Software Defined Radio for processing GNSS signals

BACHELOR'S THESIS

Sara MARTÍNEZ GUTIÉRREZ

Jean-Michel FRIEDT, Associate Professor.
École Nationale Supérieur de Mécanique et des Microtechniques (ENSMM)

Academic year 2014-2015

Acknowledgements

I would like to express my gratitude to all that have made possible this opportunity of developing my bachelor's thesis in ENSMM, which collaborates with my university ETSEIB. I would like to concretely refer to the research institute FEMTO-ST and the funding company First TF. But most specially, to my tutor Prof. Jean-Michel Friedt, from whom I have learned everything about the project, transmitting me day by day his work enthusiasm and his infinite patience. I do not have more words but to say Prof. Jean-Michel Friedt is a perseverance role model from whom I have not only learned about the technical aspects but personal. Moreover, I would like to thank Prof. Gonzalo Cabodevila for his help in the thesis subject search and for his devotion when helping us in the last part of the project in automatic's matters. Further, mention Prof. Enrico Rubiola and Prof. Pierre-Yves Bourgeois who have also contributed in this study solving doubts that arose on the way and suggesting possible improvements.

Finally, I would like to wholeheartedly thank my supportive family, my boyfriend, who always makes me continue forward, and my unforgettable friends I have met in Besançon without whom this experience would not have been the same.

Contents

Abstract

GPS satellites are fitted with atomic clocks, in which it relapses the main objective of this project, to recover some of their accuracy and stability on a ground based receiver.

This project describes the fundamentals of GPS signals, the assembly of the installation implemented to process them in software and the corresponding experiments. In order to achieve the software processing, a USB DVB-T dongle is connected to an active antenna and to the computer.

As mentioned, one of the purposes is also to understand how a GPS can be implemented by software as a the substitution of a big part of the hardware that makes it impenetrable, as they are black boxes of integrated circuits, and expensive.

It is known that a Global Navigation Satellite System (GNSS) software-defined open source receiver has already been created by people in Barcelona in “Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)”, a testbed for GNSS signal processing since it can be customized in every way. It has been used at some intermediate steps of the study while executing parallel experiments in the course of understanding how a GPS signal is digitally processed. In the meantime, some experiments have also been performed only employing hardware before implementing them in software, so that the concepts are visually reflected. When realizing software experiments, an interface called GNURadio has been used because of its enormous implementation of signal processing blocks. GNURadio can be used with external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. Nevertheless, various simulations in the GNU (Octave software environment) have also been executed as processing in real time has not been considered a goal.

However, to successfully accomplish the demodulation of the navigation data, which will contribute to restore the accuracy and stability of the satellites clocks that have sent it, the carrier frequency needs to be perfectly recovered, being this last point where the final aim of the project falls on.

List of figures and tables

List of figures

List of tables

List of abbreviations

GNSS	Global Navigation Satellite System
RF	Radio Frequency
GPS	Global Positioning System
TOA	Time Of Arrival
CDMA	Code Division Multiple Access
C/A	Coarse Acquisition
LFSR	Linear Feedback Shift Register
BW	Bandwidth
PRN	Pseudo Random Noise
BPSK	Binary Phase Shift Keying
I	Inphase signal
Q	Quadrature signal
DVB-T	Digital Video Broadcasting-Terrestrial
DC	Direct Current
LNA	Low Noise Amplifier
LO	Local Oscillator
PLL	Phase-Locked Loop
ADC	Analog-to-Digital Converter
COFDM	Coded Orthogonal Frequency Division Multiplexing
USB	Universal Serial Bus
IF	Intermediate Frequency
NF	Noise Figure
FM	Frequency Modulation
AM	Amplitude Modulation
LPF	Low Pass Filter
VCO	Voltage-Controlled Oscillator
FLL	Frequency-Locked Loop
DCO	Digitally Controlled Oscillator
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FT	Fourier Transform
DLL	Delay-Locked Loop

1. Introduction

From the inside of the Time and Frequency Department from the *École Nationale Supérieure de Mécanique et Microtechniques (ENSMM)*, the aim of this project is to reach the carrier frequency recovery from a GPS signal. The purpose of implementing a GPS in software emerges from the desire of being able to manipulate and change functions rapidly, which in an educational environment seems more suitable than analog processing. First TF is Labex (Laboratoire d'Excellence) that has funded this project, which grants encompassing innovative developments in fields related to time and frequency metrology. Regrouping all the renowned French TF laboratories (Syrte Paris Observatory, Nice Observatory, Besançon Observatory, FEMTO-ST Temps et Fréquence) it is constituted by the following partner laboratories: APC, ARTEMIS, IEMN, LAAS, LCAR, LKB, LP2N, LPMAA, PIIM, USN, XLIM. One aspect of disseminating the acquired knowledge is through the summer school EFTS. As part of this teaching activity, it was decided to expand the current GPS data usage tutorial with a detailed analysis of GPS signal acquisition in the framework of TF applications. Software Defined Radio provides an opportunity for scalable and cost effective demonstration of GPS signal acquisition and processing [1].

The Global Positioning System (GPS) is a constellation of 24 satellites arranged in 6 orbital planes with 4 satellites per plane [2]. They are at 20000 km of altitude and their orbital period is 12 hours [3], providing location and time information anywhere on Earth where there is an unobstructed line of sight to four or more GPS satellites [4].

GPS utilizes the concept of Time Of Arrival (TOA) ranging to determine user's position. This concept entails measuring the time it takes for a signal transmitted by an emitter at a known location to reach a user receiver, i.e. its signal propagation time. Afterwards, to obtain the emitter-to-receiver distance, this time interval is multiplied by the speed of the signal [2].

GPS satellites broadcast on L1 1575,42 MHz (10,23 MHz · 154) and L2 1227,60 MHz (10,23 MHz · 120) carrier frequencies, and to do so they use a method called Code Division Multiple Access (CDMA). CDMA consists on every satellite transmitting on L1 and L2, but each one with its own ranging code, the so-called C/A code and P(Y) code [5].

In this case we will be working always at L1 as they are narrow band signals containing the civilian Coarse Acquisition (C/A) Code as well as the military Precise (P(Y)), while L2 contains only the P(Y) code. However, the P code has not been implemented while

working on this project, since it is $6.1871 \cdot 10^{12}$ bits long and only repeats once a week [6].

The C/A code is a deterministic sequence called pseudorandom noise with, as its most remarkable feature, the orthogonality. It implies that these sequences only correlate when they are exactly aligned with its own copy. It is generated by two 10-bit linear feedback shift registers (LFSR) of a maximal length of $2^{10} - 1 = 1023$. However, it is not until chapter 7 that the detailed explanation regarding their generation is done. 1023 bit C/A code sampled at a 1,023 MS/s rate, results into a sequence repetition period of 1 millisecond [5, 6]. Thanks to processing C/A code in software and to its sampling rate and repetition period, a gain called Compression (coding) gain contributes in the processing, which is calculated as the following formula indicates.

$BW \cdot \tau = 10 \cdot log_{10} [(2 \cdot 1{,}023 \text{ MHz}) \cdot (0{,}001 \text{ s})] = 33 \text{ dB}$.

From an overall of 37 PRN C/A codes, 32 are assigned to 24 GPS satellites, leaving from the 33 to the 37 reserved to other uses, including ground transmitters. Moreover, 34 and 37 are identical [7].

It is thanks to the comparison between the pseudo random codes that satellite send with their identical copy codes found on the receiver, that a GPS receiver is able to determine which is the satellite sending the information and the travel time of a signal from a satellite to the receiver [8]. This comparison is accomplished by the cross-correlation, operation that is properly defined in chapter 6.

To achieve the navigation data recovery from the signal, the carrier frequency and the ranging code need to be previously removed, remaining then the Binary Phase Shift Keying (BPSK) modulation, which encodes the bits of the navigation data.

Concretely, for this particular study, the GPS L1 band signal consists of 50 bit-per-second (bps) navigation data modulated by a periodic 1023-chip C/A code at a chip rate of 1,023 megachip-per-second (Mcps) by a BPSK modulation technique [5].

A schematic summarizing the explanation realized just above is shown in figure 1.1.

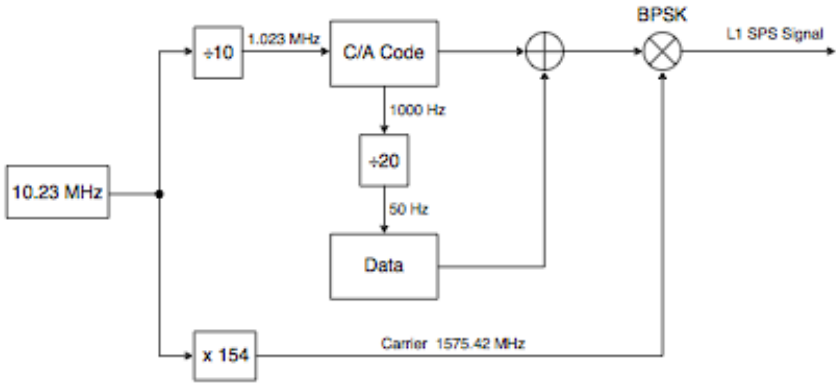

Figure 1.1: The generation of GPS signals at the satellites simplified for this study [1]

The DVB-T dongle employed to perform the digitally processing streams raw I&Q samples to the host computer. These DVB-T dongles were originally designed, as the name suggests, to be television receivers. It allows users to watch over-the-air DVB-T European broadcast television on their personal computers. Although it was not its original functionality when conceived, its use in combination with an open source software receiver for GNSS signal processing, constitute one of the cheapest alternative for the treatment of GPS signals [9].

The DVB-T dongle has a fist part of analog signal processing where the front-end performs the RF to baseband, and a second part where the analog-to-digital conversion takes part.

By implementing a GPS in software, it is obtained more flexibility at the time of changing anything and also when upgrading the code if some failure is detected or some other function wants to be implemented. This is specifically achieved by using Software Defined Radio, which in its purest form might consist of only an antenna connected to an analog-to-digital convert chip with all the rest of the processing, including filters, mixers, amplifiers, modulators, demodulators, detectors, etc, implemented in the digital domain by means of software on a personal computer or embedded system [10, 11].

2. Previous installation and specifications

In order to be sure that the previous installation of the few needed hardware implemented before the dongle is properly done we proceed to verify it.

The receiver architecture's consists of an active antenna (20 euros) connected to a bias T, and a DVB-T dongle (10 euros), resulting the overall of the hardware installation in 30 euros.

Since the DVB-T dongle was initially created to be a television receiver, it has a low sensitivity for what we wonder if it is sensitive enough for receiving GPS signals, whose satellites are located 20000 km above the surface of the earth.

Here, a simplified version of an RF signal model that represents the contribution from an M number of satellites, from which has only been considered their line-of-sight input.

$$X_{RF}(t) = R\left[\sum_{i=1}^{M} \alpha_i(t) S_{T,i}(t-T_i(t)) e^{j2\pi\left(f_c+f_{d,i}(t)\right)t}\right]+n(t)$$

Its parameters represent: $\alpha_i(t)$ the complex amplitude, $S_{T,i}$ the satellite baseband signal that arrives to the Earth's surface at -155 dBW (-125 dBm), $T_i(t)$ the time delay, f_c the carrier frequency, $f_{d,i}$ the Doppler frequency and n(t) the additive white Gaussian noise plus multipath or interference which are unwanted terms in the signal [9].

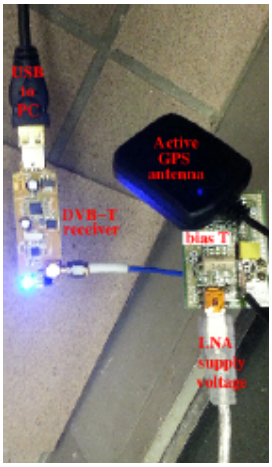

Figure 2.1: Hardware installation

The active antenna, in charge of receiving the GNSS signals, is located at the roof of the building and it is connected to a bias T (a three port network conceptually composed of a capacitor and an ideal inductor), which is in charge of powering the circuit safely at a DC voltage of 5 Volts [12]. This antenna includes a first amplification stage using a Low Noise Amplifier (LNA) (27 dB gain), an impedance and a filter. Just after the bias T, the following element in the RF chain is the DVB-T dongle.

The DVB-T dongle includes a front-end IC (an interface between the user and the back end) and the RTL2832U IC chip.

The front-end programmable tuner with 64 MHz to 1700 MHz input frequency range (with a gap between 1104 and 1253 MHz) is in charge of amplifying, thanks to a RF LNA that can provide a gain up to 30 dB, and filtering the RF signal. However, its most important function is to downconvert it to baseband using a single I/Q mixing stage, being the gain provided by the active mixers up to 12 dB. After

the mixing step between the incoming signal and, on one hand, an unchanged local copy of the carrier frequency, and on the other hand a 90º phase-shifted local copy of the carrier frequency, I&Q components are obtained. This local copy of the carrier frequency, is generated thanks to the Local Oscillator (LO) that is created by the integrated circuit. It uses an on–chip 16 bit Fractional-N Phase Locked Loop (PLL) digitally reconfigurable to cover the entire DVB- T band. The PLL uses an external reference signal from an on-board 28,8 MHz crystal oscillator. The crystal has been tuned to 28,8 MHz because it is a requirement for the USB connector. Since the reconfigurable registers of the fractional PLL and the divider are integers, there will be a tune frequency error in the LO frequency.

As the crystal oscillator is operating in both the front-end and the analog-to-digital converter (ADC) a frequency offset due to its capacitors or the temperature drift can also be introduced [9]. These sources of frequency offset are more detailed on chapter 4.

The local oscillator is the one in charge of generating a signal which mixed with a signal of interest, will transform this signal of interest into a different one [13]. This is the reason why we need to be aware of the offset that the carrier signal generated in the LO may have, as it can change the value from the resulting signal.

The I&Q analog outputs feed directly the RTL2832U chip in charge of the DVB-T COFDM demodulation, which acts in this case as an ADC (clocked by a 2^{nd} fractional PLL) that resamples the internal samples performed at 28,8 MS/s to obtain the desired sampling rate. This is achievable since the data are read through the USB from the host computer, making software-defined signal processing possible. The RTL2832U IC uses a dual-channel ADC and obtains 8-bit I&Q-samples sampled at a highest sampling rate allowed of 2,4 MS/s to not loose samples. Consequently, it has been chosen a sampling rate of 2 MS/s to stay below this threshold to remain safe. Just in some situations the ADC can work at a rate of 3,2 MS/s without dropping samples. However, when sampling at 2MS/s, because the PRN code frequency as the PRN code is 1,023 MHz means that the Nyquist criteria is not met. Nyquist Criteria establishes that a sufficient sample-rate without loosing information is therefore 2·bandwidth samples/second, or anything larger [24]. In this particular case of study, since the characteristics of the signal are well known (square wave sampled at 1,023 MS/s) and the purpose is to compare patterns, not to reconstruct a signal from zero, 2 MS/s still allows us to obtain reasonable results.

The 8 bits ADC are related to the quantization, i.e. to the minimum variation of the voltage that can be detected. It is restricted by the number of bits that it possesses, as mentioned, 8 bit for the ADC of the E4000 dongle. $\frac{1}{2^8}$ is the minimum variation voltage that it can detect, which will affect if the GPS signal that it is aimed to be detected is not separated at least of $\frac{1}{2^8}$ V from another possible strong signal.

