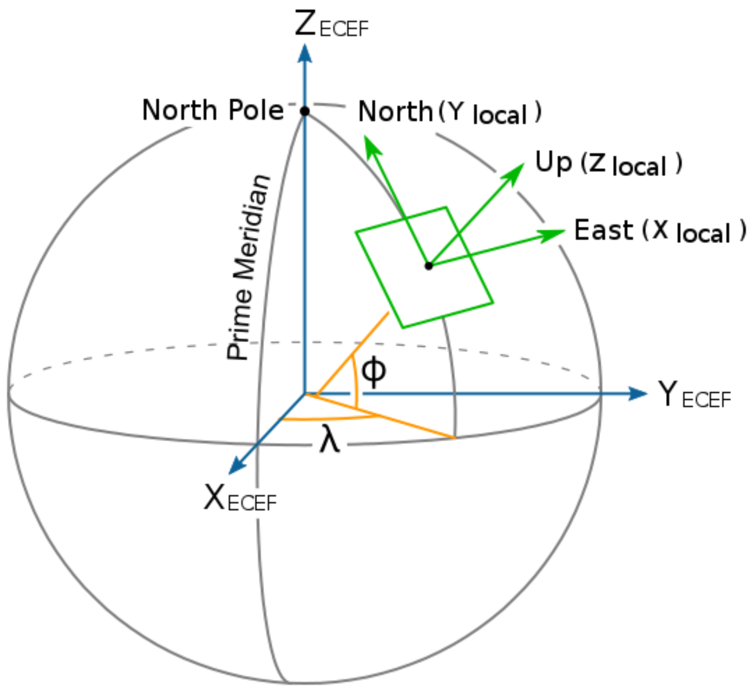


Figure 20 - Relationship between Earth Fixed Earth Centered coordinates (X,Y,Z), geographic coordinates (latitude Φ and longitude λ) and local coordinates (East, North, Up). Source: Wikipedia.

Try to open the .kml file with Goole Earth: go to https://earth.google.com/web/ , then click on File and choose "Import KML / KMZ File".

5. Exercise #2: Run your software-defined GNSS receiver with a radio-frequency front-end

In the case you have the hardware device, first you need to generate the configuration file for running the receiver with the RTLSDR v4 RF front-end. Copy the content in Appendix II to a text file and name it `rtl.conf`.

Make sure the device is correctly attached: from linux (ubuntu) terminal type:

```
lsusb
```

Once you have stored the configuration in a file, you can execute the receiver from either a linux terminal (ubuntu) or a PowerShell console by doing:

```
docker run -it --rm -v $PWD:/home \
  --device=/dev/bus/usb/xxx/yyy:/dev/bus/usb/xxx/yyy \
  -e TZ=Europe/Madrid \
  Carlesq
fernandez/gnsssdr-telecorenta \
  gnss-sdr --c=./rtl.conf
```

In this command, `xxx` and `yyy` represent the bus number and device ID obtained in the installation section. Make sure the path to the configuration file `rtl.conf` is correctly specified in the previous command.

Adjust your TZ identifier (see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones) as required to obtain time solutions adapted to your time zone. In this example it is set to Madrid.

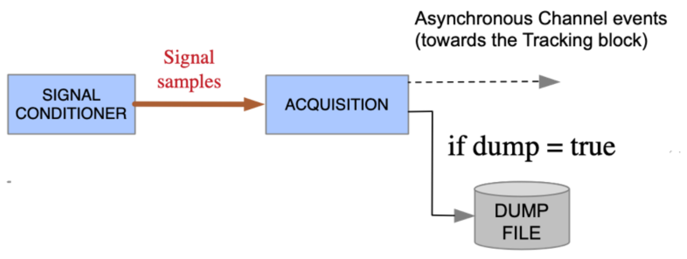
The software-defined GNSS receiver should now start operating. Stop the receiver at any time by pressing key `q` and then key `[ENTER]`.