When using the E4000 dongle, it is known that apart from the RF LNA it also possesses an Intermediate Frequency (IF) gain, which can be each set at 39 dB and 32 dB maximum gain thanks to the software application. But after the experience, the maximum usable gain for the combination of E4000 + RTL2832U is 42 dB with a noise figure of NF = 7 dB. Though, the front-end gain required to receive GNSS signals is in the order of 70 dB for a 8 bit ADC, this is the reason why an active antenna is needed [9].

The noise figure is defined as the difference in decibels (dB) between the noise output of the actual receiver to the noise output of an “ideal” receiver with the same overall gain and bandwidth when the receivers are connected to matched sources at the standard noise temperature T0 (usually 290 K) [14].

All the mentioned values in this chapter of the dongle’s components correspond to the dongle model Elonics E4000.

3. Dongle tests

3.1 Dongle's local oscillator frequency deviation

To determine the dongle's local oscillator offset, we compare the L1 carrier frequency (1575,42 MHz) generated with the signal generator device from the laboratory (ROHDE&SCHWARZ SMA 100 A. SIGNAL GENERATOR FROM 9 kHz to 3 GHz), with the L1 carrier frequency generated at the dongle's local oscillator.

The signal generator device of the laboratory has been considered as the reference because of its high precision, and moreover, because it has previously been adjusted when comparing 10 MHz of its generated frequency with the 10 MHz frequency standard from the Rubidium Atomic Clock. The Rubidium Atomic Clock is the most inexpensive, compact, and widely used type of atomic clock [15].

To accomplish the comparison, the process followed is to introduce the 1575,42 MHz output of the already calibrated signal generator Rohde & Schwarz into the input of the dongle. Afterwards, the output of the dongle is directly connected to the computer where the frequency is plotted thanks to GNURadio companion. If the frequency generated in the dongle's LO is also well adjusted at 1575,42 MHz, the graph would show a peak in the frequency domain centered at 0, since the dongle performs the mixing step between carriers and if they have the same value the result is a null value. The result obtained is a peak at 100 kHz, representative of the difference between the frequency generated by the Rohde & Schwarz and the one that the dongle's LO create.

This happens to be an offset of:

$|\frac{\Delta f}{f}| = \frac{100\ kHz}{1575{,}42 \cdot 10^3\ kHz} \cdot 10^6\ ppm$ = 63,5 ppm. Actual offset = -63,5 ppm.

3.2 Dongle's sensitivity

Before trying to start treating real GPS signals for the first time, it has been considered convenient to previously work with radio frequency modulated in Frequency Modulation (FM) and Amplitude Modulation (AM), since the verification is thought to be easier than with real GPS signals as the thermal noise takes an important role on them.

The aim of these two tests (with FM and AM) is to conclude whether these dongles are sensitive enough to receive GPS signals or not, as its initial purpose is to receive

Digital Video Broadcasting-Terrestrial (DVB-T), which does not require that much of accuracy in the signal processing. The sensitivity of an electronic device stand for the minimum magnitude of input signal required to produce a specified output signal having a specified signal-to-noise ratio, or other specified criteria [16]. The most worrying fact is that the signal arrives to rise above the thermal noise due to the poor hardware implemented, the poor power received and the restrictions established by the DVB-T dongle sampling rate.

More specifically, the aim of this verification is to determine which of the three types of dongles that the laboratory has on its possession is the one that needs the least received radiofrequency carrier power to obtain a demodulated signal 10 dB above the thermal noise, in other words, the one that needs the least incoming power to reach a demodulated signal Signal-to-Noise-Ratio of 10 dB.

Thermal noise is the electronic noise generated by the thermal agitation of the charge carriers inside an electrical conductor at equilibrium, which happens regardless of any applied voltage. When limited to a finite bandwidth, thermal noise has a nearly Gaussian amplitude distribution [17].

The frequencies chosen to simulate the signals for the experiment are 137 MHz (which is used by low earth orbiting satellites and is also allocated to aircraft communication at AM band), 434 MHz (a frequency used for low powered devices), 1090 MHz (a new Automatic Dependent Surveillance-Broadcast protocol used for locating planes) and 1575,42 MHz (the carrier frequency used at L1 by GPS). These frequencies are generated using the ROHDE&SCHWARZ SMA 100 A. SIGNAL GENERATOR FROM 9 kHz to 3 GHz, which is set to one of the four frequencies described, with FM or AM modulation in each convenient case. Its output is directly connected to the dongle, which is most importantly in charge of creating the I&Q values and performing the Analog to Digital conversion.

The circuit that allows us to do the measurements is the following. In the case of AM modulation is it shown in figure 3.1.

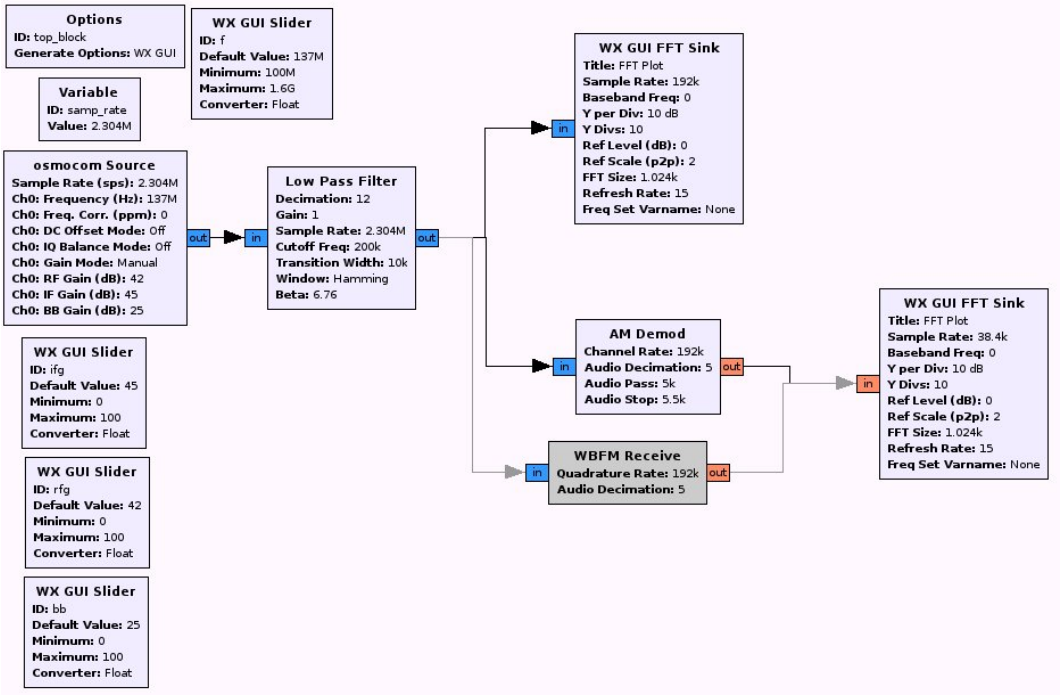

Figure 3.1: E4000 dongle receiving a 137 MHz frequency with AM modulation

In the case of FM modulation it is shown in figure 3.2.

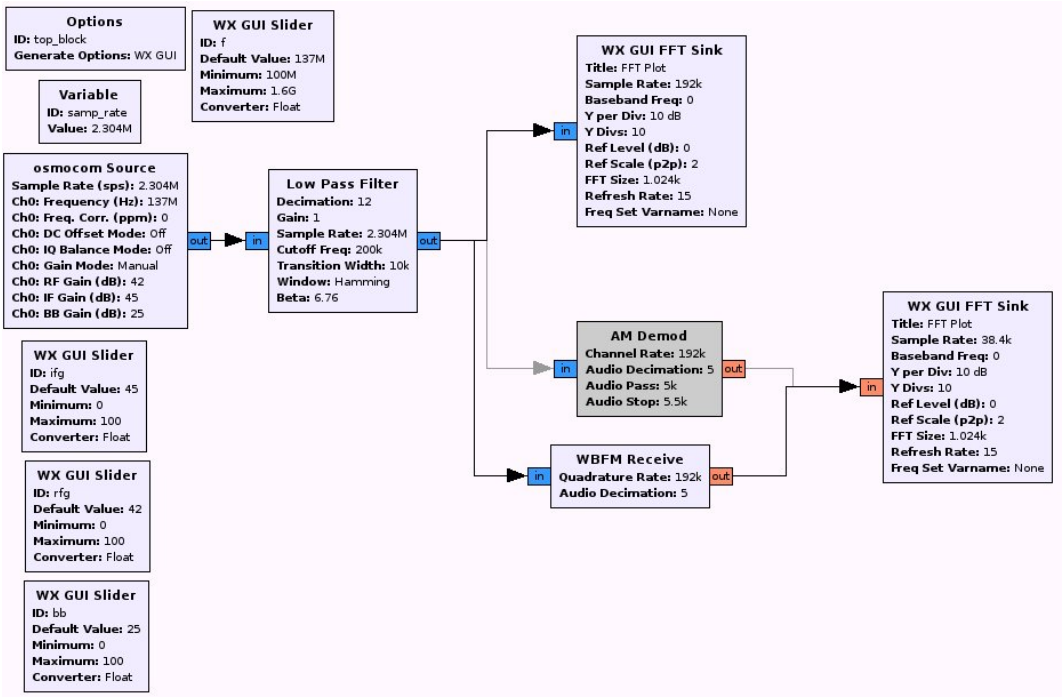

Figure 3.2: E4000 dongle receiving a 137 MHz frequency with FM modulation

The obtained values when performing the tests are the following shown in table 3.1 in the case of AM modulation and in table 3.2 for FM modulation.

Receiver	137 MHz	434 MHz	1090 MHz	1500 MHz
E4000	-109	-110	-101	-101
R820T	-117	-119	-113	**-110**
FC0013	-116	-115	-80	-62

Table 3.1: Input power (dBm) needed for a demodulation signal SNR of 10 dB (AM)

Receiver	137 MHz	434 MHz	1090 MHz	1500 MHz
E4000	-105	-105	-96	-95
R820T	-112	-112	-109	**-102**
FC0013	-112	-111	-76	-58

Table 3.2: Input power (dBm) needed for a demodulation signal SNR of 10 dB (FM)

When working with *GNURadio* companion, a program used for signal processing, three characteristic parameters for each dongle need to be firstly adjusted in the *osmocom Source* block.

Related to the GNURadio schematic plots, the *osmocom Source* block is in charge of determining the sampling rate (defined by the *variable* block), which has been set to 2,304 MS/s. Its three gain parameters, radio frequency gain (rfg), intermediate frequency gain (ifg) and baseband gain (bbg), are defined using *WX GUI Slider* blocks so that their value can afterwards be changed on the plots. For the definition of the parameter f (frequency) it has also been used the *variable* block.

The first of the mentioned parameters is the rfg and it is the only one that affects the three dongles. It needs to be set at its maximum, since the dongles' amplifiers must offer their maximum gain as no additional amplifier has been set right after the antenna. One of the positive aspects of having to take this requirement is that all the dongles are tested under the same conditions but at the same time, it is possible that the noise is also incremented since gain is pushed to its maximum. We select to compare the various dongles fitted with various front-end receivers by setting their gain to maximum value. This setting is different from one hardware implementation to another but should yield the estimation of the minimum incoming RF power needed for signal processing (demodulation).

The second parameter to appear in GNURadio is the parameter ifg, which only seems to affect the E4000 dongle. And finally, the last parameter that does not seem to affect any of them is the bbg.

The values established for each parameter and for each dongle type are the following.

Receiver	rfg	ifg	bbg
E4000	42	45	-
R820T	49	-	-
FC0013	19	-	-

Table 3.3: Gain set in [dB] for each dongle type

All measurement were performed with a 3 kHz modulation frequency, which indicates how often the carrier changes. When working with Amplitude Modulation, its depth has been set to a 30%, representative of the percentage between the maximum and minimum amplitude that shape the message wave envelope. While, when working with Frequency Modulation it has been given an interval of 5 kHz, which expresses the difference between the maximum and minimum frequencies that characterize the Frequency Modulation.

The *Low Pass Filter* block presents the most relevant following features: it decimates (because the raw RF signal contains too much information for the targeted purpose, yet the dongle hardware cannot sample below 1,5 MS/s as it is one of its limitations), it has a Cutoff Frequency, which defines the starting point in where the slope of the transfer function of the filter starts to descend, and a Transition Width, which expresses a frequencies range between the passband and the stopband.

Thanks to the decimation implemented inside the low pass filter, the bandwidth is previously low passed and then decimated, in this proper order, so that when the decimation is performed the noise is not able to rise due to aliasing (a phenomenon that incorporates information and noise inside the decimated bandwidth coming from the previous one). This is an important characteristic because if the signal is not firstly low pass filtered, the noise area keeps constant.

The *WBFM Receive* and *AM Demod* blocks are in charge of producing in each case the corresponding demodulation.

The 2,304 MS/s sample rate is the one that defines the original bandwidth in order to have ± 2,304/2 MHz at each sides of the carrier frequency. After the first decimation by 12 performed by the LPF, the bandwidth reduces to 192 kHz, which after the second decimation by 5 achieved by the demodulation blocks results in 38,4 kHz in the WX GUI FFT Sink.

The final value of 38,4 kHz is considered correct and enough to have a sufficient number of modulation frequency copies, since it is where the information remains. After the demodulation the signal appears as a function of the frequency (X axis) and dB (Y axis).

Three dongles have been tested, in order to see which one suited its function with the least power. To conclude with this experiment by observing the results obtained on table 3.1 and 3.2, it can be said that the dongle R820T (-102 dBm in FM and -110 dBm in AM) appears to be the one that needs the least incoming RF power to obtain a demodulated signal 10 dB above the thermal noise. However, the chosen dongle to continue performing the rest of the experiments throughout this project is the E4000 (-95 dBm in FM and -101 dBm in AM), as later on this study, a platform called gnss-sdr will be used to obtain some desired information, as it is a specific software that treats GPS signals. The information provided by gnss-sdr is related with the features of the front-end from E4000, since the DVB-T dongles differ in their front-end characteristics but not in their RTL2832U chip, which they all share. The FC0013 is the least suitable hardware since its sensitivity strongly drops above 1 GHz.

4. Sources of frequency offset

4.1 Doppler shift

The Doppler shift is the change in frequency of a wave for an observer moving relative to its source, or vice versa. In the case where the transmitter is approaching the receiver, each wave takes slightly less time to reach the observer than the previous wave. Hence, the time between arrivals of successive wave crests at the observer is reduced, causing an increase in the frequency. In the other case where they would separate, the result would be the opposite as the just mentioned [18].

Since GPS satellites are not geo-stationary but orbiting satellites, GPS receivers experience the Doppler shift due to satellite motion [19].

Currently, there are 24 GPS satellites that form the whole network. As it has already been mentioned, they are in GPS constellation orbits, which are at an altitude of 20000 km from the ground and, distributed in a way that at least 4 satellites are always visible from any point on the Earth at any time. Their orbital speed is 14000 km/hour and they present an orbital period of 12 hours [2].

Due to their kinematic characteristics, the received frequency from the satellites diverges from the one emitted in a predictable manner. To estimate these frequency variations, the Doppler-shifted transmissions must be searched and the demodulator in the receiver's architecture needs to be able to reproduce a copy of the carrier frequency well synchronized in phase with the incoming signal [19].

4.1.1 Doppler shift calculation (Δf)

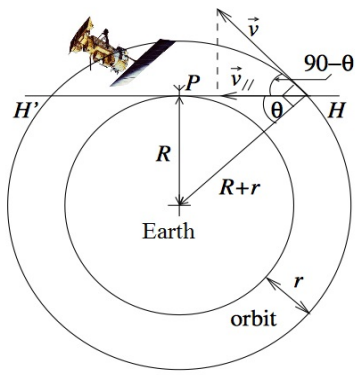

Figure 4.1: Decomposition of the satellite's velocity [20]

Satellites have a tangential velocity to its orbit ($\overrightarrow{v_{sat}}$) but only its component pointing to the receiver is the one that contributes to the Doppler shift. Consequently, the Doppler shift is 0 when the satellite is vertically above the receiver, since there is no pointing to the receiver component of the satellite's velocity.

R (radius of the Earth) = $6400 \cdot 10^{3}$ m

r (altitude of the satellite from the Earth surface) = $20000 \cdot 10^{3}$ m

$\overrightarrow{v_{sat}}$ = (2π · (6400 + 20000)) / 12 = 13800 km/h = 3840 m/s

f = 1575,42 · 10^3 KHz

$|\overrightarrow{c}|$ = $3 \cdot 10^8$ m/s

Pointing to the receiver component of the satellite's velocity: $|\overrightarrow{v_{sat\,||}}|$ = | $\overrightarrow{v_{sat}}$| · cos (90-Θ) = $|\overrightarrow{v_{sat}}|$ · sin (Θ) = $\frac{|\overrightarrow{v_{sat}}| \cdot R}{R+r}$ = 930 m/s

Δf = $\frac{|\overrightarrow{v_{sat\,||}}|}{|\overrightarrow{c}|}$ · f = **4,8 KHz ≈ 3,1 ppm**

4.1.2 Doppler shift graph using gnss-sdr software for signal processing

The first contact with gnss-sdr software is made in order to plot a Doppler shift graph. Figure 4.2 shows at the ordinate axis the frequency offset due to the Doppler shift [Hz], and at the abscissa axis the time [s].

In order to process a radiofrequency signal, the carrier frequency must be removed. This processing step is accelerated if the bias of the local dongle oscillator is known. Gnss-sdr provides a tool for estimating such a bias by comparing the incoming GPS signal with the predicted Doppler shift. This tool from the front-end-cal software integrated in gnss-sdr is also useful for assessing the Doppler shift of each visible satellite.

Thanks to the calibration (front-end-cal) software provided by gnss-sdr, which enables the user to estimate its approximate location and to obtain a first assessment of the visible satellites, some processed data is recorded in temporary files during 3 or 4 seconds every 5 minutes for a whole day. The calibration software from gnss-sdr is able to know the visible satellites when the user location is set due to its communication with the google or nokia network.

The plot (shown in figure 4.2) is performed because of the correct signal processing gnss-sdr does, since it supplies the values of the measured, predicted and corrected Doppler shift.

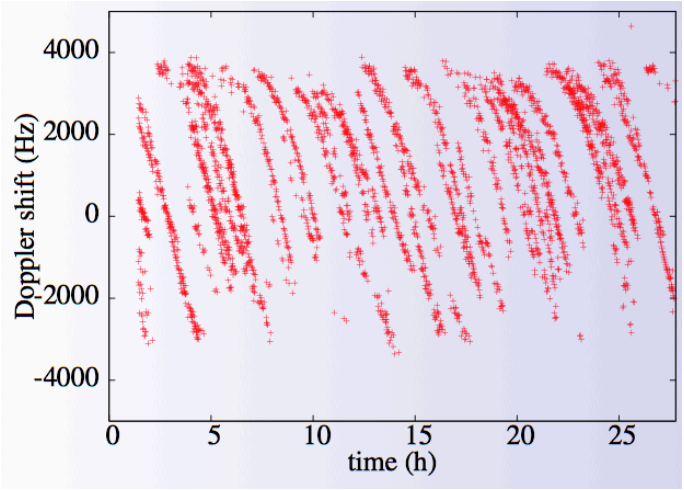

Figure 4.2: Doppler shift graph using data processed by front-end-cal.conf from gnss-sdr

The plotted values at the graph are coherent with the maximal manually calculated Doppler shift value, since it has been taken for its calculation the maximum velocity of the satellite. They are consistent as they are inside the range of ± 5 kHz.

4.2 DVB-T Dongle

The DVB-T dongle is a source of frequency fluctuations in the case of having temperature drifts or a wrong electronic setup of its LO, or it originates a bias due to the Fractional-N PLL in the front-end and the 2nd fractional PLL that clocks the ADC.

4.2.1 Temperature drift

The offset due to the temperature drift is observed to be around ± 2ppm, since some data has been recorded during one week using the front-end-cal software and it has been compared the frequency offset [ppm] as a function of temperature [ºC]. It introduces phase noise, a fluctuation in the obtained values with a mean that equals 0. The result is consistent with quartz resonator properties operating close to room temperature.

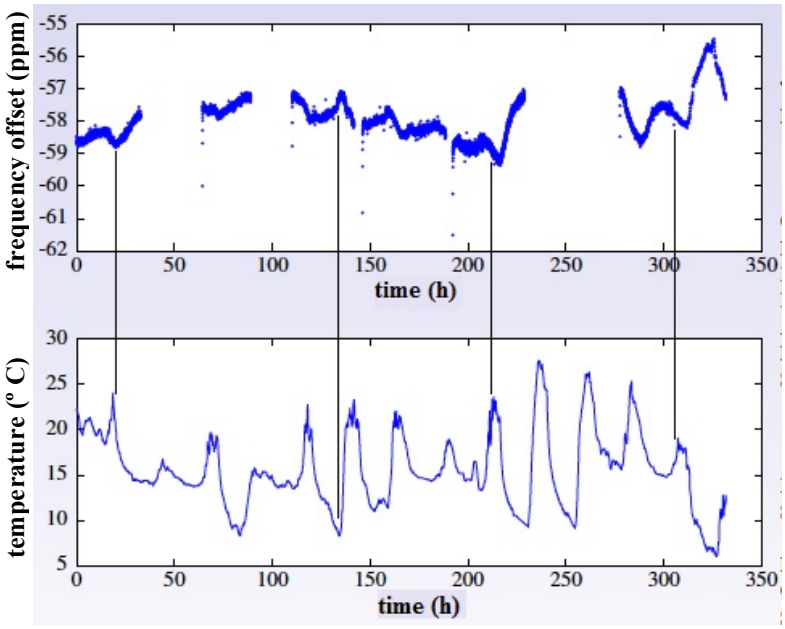

Figure 4.3: Frequency offset due to temperature drift

4.2.2 Electronic setup of the oscillator

However, the electronic setup of the oscillator contributes in a bigger frequency offset value around ± 90 kHz (± 60 ppm) due to how its capacitors have been adjusted. The oscillator consists of a saturated amplifier, which is digitally implemented by an inverter (NOT logic gate) that sets the gain to compensate for the resonator loss and contributes with a 180º phase shift, a resonator, in charge of setting the generated frequency that contributes with a -90º phase shift, and two capacitors, which properly adjusted end up compensating for some error produced in the tuned frequency on the quartz resonator. Moreover, the capacitors contribute with a phase shift too, which affects to the resonator's phase shift allowing it to change its contribution from -90º to -180º phase shift, finally making the circuit to have a total phase shift of 0º, which is the purpose of the periodic loop [21].

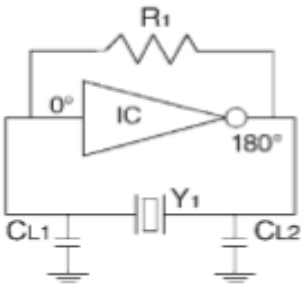

Figure 4.4: Oscillator [22]

4.2.3 Dongle's front-end Fractional-N PLL and 2nd fractional PLL in the ADC

A much less significant frequency offset source but that is, nevertheless, mentionable is the frequency drift that can be obtained because of the frequency steps performed by the fractional-N PLL in the dongle's front-end when generating the carrier frequency. A fractional-N PLL is composed of a reference oscillator that generates a stable frequency, which divided by M and compared to the target oscillator (VCO) that has also been divided but by N, yield N/M. Since N and M are integers, the front-end presents steps in the frequency generated. In the case of the R820T dongle the steps experimentally found are 439,45 Hz [23], while in the E4000 they are of 102,53 Hz [9].

The ADC, also clocked by the LO, possesses its own 2nd fractional PLL that restricts its precision in 6,8665 Hz frequency steps [8].

5. Single carrier signal processing

5.1 Analog experiment

The experiment using analog devices is thought to do the signal processing steps more understandable before implementing them in software, since an analog experiment seems to be more visual. The aim is to remove the BPSK modulation by squaring the signal and also to recover the carrier through the use of a PLL.

The experiment has been performed with a 1,21 GHz carrier frequency instead of 1,57542 GHz because of the restriction offered by the 2,4 GHz band-pass filter used further in the experiment.

However, in this experiment the PRN codes have not been introduced yet but a function generator is applied on its place instead. This frequency modulation is 1023 kHz as it is for C/A codes also, which can be approximated to 1 MHz.

The schematic representing the performed experiment is the following.

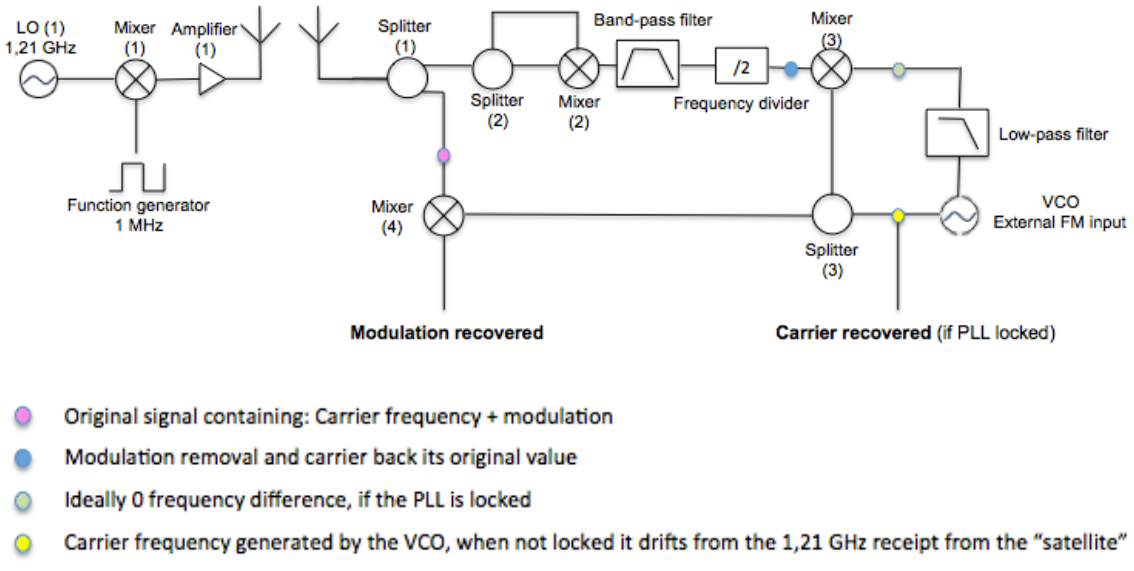

Figure 5.1: Single carrier signal processing without "local" LO

To make the comprehension of the schematic easier, it will be explained following specific areas.

Firstly, on the left side, the satellite has been simulated by mixing a function generator with a signal generator. The devices perform the modulated information onto the carrier frequency.

Analog mixers are non-linear electrical circuits, which produce the sum of the frequencies mixed and also their difference. If they are composed of diodes, the original frequencies are also output [25]. However, multiplicative mixers are also used in conjunction with oscillators to modulate signal frequencies, since IF is injected in the port I (shown in figure 5.2) and it is the L inductance's sign (positive or negative half-period) who determines if IF will be running in one or the inverse direction through the two diodes reverse biased or forward biased to saturation. Until they do not reach a definite forward voltage they do not start to conduct significantly [26, 27]. Figure 5.2 shows a mixer schematic.

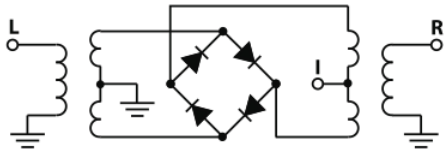

Figure 5.2: Mixer schematic

Then, the signal reaches the receiver, which in the real experiment has been implemented with a wire connecting the output of the amplifier (1) with the input of the splitter (1), while in the schematic it appears two antennas. Here is the starting point of the BPSK removal process by squaring, performed by the three blocks in between the splitter (2) and the mixer (3) on the serial chain of the receiver.

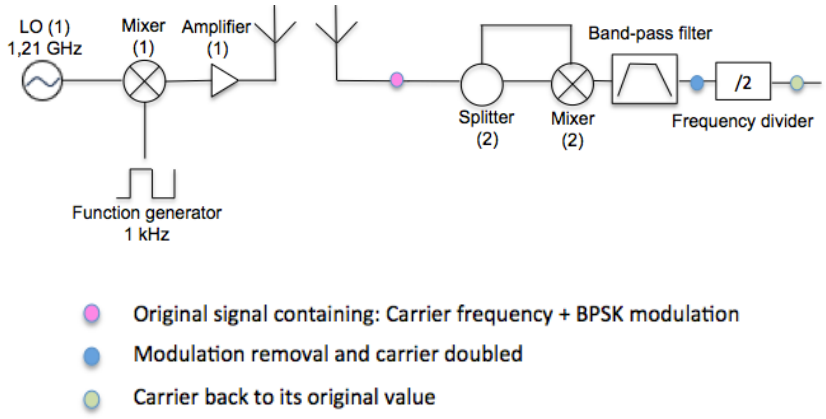

Figure 5.3: BPSK modulation removal without "local" LO

The squaring function, which is represented by the splitter (2) and the mixer (3), is in charge of removing the 180º phase shift, characteristic from the BPSK modulation. The output of the block is actually the double of the frequency, because of the characteristics that the sinusoidal signals offer.

$\sin^2(x) = \frac{1}{2} - \frac{1}{2}\cos(2x)$; where the initial frequency was x and after the squaring operation it is obtained the double, 2x.

The next step is to band-pass filter the spectrum, so that it is only kept the double of the frequency and the rest, which is a little leftover from the original carrier frequency, is removed.

Finally, just before the mixing step (3), there is a frequency divider by two, so that the frequency is returned back to its initial value since the carrier frequency has been moved to 2,4 GHz. If no VCO were implemented in the experiment, the carrier recovery would have already been reached at this point, since no drifting between both frequency synthesizers (the one that simulates the satellite and the one for the VCO) could occur. However, the modulation could not be recovered because the RF power that would reach the mixer (4) would not be enough to saturate it, and consequently, allow it to perform its function. It is at this point where the carrier recovery start through a PLL.