Please note that the antenna must have a clear line of sight to a significant portion of the sky to receive signals from enough satellites for computing Position, Velocity, and Time (PVT) solutions.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

6. Exercise #3: Plot signal acquisition

Some processing blocks have the capability of storing its internal data in adequate file formats, which then we can inspect and learn about how the block is internally operating. This is the case of the Acquisition block:



The configuration parameters for enabling this feature are:

- dump: [true, false]: If set to true, it enables the Acquisition internal binary data file logging. It defaults to false.
- dump_filename: If dump is set to true, name of the file in which internal data will be stored. This parameter accepts either a relative or an absolute path; if there are non-existing specified folders, they will be created. It defaults to ./acquisition, so files with name ./acquisition_G_1C_ch_N_K_sat_P.mat (where N is the channel number defined by dump_channel, K is the dump number, and P is the targeted satellite's PRN number) will be generated.
- dump_channel: If dump is set to true, channel number from which internal data will be stored. It defaults to 0.

In order to test that, add these lines to your configuration file (you can save it with a different name):

```
Acquisition_1C.implementation=GPS_L1_CA_PCPS_Acquisition

;... (other parameters) ...

Acquisition_1C.dump=true

Acquisition_1C.dump_filename=acq_dump

Acquisition_1C.dump_channel=0
```

Then, run the receiver again, and after the processing you will get .mat files storing the results obtained from the Acquisition block corresponding to channel 0.

The role of an Acquisition block is the detection of the presence/absence of signals coming from a given GNSS satellite. In the case of a positive detection, it should provide coarse estimations of the code phase (that is, time delay) and the Doppler shift, yet accurate enough to initialize the

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

delay and phase tracking loops. This is done in a two-dimensional search space, where the two dimensions are time and frequency. Let's try to visualize this.

The acquisition grid can be plotted from MATLAB or Octave with this script:

```
% This script plots the acquisition grid obtained by GNSS-SDR

% in MATLAB/Octave without using the GUI interface.

%

% Requires previous processing with the following lines

% in the GNSS-SDR configuration file:

% Acquisition_1C.dump=true

% Acquisition_1C.dump_filename=acq_dump

% Acquisition_1C.dump_channel=0

%

% You can call it by doing:

%

% docker run -it --rm -v $PWD:/home carlesfernandez/gnsssdr-telecorenta
octave --no-gui acquisition_grid.m

%

% Get a nice PDF with the result:

% docker run -it --rm -v $PWD:/home carlesfernandez/gnsssdr-telecorenta
epspdf acq_result.eps acq_result.pdf

%

% SPDX-FileCopyrightText: 2024, Carles Fernandez-Prades
<carles.fernandez@cttc.es>

% SPDX-License-Identifier: MIT


load('./acq_dump_G_1C_ch_0_9_sat_22.mat');  % <-- REPLACE THIS FILE WITH
YOU ACTUAL FILENAME


f = int32(-doppler_max):int32(doppler_step):int32(doppler_max)-
int32(doppler_step);

tau = linspace(0, 1023, size(acq_grid, 1));


hf = figure('visible', 'off');   % <-- REPLACE WITH hf = figure; IF YOU
ARE IN THE GUI INTERFACE
```

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

```
surf(f, tau, acq_grid);


xlabel('Doppler [Hz]');

ylabel('Delay [chips]');

title(strcat('Acquisition grid for GPS sat #', num2str(PRN)));


hh = findall(hf, '-property', 'FontName');

set(hh, 'FontName', 'Times');

print(hf, 'acq_result.eps', '-depsc');

close(hf);
```