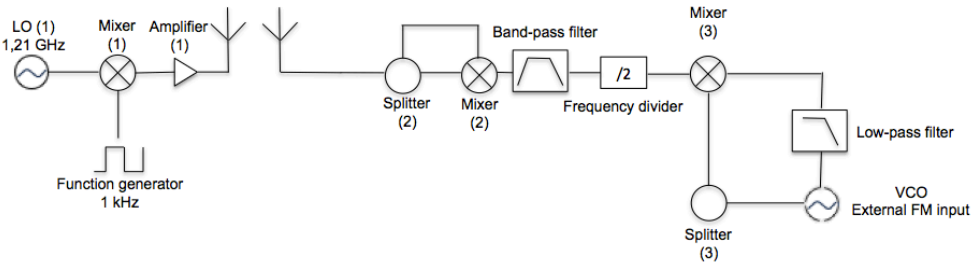

Figure 5.4: Carrier recovery without “local” LO

Secondly, on the right side of the signal processing circuit there is the already mentioned *Phase Lock Loop*. Its function is to compare the phase of the incoming periodic signal, from which its modulation has been removed, with the phase of the signal that the VCO generates. Once the mixing step is accomplished, its output is “fed back” toward the input forming a loop. It is more worth comparing phases rather than frequencies, which would involve implementing a Frequency Lock Loop (FLL) rather than a PLL, because since the phase is the time integral of the frequency the obtained results are more reliable as the error accumulates.

A VCO is an electronic oscillator, which with the applied input voltage it is able to determine the instantaneous oscillation frequency [28]. The VCO wants to compensate the frequency offset and ideally, the resulting frequency at the output of the mixer would be 0, which will only happen at the end, once the PLL is finally locked. This resulting frequency will then be low-pass filtered so that it is only kept the difference between frequencies of the two mixed signals.

For the modulation recovery process, one of the split outputs of the VCO is mixed with the original signal, which contains the carrier frequency and the modulation. Since the output of the VCO only contains the carrier frequency, the result of the mixing step is the difference between the both just mentioned, i.e. the modulation itself at baseband.

The devices and mini-circuits used for the experiment are:

- Frequency synthesizer: ROHDE&SCHWARZ SMA 100 A. SIGNAL GENERATOR FROM 9 kHz to 3 GHz.
  It represents the satellite's LO (LO(1)).
- Function generator: TEKTRONIX AFG 3102 Dual Channel Arbitrary/Function Generator. 1 GS/s.
  It generates a square function that represents the modulated data navigation send from the satellite at a 1 ms periodic rate.
- Frequency synthesizer: ROHDE&SCHWARZ SMA 100 A. SIGNAL GENERATOR FROM 9 kHz to 6 GHz.
  It is used to perform the VCO, since it is provided with an external FM input that can be tuned with a voltage and will output a frequency.
- 4 mixers: Mini–Circuits ZX05-43MH-S+ frequency range 824-4200 MHz.
- 3 splitters: Mini-Circuits ZX10-2-42-S+ frequency range 1,90-4,20 GHz.
- Amplifier: ZX60-272LN-S+ frequency range 2300-2700 MHz, 14 dB gain.
- Band-pass filter: For a 2,4 GHz working frequency.
- Frequency divider by two: HITTITE 104627-3.
- Oscilloscope: LE CROY WAVERUNNER. LT374M 500 MHz 4GS/s DS0.
- Spectrum analyser: IFR 2399 9 kHz to 2.9 GHz.

5.2 Software experiment

Instead of using all the devices mentioned on the experiment above, the only ones kept to execute the software experiment are the function generator, which now generates a square function of 100 kHz, to have the modulation peaks closer so that the experiment in a 2 MHz bandwidth is more visual and the Nyquist criteria is met, and one frequency synthesizer.

The frequency synthesizer is in charge of simulating the satellite carrier by generating a 1,21 GHz carrier frequency instead. Right after the mixing step between both of them, the incoming signal is transferred through a wire representing the two antennas from the satellite to the receiver. The E4000 dongle DVB-T is plugged into the other extreme of this wire and then, plugged to the computer. The dongle is set at 2 MS/s originating

then a 2 MHz bandwidth, which in this case since the frequency modulation is 100 kHz is not necessary to have a bandwidth that wide. It is decimated so that it acquires a value around 200 kHz (enough to not lose information) and 1 MHz.

The dongle itself performs a first amplifying step of 42 dB [9] from what it gets from the wire and then it mixes it with the carrier frequency from its LO, in an unchanged and a 90º phase-shifted version, creating this way the I&Q values. The local copy of the carrier frequency generated by the dongle's LO is also set at 1,21 GHz but it can present a frequency offset because of the reasons exposed on chapter 4.2.

To start with, before implementing the rest of the circuit functions using GNURadio, the steps that need to be followed, which at some point differ from the ones pursued on the analog experiment, are thought and exposed in figure 5.5.

In order to be sure that, once the performance of the mixing step between the incoming signal and the locally generated carrier frequency has been undertaken by the dongle, the range of the frequency offset is covered, it has been cautiously estimated to have a frequency lowered to 100 kHz for the initial computations.

After having had the maximum estimated frequency offset value, which will show a downward trend when the PLL tracks its phase, it can be said that there is no need of implementing a divide by two block after the squaring function in this experiment. This assumption can be realized because, in contradiction to the analog experiment, the frequency value after having squared the signal is not relevant any more, since it represents a frequency offset and not the value of the original carrier frequency.

In the analog experiment, having the carrier generated by the VCO was the only way to recover it when it becomes locked. While, the aim in the software experiment is to remove the remaining offset between the incoming carrier and the one generated by the dongle's LO. In other words, the objective is to reach $\Delta f=0$, the doubling does not matter as the condition is also met if $2\Delta f\approx 0$ as long as $2\Delta f$ remains within ± Nyquist frequency.

A schematic exposing the related explanation is shown in figure 5.5.

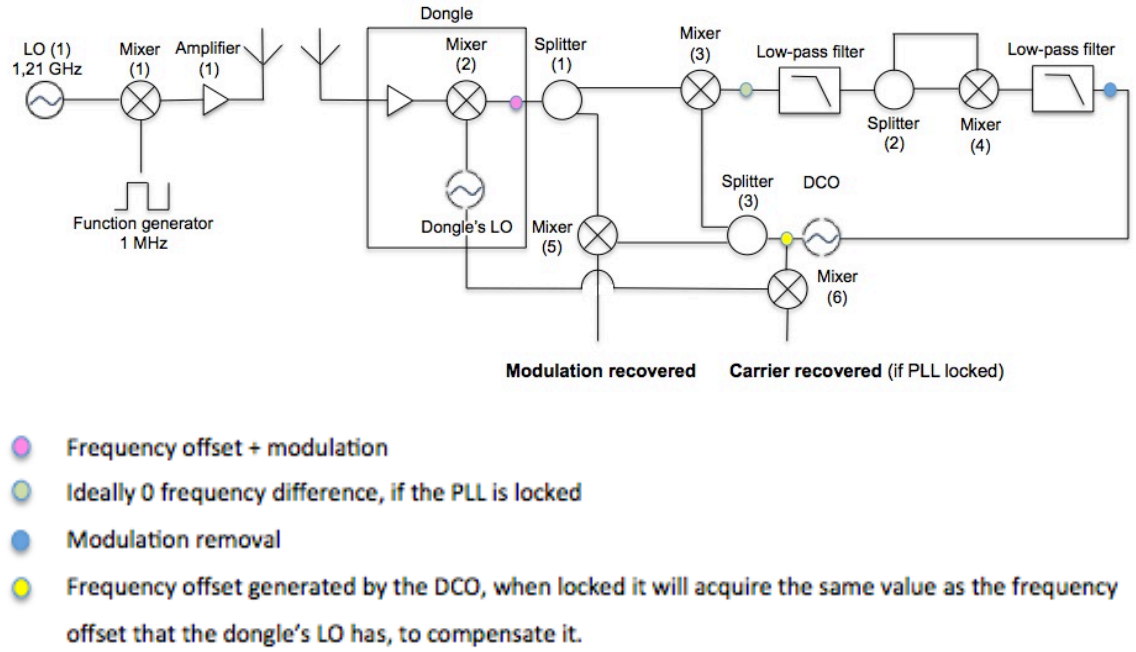

Figure 5.5: Single carrier signal processing in software with the DVB-T dongle

Once the ideas are clear in mind, it has been proceeded to achieve simulated results using GNURadio by replacing the blocks on the right side of the dongle in figure 5.5 for the following schematic.

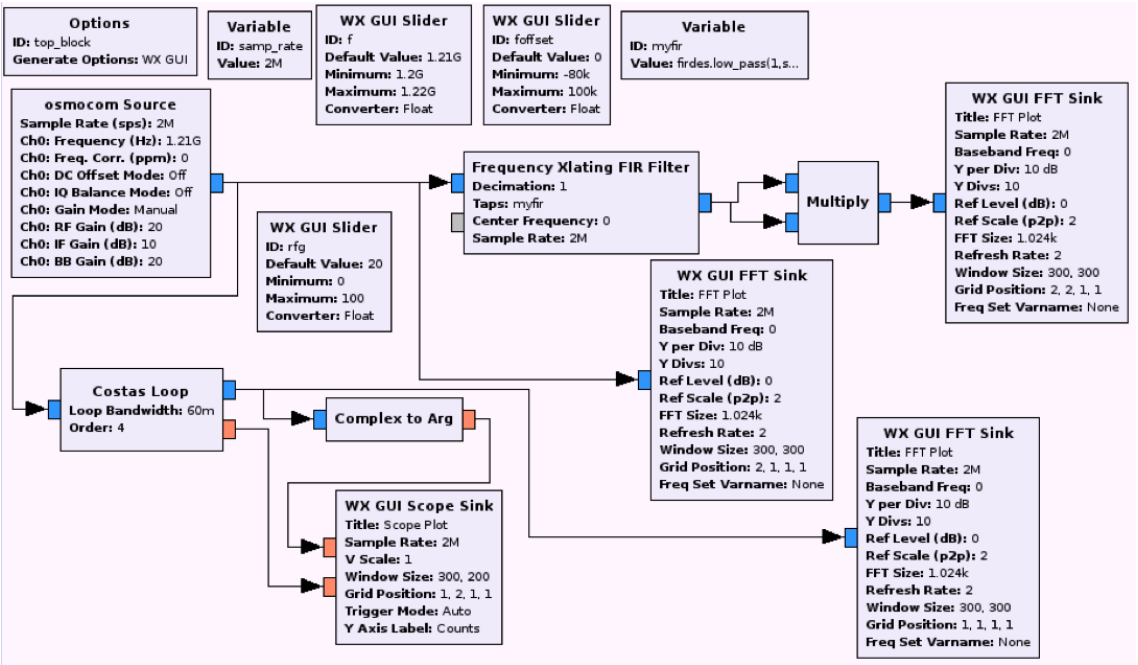

Figure 5.6: GNURadio acquisition of a simulated signal

The schematic is not as simple as described before the implementation in GNURadio, the reason being that some previous blocks before the discovery of the existence of the *Costas Loop* block have been maintained. This way we are able to plot some processing steps that are happening inside the Costas Loop block that otherwise they are not visible. However, the initial purpose was to manually implement Costas Loop following the scheme implemented in hardware but adapted in figure 5.5. This was successfully achieved until closing the loop by attempting to control the Digitally Controlled Oscillator (DCO) with the output of the carrier recovery, since closing the loop outside a processing block is not allowed in GNURadio-companion. The Costas Loop block is a type of PLL specially used in BPSK modulated signals, since it is insensitive to 180º rotations. The reason why an insensitive discriminator is needed is because what is performed in the Costas Loop is a mixing step between the incoming signal, which contains a carrier frequency and navigation data encoded by a BPSK modulation, and a frequency generated by a DCO, which does not have a modulation at all. So to compare them, the modulation from the first one needs to be removed.

The output of the *osmocom Source*, where the sample rate is set, is directly connected to the *Costas Loop* block, which exercises the function of the filter and the DCO.

In order to compare the case when the PLL is locked and when it is not, 4 plots have been done for both different situations. On the one hand, it has been provoked the unlocked case by setting the frequency offset away from the 0 value so that 2Δf lies outside the Nyquist frequency range. And on the other hand, it has been done the other way around, it has been set the frequency offset right close to the 0 value.

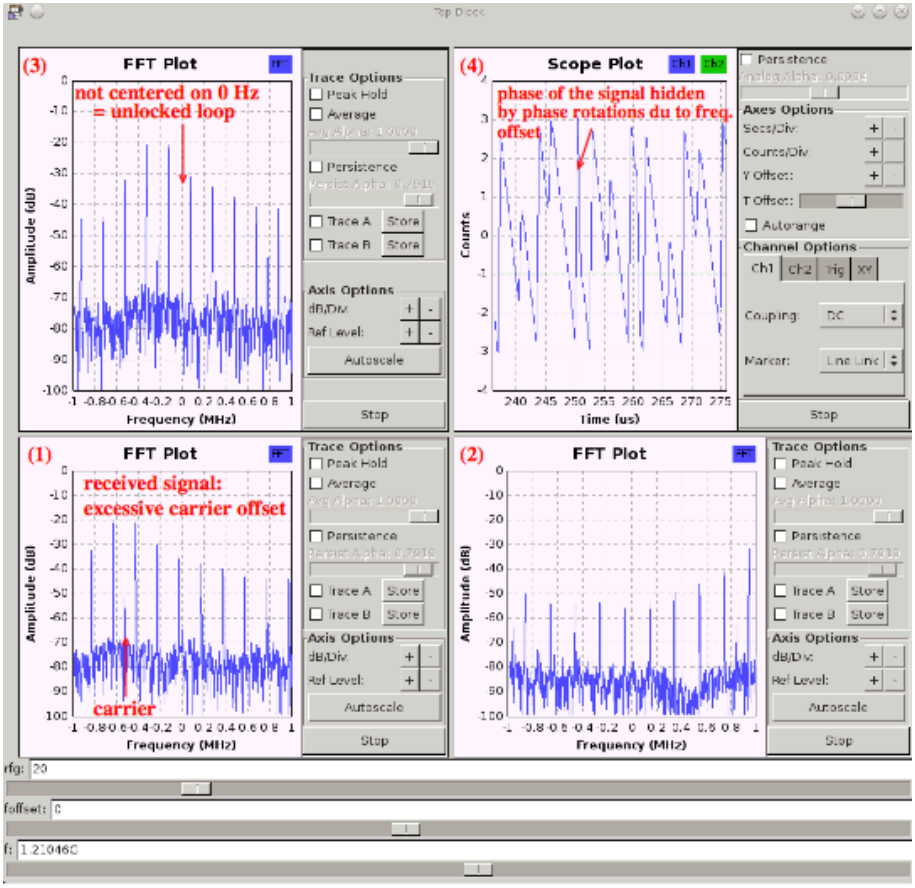

Figure 5.7: Unlocked case

- (1): On the bottom left, there is the peak from the dongle's mixer output, which has already been sampled by the A/D converter. It represents the frequency offset that has been set in this case to 0.5 MHz. This graph shows the signal before having been processed.
- (2): On the bottom right it has been plotted, in the frequency domain, the output of the *Multiply* block. The original signal has gone through the *Frequency Xlating FIR Filter* block, which actually contains a DCO and a low-pass filter, and through the *Multiply* block, which is squaring the signal. Thanks to the squaring, the modulation peaks have been lowered. Though, as there is a wide frequency offset, its peak, which now represents the carrier frequency that has been moved to a few kHz, gets out of the bandwidth when squaring it and it is not distinguishable any more in this plot. This is the most obvious reason why it is the unlocked case, because the frequency offset has not been compensated since it is out of the ADC bandwidth.

  When working in digital, the filter is not the limiting factor anymore but the ADC bandwidth.

- (3): On the top left, the output of *Costas Loop* in the frequency domain is plotted. Ideally, what the mixing step in the Costas Loop aims to reach in the locked case is a carrier removal, leaving only in its output the modulation peaks. As mentioned, as it is the unlocked case, the carrier frequency is not centered at 0 and its peak is smaller than the modulation peaks in 30 dB.
  Figure 5.8 shows the exact point of the Costas Loop where the plot has been realized.

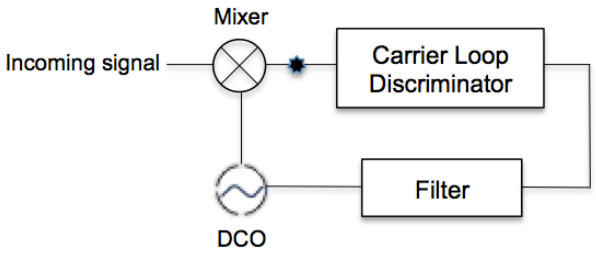

Figure 5.8: Costas Loop's schematic

- (4): Finally, on the top right, there is the plot generated by *Scope Plot,* in which appear two lines that correspond to the channels 1 (phase Θ) and 2 (frequency output of the mixing step in Costas Loop, i.e. the actual offset between the frequency from the LO of the satellite and the one of the dongle (f-f').

  To explain things clearer and see the contribution of each one of the mentioned lines, the equation of the resulting signal from the Costas Loop is written:

  $S = A \cdot \sin \Theta$; where $\Theta = 2\pi(f-f')t + \phi$.

  If the contribution of the offset f-f' is very high, the line representing the phase Θ from channel 1 will be continuously oscillating, leaving the phase information masked. Only when this offset is compensated for, will the phase in channel 1 stabilize and the slope of f-f' in channel 2 will become flat.

Then, the 4 plots for the locked case have been executed at the same exact points as they have for the unlocked PLL case. The difference is that now the set frequency offset is inside the range in which the PLL can become locked.

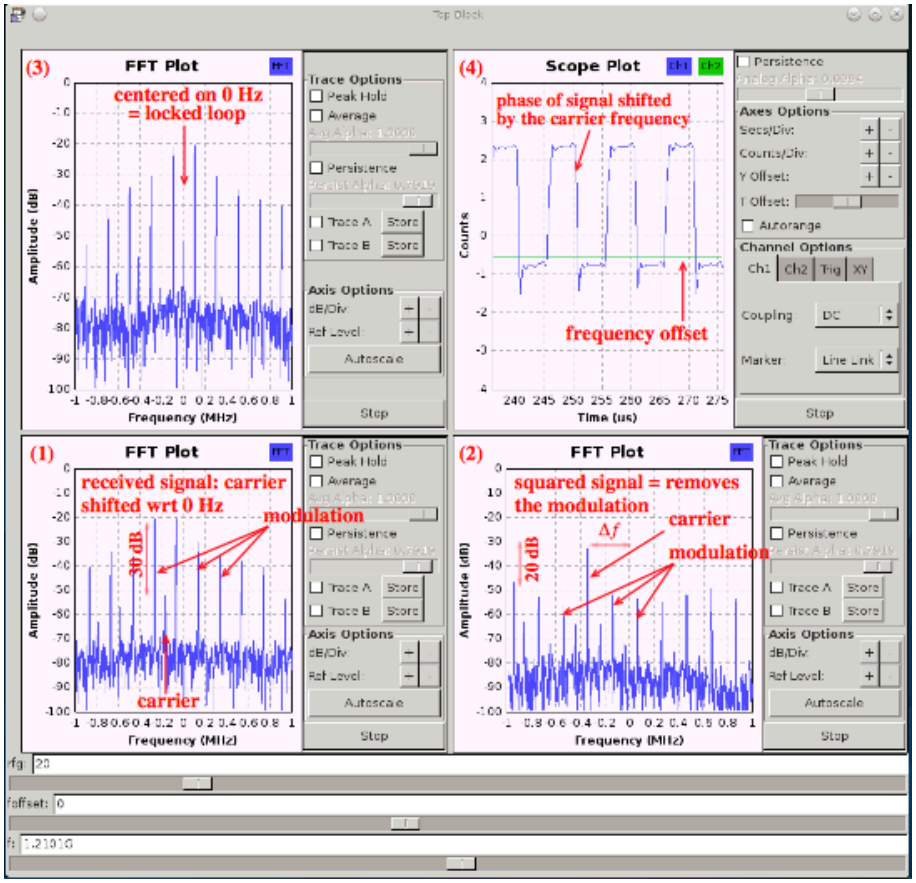

Figure 5.9: Locked case

- (1): To start the same way as in the unlocked case, on the bottom left, it appears the actual frequency offset, which has been set at 0,2 MHz. It is observable that the carrier frequency peak is 30 dB lower than the modulation peaks.
- (2): On the bottom right, after the squaring function, the modulation peaks are 20 dB less powerful than the carrier frequency peak, which has not been centered to 0 yet, since it is only at the output of the multiply block where the modulation is removed. In contradiction to the same performance for the unlocked case, on this plot, as much as the carrier frequency has been squared, the double of its frequency remains inside the bandwidth because it has the value of the frequency offset, which is now smaller.

  The PLL can be manually tuned by using the *foffset* parameter, which appears at the bottom of the plots. It is at this step where the foffset parameter can be used, since in plot (2) the Costas Loop has not been implemented yet,

otherwise it is tuned itself. We experimentally observed that the system becomes locked when the line from the channel 2 (the command for the Costas Loop discriminator) is between the ± 1 value on the Y axis and the foffset parameter reaches -69,2 kHz.

- (3): On the top left, now that the PLL is locked, the carrier frequency peak is well centered at the 0 value, with the modulation peaks at both sides.
- (4): Finally, on the top right plot, there is no slope anymore and the phase information has been recovered.

6. Acquisition

There are three main steps that a receiver executes when operating with GPS signals.

1) Acquisition
2) Code and carrier tracking
3) Data processing for positioning

The acquisition is the first state in which two properties of the signal, the frequency and the code phase, are determined through a search process. The code phase is the time alignment of the PRN code in the current block of data, which in other words can be understood as it refers to the initial position of the PRN code in an overall of 1023 possibilities. It is in the acquisition part where the visible satellites are detected. The frequency is affected by the Doppler shift, which establishes from the beginning a frequency search range, and the code phase denotes where in the actual data block the C/A code starts [2, 7].

There exist three standard methods that can be applied to software receivers.

6.1 Serial Search Acquisition

It is the simplest and most frequently used method. The signal must correctly match in the two dimensions, with the carrier and the code. This is the reason why it is required the replication of both.

Firstly, the incoming signal from a given satellite is multiplied by the properly aligned in time code replica that corresponds to the same satellite, which implies having the correct code phase and the nearly removal of signals from other satellites.

Secondly, after the multiplication between codes, the signal must be multiplied by a locally generated carrier frequency that is close to the signal carrier frequency. After the multiplication between frequencies the inphase signal I is generated, the same way as the quadrature signal Q is generated after the multiplication, in this case, with a 90º shifted version of the local copy of the carrier. The I and Q signals are then integrated over 1 ms, which means that all the points corresponding to the length of the processed data are summed. This operation will ideally locate the signal power in I since the C/A code is only modulated onto that. However, this will only occur when the signal has been demodulated but for the moment the phase of the received signal is

still unknown. Concluding, at this step both I and Q need to be investigated. Afterwards, I and Q have to respectively be squared in order to obtain the power level and finally added [2, 7]. These last two operations are equivalent to having the squared modulus of a cross-correlation result. The squaring operation allows us to properly observe if a maximum correlation is reached, since if the result of the correlation is a -1 it will become a 1 and if it is already a 1 it will remain the same. Figure 6.1 shows the schematic that summarizes the just described description.

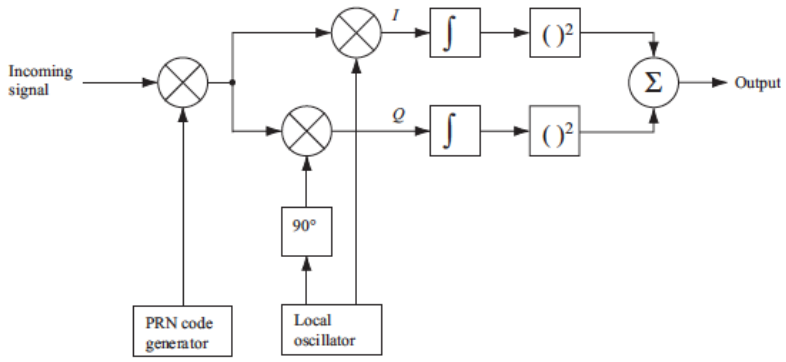

Figure 6.1: Serial search acquisition [7]

To identify visible satellites, the frequency must be swept over all possible carrier frequencies inside the range of 90 kHz (the frequency leftover after having had the incoming signal multiplied by the locally generated in the dongle) ± 10 kHz, since it is the maximum expected Doppler shift value due to the satellite motion (5 kHz) added to the maximum expected Doppler shift value due to the receiver motion (5 kHz). However, our receiver is considered static for this particular study, so in contradiction to the literature it should be considered all possible frequencies of 90 kHz ± 5 kHz. The sweeping is done in steps of 500 Hz, which is the frequency offset that guarantees the maximum correlation peak to be still visual, and each C/A code over all 1023 different code phases. This results in 41 different frequencies multiplied by the 1023 code phases [2, 29].

$$\left.\begin{array}{l} 2 \cdot \frac{10000}{500} + 1 = 41 \text{ frequencies} \\ 1023 \text{ code phases} \end{array}\right\} \quad 41 \cdot 1023 = 41943 \text{ combinations}$$

6.2 Parallel Frequency Space Search Acquisition

Fortunately, this method parallelizes, or in other words divides, the search onto one

parameter, since it is a much more time-consuming procedure when the two parameters, frequency and code phase, are desired to be sequentially searched. Using this method the necessity of searching through all the 41 frequencies has been eliminated. Its PRN code replica is the only parameter that multiplies the original signal, resulting in 1023 of search combinations that need to be examined to find the initial position of the PRN code.

Unlike the Serial Search Acquisition process, the Parallel Frequency Space Search Acquisition operates in the frequency domain thanks to the implementation of the Discrete Fourier Transform (DFT) or its faster method the Fast Fourier Transform (FFT) [6]. The DFT is the spectral domain of time domain cross-correlation and decomposes a sequence of values (a signal that is a function of time) into components of different frequencies that make it up [30].

The definition of the Fourier transform for a continuous sequence is the following.

$$X(\xi)=\int_{-\infty}^{\infty} x(t)e^{-j2\pi t\xi}\, dt$$

Where ξ is the frequency in Hertz and t is the time in seconds [31].

And in the case of a finite sequence of length N it follows the next discretized formula.

$$X(\xi) = \sum_{n=0}^{N-1} x(n)^{-j2\pi t\xi n/N}$$ [7].

For this specific case of study, it has been adopted the FFT in all situations. The FFT is an algorithm to compute the DFT and the reason why it grants a faster execution when realizing it for N points is that it requires $N \cdot \log_2(N)$ operations instead of N^2 as the DFT does [30]. A positive aspect of implementing the FFT is that it analyses all possible combinations itself, since it makes best use of symmetry conditions, which cannot be used in the time domain theory knowledge. The FFT is applied to the continuous wave, which is the result of the multiplication between the original signal and the PRN replica code with its 1023 different code phase.

Then, the squared modulus of the outcome of the FFT is performed, since all the operations have been performed with complex values. The result, if the replica code is well aligned with the incoming signal; is a frequency peak located at the frequency offset plus the frequency offset [7]. Figure 6.2 represents the operations performed in the Parallel Frequency Space Acquisition method.

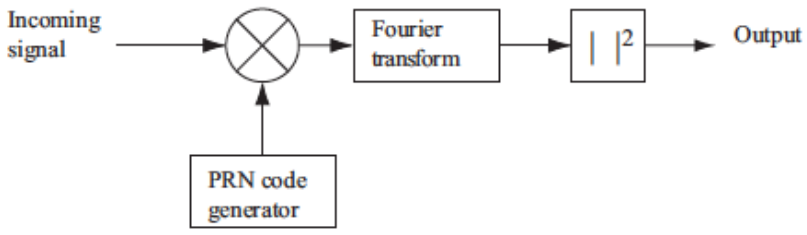

Figure 6.2: Parallel Frequency Space Search Acquisition [7]

6.3 Parallel Code Phase Search Acquisition

In contradiction to the Parallel Frequency Space Search Acquisition method, what this third method pretends is to search only into the 41 different frequency combinations [7]. It is achieved through the cross-correlation in the frequency domain between the I&Q values of the original signal and a non-shifted PRN code.

Figure 6.3 shows the implemented operations in the Parallel Code Phase Search Acquisition and more detailed, the decomposition of the mentioned cross-correlation.

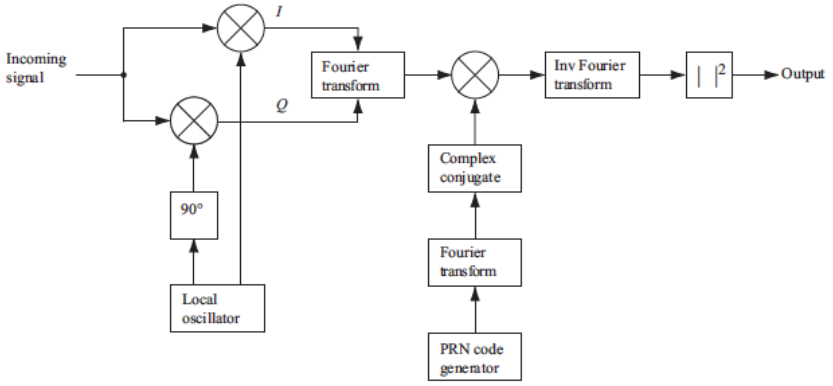

Figure 6.3: Parallel Code Phase Search Acquisition [7]

On one side, the incoming signal is multiplied, on the one hand, by an unchanged local copy of the carrier frequency and, on the other hand, by a 90º phase-shifted local copy of the carrier so that the I&Q values are created, which only contain the modulated PRN code of the signal. Then, the I&Q complex values are transformed into the frequency domain.

On the other side, a signal containing the PRN replica code is also converted into the frequency domain through the Fourier Transform and then it is complex conjugated.

Afterwards, a multiplication involving both sides is realized. The overall of the executed operations is equivalent to performing a Fourier Transform of the cross-correlation

between the I&Q and the PRN code generator. This last definition corresponds to the introductory explanation of this acquisition method.

Once the circular cross-correlation has been accomplished, the square of its modulus is performed, which value will create a peak representing the correlation between the incoming signal and the PRN code, being the index of this peak the PRN code phase of the signal [7].

The mathematical demonstration for the equivalence between operations is the following.

6.3.1 Mathematical relation between convolution and correlation

It gives the overlap area between the two functions as a function of the amount that one of the original functions is translated [32]. One of the principle characteristics of the convolution is the sign of the variable that defines each sequences, which is positive in one and negative in the other, implying the matching between both is reached by moving along each sequence in contrary directions. This property changes in respect to the correlation operation, which shares the sign of the variable for both sequences.