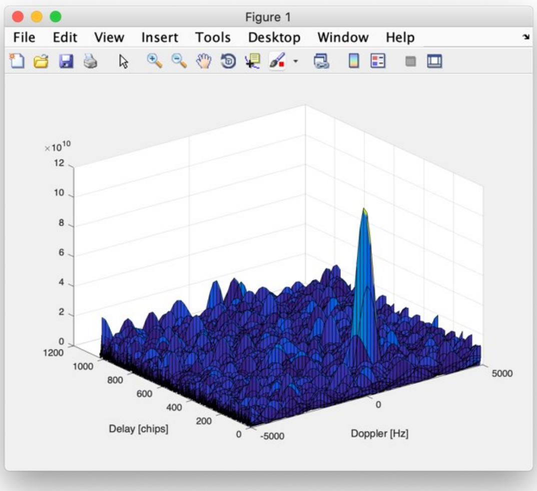
If you copy this script in a file and name it `acquisition_grid.m`, then you can run:

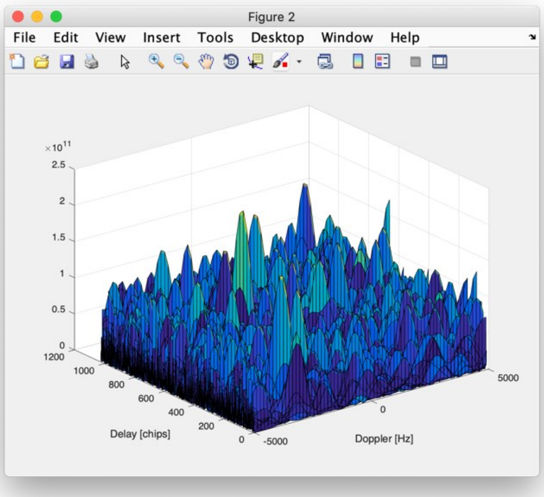
```
docker run -it --rm -v $PWD:/home carlesfernandez/gnsssdr-telecorenta
octave --no-gui acquisition_grid.m
```

and then convert the result to a PDF file for easier visualization:

```
docker run -it --rm -v $PWD:/home carlesfernandez/gnsssdr-telecorenta
epspdf acq_result.eps acq_result.pdf
```

You should see something like:





If a distinct peak is observed, it signifies the detection of a satellite signal. The position of this peak enables the estimation of both the Doppler shift and the time delay. Conversely, if no peak

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

stands out from the background, the satellite being searched for has not been detected. In this case, it is recommended to try observing a different channel.

7. Conclusions

An open-source software-defined GNSS receiver, such as GNSS-SDR, is an invaluable resource for both learning and research. It enables users to configure receivers with custom features, study the effects of specific parameters, experiment with novel algorithms, and gain insight into the inner workings of an actual GNSS receiver. This Workshop covered only a small portion of what is possible with GNSS-SDR, and we encourage students to delve further into the exciting field of satellite-based navigation and conduct additional experiments on their own.

For a foundational overview of software-defined radio and its applications to GNSS, refer to https://gnss-sdr.org/docs/fundamentals/. The complete GNSS-SDR configuration documentation can be accessed from https://gnss-sdr.org/docs/sp-blocks/

For completeness, this document includes an Annex with detailed documentation on GPS L1 receiver configuration.

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

8. Annex I: Configuration file for FILE processing version

Copy this text into **my-first-GNSS-SDR-receiver.conf** text file with the preferred text editor. Make sure the data file path (in yellow) matches the actual location.

```
[GNSS-SDR]


;######### GLOBAL OPTIONS ##################

GNSS-SDR.internal_fs_sps=2000000


;######### SIGNAL_SOURCE CONFIG ############

SignalSource.implementation=File_Signal_Source

SignalSource.filename=./data/2013_04_04_GNSS_SIGNAL_at_CTTC_SPAIN.dat

SignalSource.item_type=ishort

SignalSource.sampling_frequency=4000000

SignalSource.samples=0


;######### SIGNAL_CONDITIONER CONFIG ############

SignalConditioner.implementation=Signal_Conditioner

DataTypeAdapter.implementation=Ishort_To_Complex

InputFilter.implementation=Pass_Through

InputFilter.item_type=gr_complex

Resampler.implementation=Direct_Resampler

Resampler.sample_freq_in=4000000

Resampler.sample_freq_out=2000000

Resampler.item_type=gr_complex


;######### CHANNELS GLOBAL CONFIG ############

Channels_1C.count=8

Channels.in_acquisition=8

Channel.signal=1C


;######### ACQUISITION GLOBAL CONFIG ############
```