The convolution between two sequences corresponds to the following formula.

$$(u*v)(s) = \int u(t) \cdot v(s-t)dt$$

The Fourier Transform of the above convolution is written following the next formula.

$$\int (u * v)(s) \cdot e^{sr} ds = \int (\int u(t) \cdot v(s-t)dt) e^{sr} ds$$

with a variable transform as [y=s-t] then, [s=t+y]

Developing the right side of the equivalence, what is obtained is the following result.

$$\int (\int u(t) \cdot v(y)dt) e^{(y+t)r} dy = \int (\int u(t) \cdot v(y)dt) e^{yr}\, e^{tr} dy$$

$$\int (\int u(t) \cdot v(y)dt) e^{yr}\, e^{tr} dy = (\underbrace{\int u(t) e^{tr} dt}_{FT(u)}) \cdot (\underbrace{\int v(y) e^{yr} dy}_{FT(v)})$$

$$FT(u*v) = FT(u) \cdot FT(v)$$

This same equivalence is also fulfilled with the correlation operation but introducing a change. It is necessary to do the complex conjugate in one of the sequences once applying the Fourier Transform, in order to invert one of the variable signs so that both sequences have the same direction. This results into the following formula.

$$FT(u \bigstar v) = (\int u(t) e^{tr} dt) \cdot (\int v(y) e^{-yr} dy)$$

$$FT(u \bigstar v) = FT(u) \cdot \text{complex conjugate } (FT(v)) \text{ [33].}$$

7. Parallel Code Phase Search Acquisition experiments

Actual GPS satellites have their own ranging codes, each one different to the other, so that they are able to broadcast their characteristic data navigation at the same carrier frequency. This is the so-called CDMA method explained in the introduction.

If a PRN's code generator is plugged instead of the function generator used until this moment, the reality of satellites will be more accurately simulated but a phase shift removal needs to be executed in the cross-correlation. If the cross-correlation between the incoming signal and its replica code is not performed in advanced, the *Costas Loop* cannot be locked as it receives the multiple carrier frequencies from all the satellites each one with its own frequency offset. Consequently, the system does not know which offset needs to be adjusted because it exists one for each satellite. This is the reason why a process with the name of Parallel Code Phase Search needs to be introduced to recover the carrier frequency, so that the receiver focuses on the visible satellite or satellites.

When a maximum correlation is reached, a perfect matching between one of the replica codes and the code from the visible satellite is obtained, meaning finally that the receiver is able to focus only on the visible satellites from the overall of the 24 that are sending information in 32 different PRN codes.

For the following explanations, C_s will correspond to the PRN code that sends a given satellite, while C_r refers to the PRN code replica generated by the receiver.

$C_s \star C_r = 0$ Means that C_r does not correspond to the visible satellite.

$|C_s \star C_r| > 0$ Means that C_r matches with the C_s code that the visible satellite is sending.

7.1 7 bit PRN code

Following the same methodology as in the experiments previously performed, the objective when introducing the PRN code concept for the first time is to start using available hardware in the department thanks to usage of ancient's colleague's work before trying to achieve software results.

7.1.1 7 bit PRN code generator device

A PRN code generator device designed by Christophe Fluhr, an electronic's master's student working in the Time and Frequency Department of ENSMM, is set on the installation as if it was the PRN's code that is found on the incoming signal sent from the satellite, i.e. the before mentioned C_s.

It is a 7 bits shift register for which 2 primitive polynomials have been chosen in order to generate 2 different codes (C_1 and C_2) for C_s. The two codes will represent as if there were only 2 satellites in orbit, which have been coded using either one or the other of both polynomials below:

(1): $x^7 + x^4 + 1$

(2): $x^7 + x^6 + 1$

The sequence has $2^7 - 1 = 127$ bits length. For the good performance of the PRN code generator device this needs to be connected to a *Waveform Generator (*in this case a 50 MS/s Waveform Generator VW5061), which works as what would be its internal clock. The clock is set to 111,75240 KHz instead of 100 KHz because of its known offset. Its power supplied is 3,3 V of amplitude of a square wave. The output of the PRN code generator needs an attenuator of 1 μF so that the mean of the square wave is removed, i.e. it reaches the removal of DC component.

An attenuator is the opposite of an amplifier, since it provides loss or gain less than 1, i.e. it reduces the power of a signal without appreciably distorting its waveform [34].

A schematic representing the linear feedback shift register (LFSR) that generates; either one or the other of these 7-bit PRN sequences, is shown in figure 7.1. A deep explanation regarding how their generation is performed is realized after figure 7.3 because it makes it simpler.

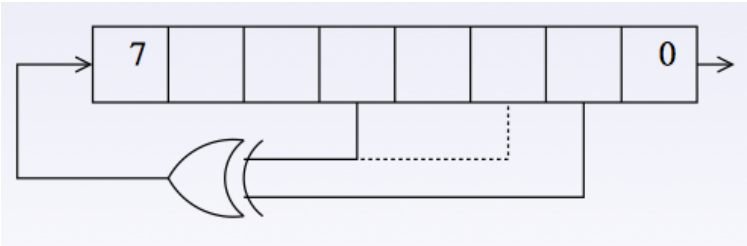

Figure 7.1: Schematic of the generation of the 7 bits PRN codes

7.1.2 7 bits PRN code implemented on an Octave script

In order to be able to execute the correlation that allows us to find out the visible satellites, a C_r code is generated in the receiver thanks to an octave script called *lfsr.m*. C_r is created so that it has the same characteristics as C_s, having the possibility to adopt the same two different sequence values.

But first, before implementing the PRN code generated in octave in feature experiments, a basic test is executed. It is performed in order to prove the clear difference between when a code is cross-correlated with itself and when it is cross-correlated with a code that does not have its same nature. On the one hand, the graph shows with the red line that a maximum cross-correlation is reached when auto-correlating a code. While on the other hand, the blue line shows the null cross-correlation between C_1 and C_2, as their shape do not fit together when they are superposed.

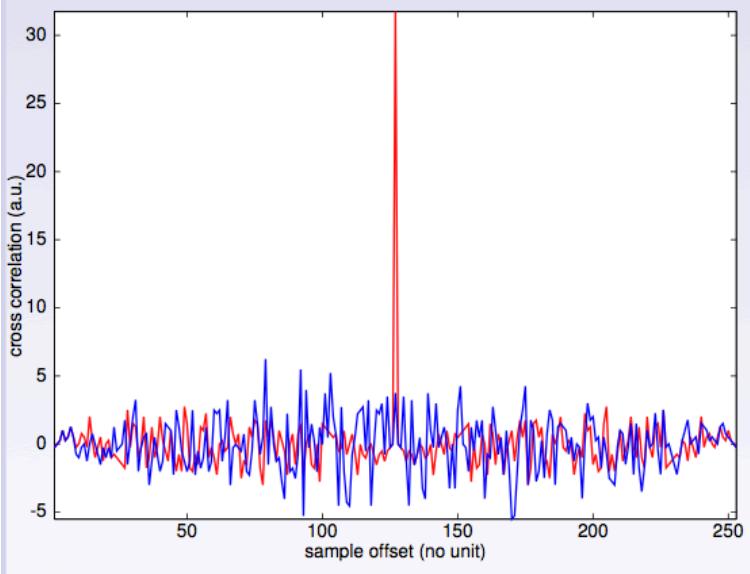

Figure 7.2: Auto-correlation (red) and no-correlation (blue)

The brief Octave script in charge of generating the PRN code replicas in the receiver is the following.

```
1 // http://en.wikipedia.org/wiki/Linear_feedback_shift_register
2 # include <stdint.h>
3 int main(void)
4 {uint8_t start_state = 0x40;
5  uint8_t lfsr = start_state;
6  unsigned bit,period = 0;
7  do
8  {// taps: 16 14 13 11; feedback polynomial: x^16+x^14+x^13+x^11+1
9   // bit=((lfsr>>0)^(lfsr>>1))&1; // X^7+X^6+1
10     bit=((lfsr>>0)^(lfsr>>3))&1; // X^7+X^4+1
11     printf("%d %d\n",lfsr,bit);
12     lfsr =  (lfsr >> 1) | (bit << 6);
13     ++period;
14    } while (lfsr != start_state);
15    return 0;
16 }
```

Figure 7.3: 'lfsr.m' octave script for generating PRN replica codes

A quick summary of how logic gates are written in software language is done, so that a correct following of the script is performed:

- ^ is equivalent to XOR.
- & is equivalent to AND.
- | is equivalent to OR.

Moreover, it also appears this symbol:

- y >> x, which indicates that the sequence (y) must move each of its values (x) positions to the right.
- y << x, which has the same meaning as the statement just above but the movement now needs to be done to the left side.

The explanation of the most important lines of the script is the related below:

- In **line 4** it has been stated the first value that the sequence acquires, which on the following line becomes the value of lfsr. It is expressed in hexadecimal 0x40, which in binary represents the 0100 0000. This first chosen value could have been any other despite the null value, which is prohibited.
- When the 2 slash bars (//) appear, means that the script is not executing the line comment.
- In **line 10** is the first time that the bit variable appears. It is defined to be the last bit of the sequence, which is calculated using the XOR function between the chosen values of the working polynomial. In this case, the working polynomial is (1): $x^7 + x^4 + 1$, so the XOR will act between the value of the seventh and forth

degree. Consequently, the AND logic gate is implemented to obtain the last bit of this sequence. It operates between the current result of the operated sequence and the value one (0000 0001), as the multiplication (operation of the AND gate) by one does not alter the result of the last bit and the seven zeros in front (**0000 000**1) will cancel the remaining values of the lfsr sequence.

This last bit is the output of the PRN code generator device and it is the value that characterizes the next state of lfsr.

- In **line 12** the value of lfsr is updated by employing an OR logic gate between the lfsr sequence shifted one position to the right and the bit value shifted 6 to the left. This way the new lfsr has the previous lfsr sequence value shifted one to the right and as first value, the bit.

This loop will continue working until the lfsr sequence acquires again its initial value, in this case 0100 0000.

7.1.3 Acquisition of a simulated signal

Once we have the PRN codes representing the ones that satellites send and the ones stored in the receivers, it is time to proceed with the Parallel Code Phase Search method that will provide us with the acquisition of a signal, for the moment, simulated. To do so, an Octave script has been created for a specific frequency offset value (Deltaf) to compute the cross-correlation between the incoming signal and the replica codes.

```
1  load 741.txt  % load PRN1
2  load 761.txt  % load PRN2
3  X741=X741(:,2)-mean(X741(:,2));
4  X761=X761(:,2)-mean(X761(:,2));
5
6  x=read_complex_binary('g.bin');
7  time=[0:1/2e6:length(x)/2e6]';
8  time=time(1:end-1);
9
10 Deltaf=-1600;  % carrier offset
11 mysine=exp(j*2*pi*Deltaf*time);
12 xx=x.*mysine;
13 xx=xx(1:20:end);  % 2MHz/100kHz
14
15 plot(abs(xcorr(X741,xx)),'r');xlim([0 N]);
16 hold on
17 plot(abs(xcorr(X761,xx)),'b');xlim([0 N]);
```

Figure 7.4: Octave script for the acquisition on a 7-bit PRN code simulated signal

The explanation regarding the octave script to reach the acquisition of a simulated signal is the following.

Firstly, in lines 1 and 2, two files named '761.txt' and '741.txt' are loaded containing the sequences generated by the lfsr.m file using both, the first (in '741.txt') and the second polynomial (in '761.txt').

Secondly, in lines 3 and 4, its mean is removed so that when there is no cross-correlation between codes, a flat curve is the resultant plot. Otherwise, it would result as a triangle, making difficult to see if any maximum cross-correlation peak was obtained.

In line 7, the time variable is created and it goes from 0 to length(x)/2·10^6 in steps of 0,5 μs, which shows the recorded number of seconds ($\frac{length(x)=total\ number\ of\ samples}{2{\cdot}10^6\ samples/s}$) every time one sample is executed, since the dongle samples at 2 MS/s.

Then, in line 6, a file named 'g.bin', which has been created previous to the execution of this Octave script, is called. The 'g.bin' file contains a 7 bits PRN code simulated signal generated thanks to the mixing step between the output of the PRN code generator device with the ROHDE&SCHWARZ frequency synthesizer in charge of originating the carrier frequency at 1,21 GHz. Right after the mixing step it comes the dongle, which will create and sample the I&Q values, since it mixes the input with its LO that has been set to the same carrier frequency value.

The 'g.bin' file is read using the 'read_complex_binary' function, which is a function from GNURadio. What this function does is to interpret alternating real and imaginary part of complex values represented in floating point format.

In line 10, in order to get rid of the carrier frequency, which has been moved to the frequency offset value between the synthesizer and the dongle, a so-called parameter 'Deltaf' is defined to characterize this offset. 'Deltaf' is the frequency parameter for a new signal named mysine created in Octave in charge of the compensation of the frequency offset.

When working with both, the signal from the theoretical receiver ('g.bin'), which now is in complex values, and the signal characterized by the frequency offset, this last one needs to also be in complex values so that the mixing is done properly. This is why the exponential is done, so that the new created signal is written also as a complex.

In line 12, the multiplication between the incoming signal and mysine is performed, remaining then in xx the PRN code only. Afterwards in the following line, 1 every 20 samples are taken from the resultant multiplication vector (xx) so that, before performing the cross-correlation with the code replica generated by lfsr.m, they posses

the same sampling rate, since the dongle produces samples every 0,5 μs and the lfsr.m every 10 μs as it has the same properties of the PRN code generator device.

The last operation executed in the last lines is to cross-correlate xx with both 741.txt and 761.txt so that it results in maximum cross-correlation with either one or the other, depending on which its replica is.

After performing this last experiment of the acquisition of a simulated signal, two questions emerged:

1) Which is the maximum frequency offset that can present the system and still allow the maximum cross-correlation peaks to come out?
2) Will the experiment work the same way when instead of working with a 7 bit PRN code a 10 bit PRN code is implemented as it is in a real GPS? Will it increase or decrease the robustness?

The solutions proposed are subsequently exposed.

7.1.4 Acquisition of a simulated signal varying the frequency offset

To try to answer the first question, it has been taken the case in which the incoming signal gives a maximum cross-correlation when cross-correlating it with the C_2 replica code with its frequency offset compensated. It has been followed the same strategy as before, although this time, the signal in charge of compensating the frequency offset has a varying 'Deltaf' parameter, which in each of the three plots on figure 7.5 adopt a different value.

(1.1)

(1.2)

(1.3)

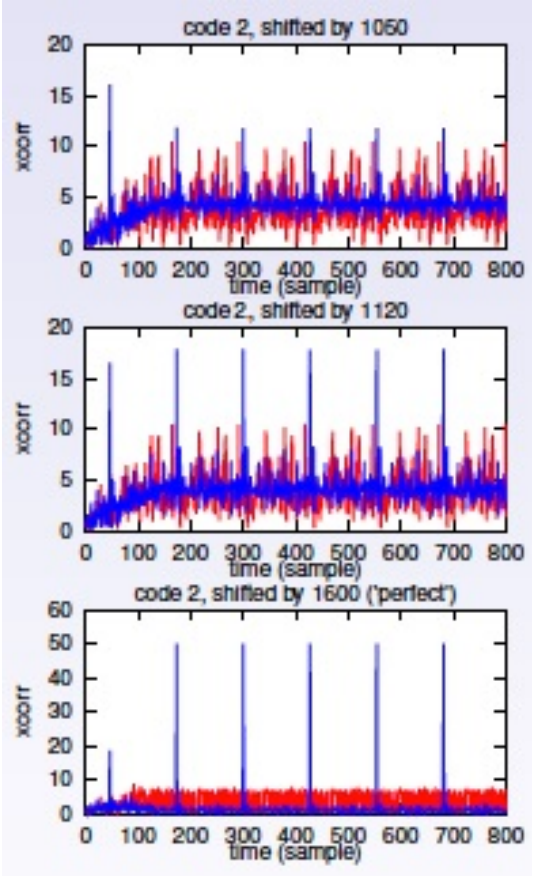

Figure 7.5: Incoming signal with different frequency offsets cross-correlated with C_2

- (1.1): At first, on the top plot, the frequency synthesizer has been tuned to 1,21 GHz + 1050 Hz. The cross-correlation result with C_2 (blue line) does not differ much with the cross-correlation result with C_1 (red line), which we know is minimal. The reason is that the frequency offset has not still been compensated and the carrier frequencies generated by the synthesizer and the dongle's LO do not match yet.
- (1.2): On the second plot, the frequency generated by the synthesizer is now 1,21 GHz + 1120 Hz, and the result is better, which means that both carriers are getting closer. The blue line comes up from the red one that is flatter but still some improvement can be done.
- (1.3): Finally, on the last plot, the frequency synthesizer is set to 1,21 GHz + 1600 Hz, showing a bigger difference between the blue and the red peaks from the other plots.

After the performance of this experiment it can be affirmed that when the carrier frequency of the synthesizer has been moved 1600 Hz from its original value, the frequency offset is removed and the result of the maximum cross-correlation is perfectly observable.

7.2 Real 10-bit GPS PRN code

With the finality of answering the second question, it is necessary to create 10-bit PRN codes, which are the real codes that GPS satellites send, the so-called C/A codes described on chapter 2. But first, it needs to be known how they are generated.

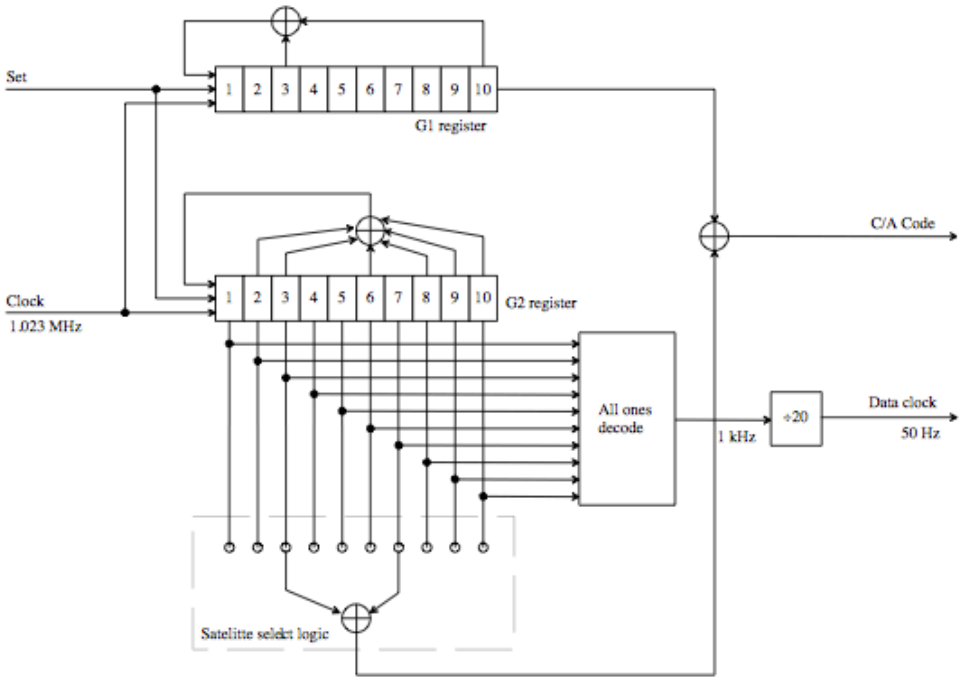

Figure 7.6: 10 bit GPS PRN code architecture generation [29]

Two 10-bit LFSR, which are shaped by the two following polynomials $G_1(x) = 1 + x^3 + x^{10}$ and $G_2(x) = 1 + x^2 + x^3 + x^6 + x^8 + x^9 + x^{10}$, generate pseudo random codes with a length of $2^{10} - 1 = 1023$ bits.

These PRN codes are generated in a similar way as the 7-bit PRN codes were but with some changes. Since they are two LFSR that work together, it exists the possibility of generating a bigger number of pseudo random codes. The executed operation to create C/A code is to exclusive-or the output of the G_1 LFSR and a delayed version of the output from the G_2. To obtain the delayed effect for the G_2 LFSR, two selected stages from the G_2 LFSR need to be exclusive-or [29]. The selection of states for the modulo-2 adder is called the phase selection.

The G1 register always has a feedback configuration with its polynomial, the same way that the G_2 register has with its own one.

Table 1 in Appendix A shows the combination of the phase selections for each C/A

code and the first 10 chips of each code in octal representation [7].

7.2.1 Robustness of a 10-bits PRN code with a varying frequency offset

The purpose of the following test is to answer the second question posed at the end of chapter 7.1.3. The objective is to prove the robustness of the GPS PRN code, which as its definition suggests is the ability of a system to resist in front of changes without modifying its initial stable configuration [35].

Consequently, an octave script has been created to test the result of the autocorrelation with one of the 37 codes with itself but with different chipping rate values in each situation, since it has been thought it contributes the same way as if a frequency offset in the carrier was introduced. Figure 7.7 shows the mentioned script.

```
more off; close all; clear all
set (0, "defaultaxesfontname", "Helvetica")

a=cacode([1:31,1);
a=a';
a=a-mean(a);
plot(xcorr(a(:,1),a(:,1)),'r')
hold on
plot(xcorr(a(:,1),a(:,2)))
xlabel('sample offset (no unit)')
ylabel('xcorr(a.u.)')

hold on
a2=cacode(1,1.003);a1=cacode(1,1);  % 3.069kHz/1023kHz=0.003
plot(xcorr(a1-mean(a1);a2-mean(a2)),'c')
a2=cacode(1,1.01);a1=cacode(1,1);  % 10kHz/1023kHz=0.01
plot(xcorr(a1-mean(a1);a2-mean(a2)),'g')
```

Figure 7.7: Octave script to test GPS PRN codes' robustness

As it is observable in the script of figure 7.7, it has been searched in advanced a function that generates the C/A code. The 10 bits PRN codes are generated thanks to the Matlab script *cacode.m* available at *Matlab Central* [36] and shown in Appendix B.
The function cacode(sv,fs) generates C/A codes for selected PRNs, up to 37 codes, although the GPS satellites use only 32 of them [7], as explained in chapter 1. The variable 'sv' refers to a vector containing PRN numbers to be generated, from 1 to 37, and the variable 'fs' stands for the number of samples per chip desired in the code sequence. In fs fractional samples are allowed but they must be 1 or greater.

Firstly, in the script all the 37 PRN codes have been created with one sample per chip, which is equivalent to possess a null frequency offset. Then, the mean is removed as every time the cross-correlation operation is implemented. And finally, the diverse cross-correlation are performed and shown in figure 7.8.

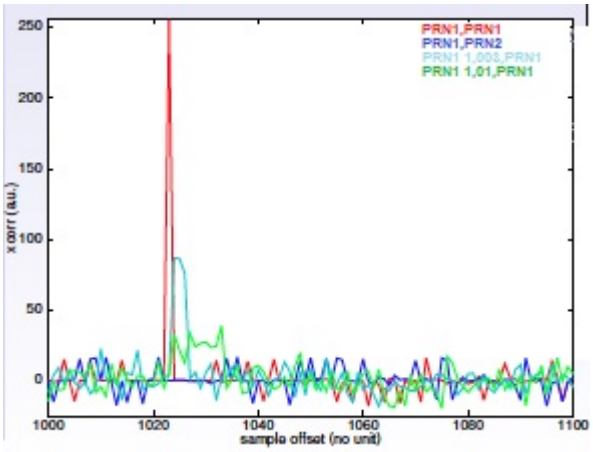

Figure 7.8: C/A codes cross-correlation

The most obvious cross-correlation line to start the explanation with is the red one, which represents the cross-correlation between code 1 and itself. While the blue one represents the non cross-correlation between code 1 and 2. In both situations it is chosen 1 sample per chip are 1.

Afterwards, two ranges of frequency offsets are introduced. On the one hand, the cyan line shows a decrease in the correlation result, since the chipping rate of one code has been incremented in a 0,3% (3,069 kHz/1023 kHz=0,003). In other words, a 3,069 kHz offset has been introduced in the C/A code chipping rate producing more samples than in the no offset case. This value still allows the maximum auto-correlation peak to be noticeable.

On the other hand, the green line corresponds to the result of the correlation between a code non frequency offset and one that its chipping rate has been incremented in a 1% (10 kHz/1023 kHz=0,01) from the initial value of 1, which represents a frequency offset of 10 kHz. It is too wide to allow the maximum auto-correlation peak to be undistinguishable from the blue line that represents a non cross-correlation.

7.2.2 Acquisition of true GPS signals

The method followed to acquire data information continues to be Parallel Code Phase Search Acquisition but the difference now is that the source is the GPS antenna connected to the DVB-T dongle.

```
more off;close all;clear all;set (0, "defaultaxesfontname","Helvetica")

%x=read_complex_binary('1413968290_capture.dat');
%x=read_complex_binary('1413968609_capture.dat');
x=read_complex_binary('1413968929_capture.dat');
time=[0:1/2e6:length(x)/2e6]';time=time(1:end-1);

for m=1:31
a=cacode(m,2/1.023); a=a-mean(a);
l=1;
m
for freq=8e4:200:1e5  %run through possible frequency offsets
    mysine=exp(j*2*pi*(-freq)*time);
    xx=x.*mysine;
    [u(l,m),v(l,m)]=max(abs(xcorr(a,xx)));
    l=l+1;
end
end
figure
imagesc(abs(u))
```

Figure 7.9: Octave script for the acquisition of a true GPS signal

The I&Q values are sampled at a sample rate of 2 MS/s and then saved in temporary files, which are read using the 'read_complex_binary'. They contain the carrier frequency moved to the frequency offset value, since it has not been compensated yet, the PRN code and the data navigation information. In order to observe maximum cross-correlations between these signals and the PRN codes replicas, the carrier frequency needs to be removed in advanced. To do so, a signal is created with the frequency offset as its characteristic parameter, like it has been realized in previous experiments. After mixing it with the output of the dongle and consequently, having removed the carrier frequency, the cross-correlation can be executed between the resulting signal and the PRN code replica generated with the *cacode* function.

A 2D plot showing the PRN number as X-axis and frequency offset in Y-axis is created.

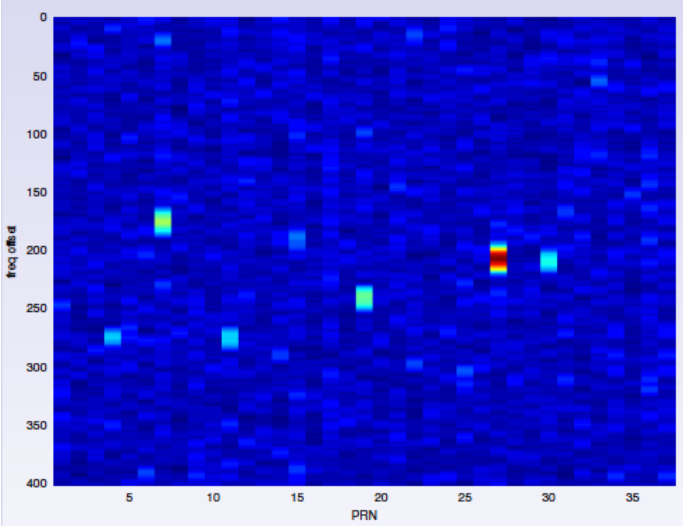

Figure 7.10: 2D plot acquisition

It represents the visible satellites from the user's location found thanks to the performance of the Parallel Code Phase Search Acquisition method implemented on the octave script.

Afterwards, as used before, thanks to the calibration software provided by gnss-sdr, the same data is processed to compare it with the previous Parallel Code Phase Search Acquisition results. The obtained values from gnss-sdr are the following.

```
Latitude=47.3 []
Longitude=6 []
Altitude=10 [m]
Doppler analysis results:
SV ID  Measured [Hz]   Predicted [Hz]
  4   93875.00   3547.36
  7   88625.00   -1439.45
  11   93625.00   3455.35
  15   89437.50   -694.02
  19   92000.00   1783.79
  27   90250.00   96.13
  30   90375.00   139.85
Parameters estimation for Elonics E4000 Front-End:
Sampling frequency =1999885.64 [Hz]
IF bias present in baseband=90082.96 [Hz]
Reference oscillator error =-57.18 [ppm]
Corrected Doppler vs. Predicted
SV ID  Corrected [Hz]   Predicted [Hz]
  4   3792.04   3547.36
  7   -1457.96   -1439.45
  11   3542.04   3455.35
  15   -645.46   -694.02
  19   1917.04   1783.79
  27   167.04   96.13
  30   292.04   139.85
GNSS-SDR Front-end calibration program ended.
```

Figure 7.11: Gnss-sdr results from the calibration software

Firstly, what the plot information describes is a brief summary of the user's location and secondly, detailed data about Doppler shift measurements is provided.

On the one hand, the values of the Doppler analysis given by gnss-sdr appear in the first two columns, which are useful to make comparisons with the ones manually calculated. The difference between the columns on the top, which show the measured and the predicted Doppler Shift in Hz, results to be the frequency offset introduced by the dongle's LO. This dongle's LO offset or crystal input is mentioned just below as 'IF bias present in baseband' or 'Reference Oscillator error', which in this case is 90082,96 Hz or -57,18 ppm. Finally, the two columns below show the corrected vs. the predicted Doppler shift in Hz, being the difference between both the effect of the temperature drift.

To conclude with, in both analysis the visible satellites are the same, which allows us to prove that the implementation of the Parallel Code Phase Search Acquisition on the octave script is correct. However, it was after the performance of this last experiment that we realized of the importance of the carrier recovery when wishing to recover the phase information to decode the navigation data. If the carrier is not perfectly tracked it subsequently means that the difference between the phase of the incoming signal and the one generated in the DCO is not null yet and therefore, the navigation data cannot be recovered. Figure 7.12 shows the result of the explanation for a determined frequency offset.

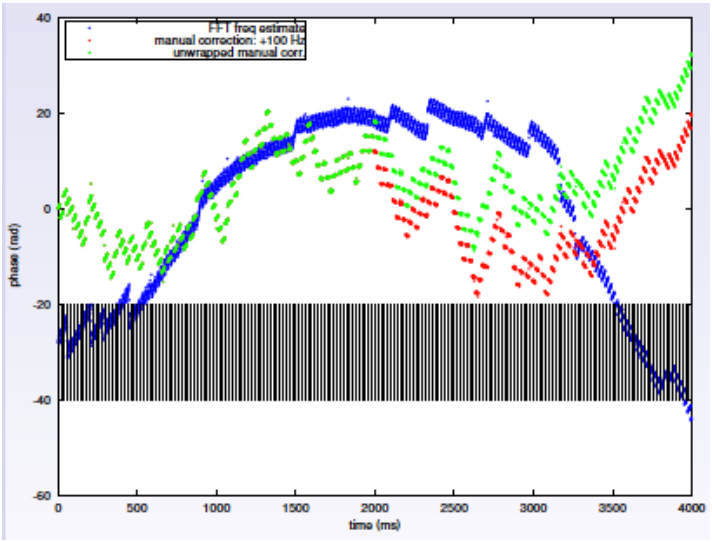

Figure 7.12: Phase-time plot

The blue line represents the first value obtained for the phase after a few seconds of record, which is not null at all and even, because of its changing slope, the navigation data bits are almost undistinguishable. The red line represents the manual correction of the phase, which is reached when tuning the carrier frequency with a deviation of 100 Hz, which allows us to differentiate the navigation data bits at least. The last line to

mention, the green one, shows the unwrapped phase correction, what means that the 2π phase jumps are removed.