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

```
Acquisition_1C.implementation=GPS_L1_CA_PCPS_Acquisition
Acquisition_1C.item_type=gr_complex
Acquisition_1C.pfa=0.01
Acquisition_1C.doppler_max=10000
Acquisition_1C.doppler_step=250
Acquisition_1C.blocking=true

;######### TRACKING GLOBAL CONFIG ############
Tracking_1C.implementation=GPS_L1_CA_DLL_PLL_Tracking
Tracking_1C.item_type=gr_complex
Tracking_1C.pll_bw_hz=40.0;
Tracking_1C.dll_bw_hz=4.0;

;######### TELEMETRY DECODER GPS CONFIG ############
TelemetryDecoder_1C.implementation=GPS_L1_CA_Telemetry_Decoder

;######### OBSERVABLES CONFIG ############
Observables.implementation=Hybrid_Observables

;######### PVT CONFIG ############
PVT.implementation=RTKLIB_PVT
PVT.positioning_mode=Single
PVT.output_rate_ms=100
PVT.display_rate_ms=500
PVT.iono_model=Broadcast
PVT.trop_model=Saastamoinen
PVT.flag_rtcm_server=true
PVT.flag_rtcm_tty_port=false
PVT.rtcm_dump_devname=/dev/pts/1
PVT.rtcm_tcp_port=2101
PVT.rtcm_MT1019_rate_ms=5000
PVT.rtcm_MT1077_rate_ms=1000
```

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

```
PVT.rinex_version=2
```

Financiado por la Unión Europea
NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

9. Annex II: Configuration file for RTL device version

Copy this text into **rtl.conf** text file with the preferred text editor

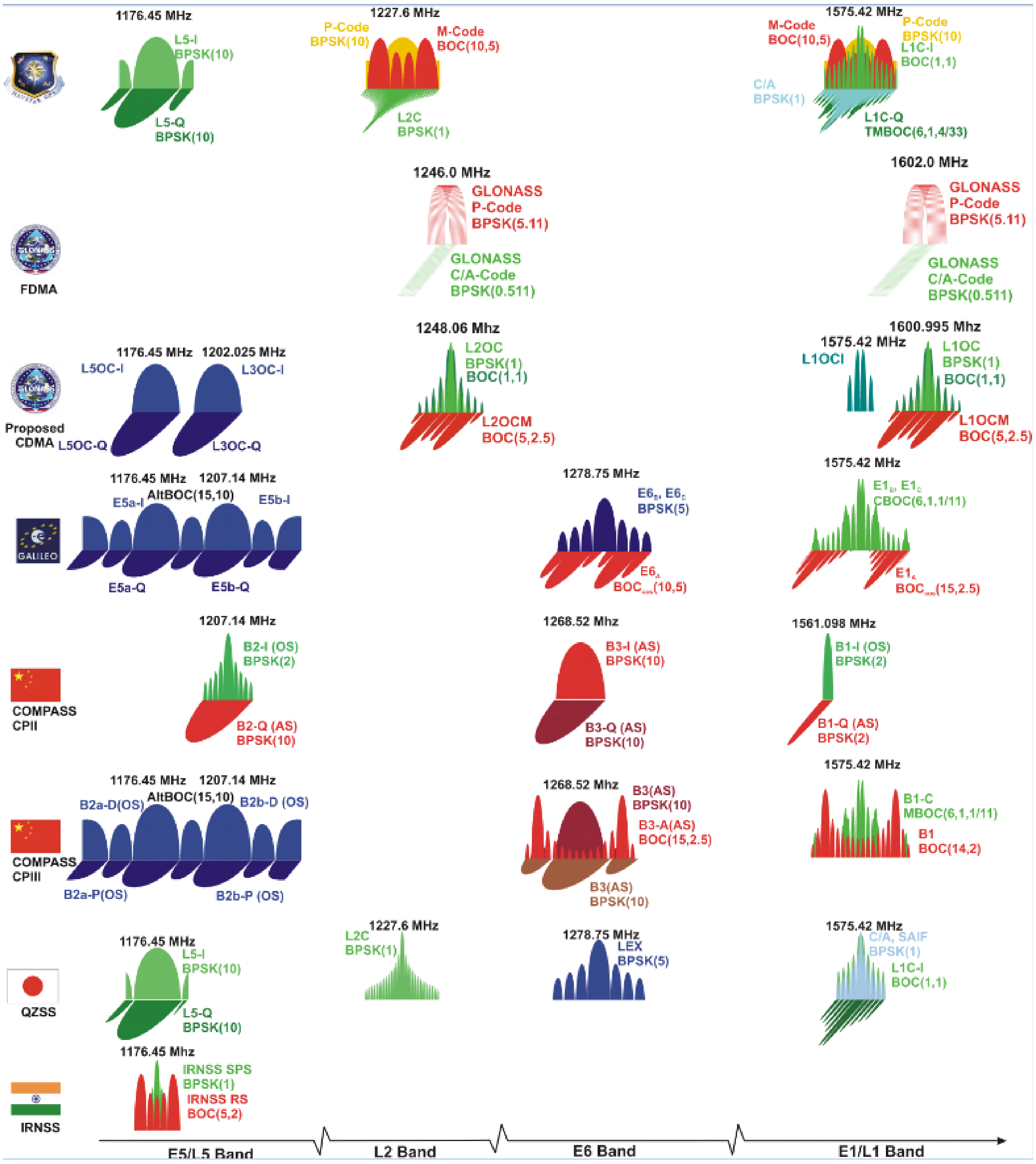
```
[GNSS-SDR]


;######### GLOBAL OPTIONS ##################
GNSS-SDR.internal_fs_sps=2000000


;######### SIGNAL_SOURCE CONFIG ############
SignalSource.implementation=Osmosdr_Signal_Source
SignalSource.item_type=gr_complex
SignalSource.sampling_frequency=2000000
SignalSource.freq=1575420000
SignalSource.AGC_enabled=true
SignalSource.osmosdr_args=rtl,bias=1


;######### SIGNAL_CONDITIONER CONFIG ############
SignalConditioner.implementation=Pass_Through


;######### CHANNELS GLOBAL CONFIG ############
Channels_1C.count=8
Channels.in_acquisition=1


;######### ACQUISITION GLOBAL CONFIG ############
Acquisition_1C.implementation=GPS_L1_CA_PCPS_Acquisition
Acquisition_1C.item_type=gr_complex
Acquisition_1C.pfa=0.01
Acquisition_1C.doppler_max=5000
Acquisition_1C.doppler_step=250


;######### TRACKING GLOBAL CONFIG ############
Tracking_1C.implementation=GPS_L1_CA_DLL_PLL_Tracking
```

Financiado por la Unión Europea NextGenerationEU · Plan de Recuperación, Transformación y Resiliencia · UNICO I+D · GOBIERNO DE ESPAÑA

```
Tracking_1C.item_type=gr_complex

Tracking_1C.pll_bw_hz=40.0

Tracking_1C.dll_bw_hz=4.0


;######### TELEMETRY DECODER GPS CONFIG ############

TelemetryDecoder_1C.implementation=GPS_L1_CA_Telemetry_Decoder


;######### OBSERVABLES CONFIG ############

Observables.implementation=Hybrid_Observables


;######### PVT CONFIG ############

PVT.implementation=RTKLIB_PVT

PVT.positioning_mode=Single

PVT.output_rate_ms=100

PVT.display_rate_ms=500

PVT.iono_model=Broadcast

PVT.trop_model=Saastamoinen

PVT.show_local_time_zone=true
```

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

10. Annex III: GNSS frequency bands



Source: Navipedia.

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

Advanced research for everyday life