8. Tracking

Once the acquisition part is realized, the receiver has the first rate for the frequency and code phase, and it is at this point where the tracking needs to be continuously running so that the system does not get unlocked because of the changes introduced on the values. Otherwise, a new acquisition must be performed for that particular satellite. Another purpose of tracking the carrier and the code is to determine more refined values [7].

The carrier tracking is accomplished by the Phase Locked Loop (PLL) that continuously compares the phase of the received carrier frequency from the satellite with the one from the carrier frequency generated in the local oscillator of the receiver until their difference becomes compensated.

At the same time, to accomplish the code tracking a process named Delay Locked Loop (DLL) is implemented, in which the receiver slides its replica code early or late in time until it synchronizes with the satellite code or i.e., the code gets tracked, resulting the signal's travel time the amount it needs to slide the replica code [8].

Figure 8.1 shows both tracking loops.

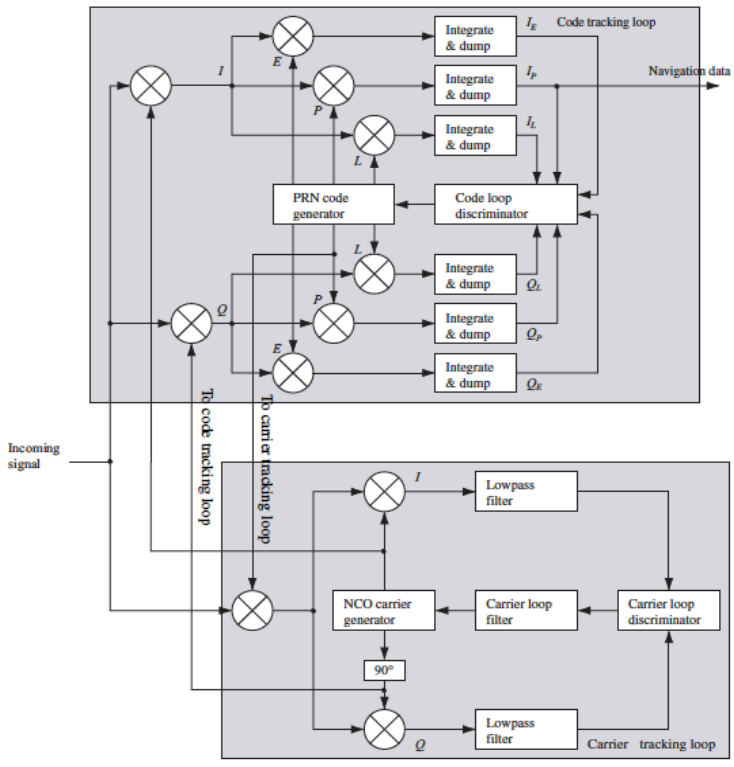

Figure 8.1: Block diagram of the DLL and PLL tracking loop [7]

8.1 Carrier tracking

What is accomplished with the carrier tracking is to compensate step by step the frequency offset of the dongle's LO so that the frequency that it generates gets closer until it reaches the one of the satellite. It is achieved through the already mentioned PLL, as a loop provided with: an error detector, a loop filter (that in fact is a controller) and a VCO, is continuously comparing both of their phases.

To perform the carrier tracking, a PLL is must be implemented but with the special feature that it has to be insensitive to 180 º phase rotations. On the opposite side, if the navigation data wants to be recovered, the phase shifts has to remain, since the data is the bit information changing at every period of time, i.e. the phase encodes the BPSK information. This is the reason why the PLL uses arc tangent as its discriminator, which does not keep the sign from the sinus (Q) and cosinus (I), while the function arc tangent 2 (atan2 in octave) keeps the sign from both, i.e. it provides the full phase including 180º rotations, and it is used for the data navigation recovery [1].

In a PLL it is desired a steady state error = 0.

In a FLL it is desired a steady state error = constant value.

When implementing a PLL using software, in this precisely case Octave, it is necessary to define the corrector, and the system that will be controlled. Figure 8.2 shows the structure of a PLL in an automatic way.

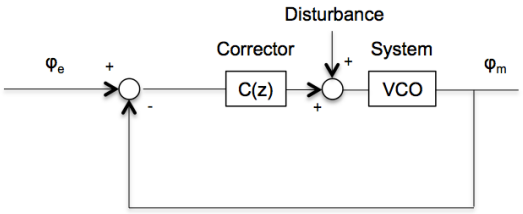

Figure 8.2: PLL in automatics

The system is the first one to be implemented because afterwards, the corrector will be created to compensate some of its effects. Since the VCO inputs a voltage and outputs a frequency but in a PLL the valuable information is a phase, an integrator is needed in the system that will have a shape similar to the following one.

$$\text{VCO} = \frac{k \cdot z^{-1}}{1 - z^{-1}}$$

The corrector in the PLL needs to be designed so that the loop converges to a stable solution and the output of the difference between phases is 0, since it is the aim of the loop. In this case of study it has been followed the Volgin method to design the

controller, in which three correctors are thought so that each one fulfils one function and at the end, they are mixed to work together as a whole one. To design each one of them they need to accomplish one of the following conditions.

1. The steady state error needs to be eliminated.
2. All what is compensable from the system, i.e. all of its stable functions, needs to be compensated.
3. It is a polynomial that adopts the shape of a desired corrector.

The correctors that suit the requirements the best are the following.

1. $C_1 = \frac{1}{(1-z^{-1})^{1,2\ or\ 3}}$

   The number of integrators is chosen depending on the input and disturbance type. Table 8.1 states the mentioned relation.

Nº of integrators	Step	Ramp	Parabola
0	$\frac{1}{1+K}$	∞	∞
1	0	$\frac{Te}{K}$	∞
2	0	0	$\frac{2Te^2}{K}$
3	0	0	0

Table 8.1: Error in function of the input and the number of integrators its controller has [37]

2. $C_2 = \frac{1}{k}$
3. $C_3 = \frac{N_3}{D_3}$

Once the three correctors have been designed, they result into C(z).

$$C(z) = C_1 \cdot C_2 \cdot C_3$$

$$\text{Open Loop} = C(z) \cdot VCO$$

$$\text{Closed Loop} = \frac{Open\ Loop}{1+Open\ Loop}$$

To find the values of N_3 and D_3 the closed loop needs to be equivalent to a function where we have decided which its poles are. The equation system is solved through the method of the Diophantine equation.

8.2 Code tracking

The aim of the code tracking is to estimate the distance travelled by the incoming signal as accurately as possible. For the code tracking to be set, three copies with a small offset from the reference initial position found in the acquisition part must be done. The three copies are one early, one prompt and one late, which differ between each one in ½ chip. More specifically, the prompt copy is the replica that is meant to have no offset with the PRN code received from the satellite, so the early copy is shifted to ½ chip earlier and the late copy is shifted to ½ chip later from the prompt that is the reference [7].

8.3 Carrier and code tracking implemented in software

Figure 8.3 shows the result of 9 seconds of recorded data tracked by a PLL and a DLL implemented in Matlab.

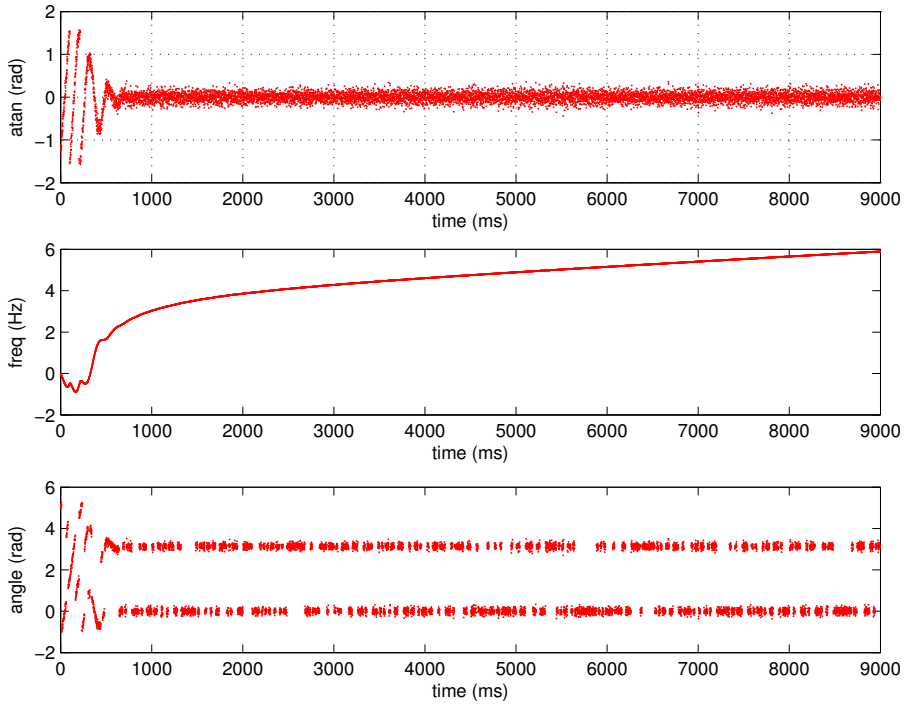

Figure 8.3: Results of 9 seconds of recorded data tracked by PLL and DLL

A stable result is obtained in which it is observable that the phase difference is very close to 0, the frequency tends to achieve a constant value while stabilizing and the navigation data bits, shown in the third graph, are distinguishable. The value for the parameter k that sets the gain of the VCO, which is later on compensated by the controller, has been experimentally found to be k = $\frac{1}{2\pi Te}$, being Te the time interval

between peaks when executing the cross-correlation between PRN codes (Te = $\frac{1023\ samples}{1{,}023{\cdot}10^{6}\ samples/s}$ = $1{\cdot}10^{-3}$ s).

9. Conclusions and future work

The ultimate purpose of this project was to be able to track the carrier frequency and the PRN code received from satellites emissions.

With the data recorded from the hardware installation in the laboratory, thanks to GNURadio-companion, it has been achieved a stable feedback loop during all the data record, which is 9 seconds. It tracks the received frequency also performing the phase modulation for the tracking and recovering the BPSK information. However, since the acquisition part has taken the most important role and quantity of testing time in this project, it is noticeable a lack of deeper investigation in the carrier and code tracking. In contraposition, the acquisition part has been perfectly fulfilled carrying out more complete tests while getting deeper in all its concepts. Moreover, it has been achieved the identification of the frequency offset source and established a range between the incoming carrier and local oscillator. The CDMA detection approach has been reached by the cross-correlation between the pseudorandom sequences, which has enabled us to afterwards, process real signals for acquiring the visible satellites and its frequency offset.

Some aspects remain uncertain when designing the loops, especially in the PLL filter or controller, it depends if the automatic's or electronics' term is employed, and in the true meaning of the DLL in the code tracking. Since the aim of implementing the tracking loops was not just to copy the filters realized by others Software Defined Radio producers but to understand how to manipulate them, we have been helped by Professor Gonzalo Cabodevila to design our own controller, obtaining results difficult analyse. One possible focus of problems could be the 2 MS/s sampling rate of the dongle, which does not meet the Nyquist criteria because the PRN code is sampled at 1.023 MS/s (bandwidth ± 1,023 MHz) that would imply to have at least a sampling rate of 2·1,023 MS/s performed by the dongle.

It has been proposed for the future work to record longer time data, although it is very time consuming to process them, and also to work with FPGA cards in order to be the substitution of the DVB-T dongle so that higher sampling rates can be implemented. However, in this last objective the goal of developing Software Defined Radio using very little and cheap hardware is lost, since these cards have an approximate price of 400 euros. The sense though, remains in the will of achieving better results and because the investigation is being undertaken in a research laboratory this possibility remains real.

Bibliography

[1] First-TF: partners. http://www.first-tf.com

[2] Kaplan, Elliott D., and Christopher J. Hegarty. *Understanding GPS. Principles and Applications.* Norwood: Artech house, 2006.

[3] Pogge, Richard W. *"Real World Relativity: The GPS Navigation System".* http://www.astronomy.ohio-state.edu/~pogge/Ast162/Unit5/gps.html. April 10, 204.

[4] Wikipedia: Information about the Global positioning System. http://en.wikipedia.org/wiki/Global_positioning_System

[5] Nutaq blog: An overview of GPS signal generation. http://nutaq.com/en/blog/overview-gps-signal-generation

[6] Wikipedia: Information about the Demodulation and decoding. http://en.wikipedia.org/wiki/GPS_signals#Demodulation_and_decoding

[7] Borre, Kai et. al. *A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach.* New York: Birkhäuser Boston, 2007.

[8] Trimble: Transforming the way the world works: Code-Phase GPS vs. Carrier-Phase GPS. http://www.trimble.com/gps_tutorial/sub_phases.aspx

[9] Fernández-Prades, Carles et. al. "*Turning a Television into a GNSS Receiver*". http://www.cttc.es/wp-content/uploads/2013/09/Turning_TV_into_GNSS_Rx1.pdf Septembre 15, 2013.

[10] ARRL: The national association for AMATEUR RADIO. http://www.arrl.org/software-defined-radio

[11] Wikipedia: Information about Software defined radio. http://en.wikipedia.org/wiki/Software-defined_radio

[12] Wikipedia: Information about Bias tee. http://en.wikipedia.org/wiki/Bias_tee

[13] Wikipedia: Information about Local oscillator. http://en.wikipedia.org/wiki/Local_oscillator

[14] Wikipedia: Information about Noise figure. http://en.wikipedia.org/wiki/Noise_figure

[15] Wikipedia: Information about Rubidium standard. http://en.wikipedia.org/wiki/Rubidium_standard

[16] Wikipedia: Information about Sensitivity. http://en.wikipedia.org/wiki/Sensitivity_(electronics)

[17] Wikipedia: Information about Johnson-Nyquist noise http://en.wikipedia.org/wiki/Johnson–Nyquist_noise

[18] Wikipedia: Information about Doppler effect Satellite communication http://en.wikipedia.org/wiki/Doppler_effect#Satellite_communication

[19] Kawasaki, Kenichiro. (Pioneer Electronic Corporation). *"Satellite transmission capturing method for gps receiver"*. http://www.google.com/patents/US5203030 US Patent 5203030 A. February 13, 1990.

[20] Friedt, Jean-Michel. "*La réception de signaux venus de l'espace par récepteur de télévision numérique terrestre*". http://jmfriedt.free.fr/sdr2.pdf. November 7, 2014.

[21] Rubiola, Enrico. *"Phase Noise and Frequency Stability in Oscillators".* New York: Cambridge University Press, 2009.

[22] Quartz crystal: Design notes. http://www.foxonline.com/pdfs/xtaldesignnotes.pdf

[23] Bavaro, Michele. "*GNSS carrier phase, RTLSDR, and fractional PLLs (the necessary evil)".* http://michelebavaro.blogspot.fr. May 12, 2014.

[24] http://en.wikipedia.org/wiki/Nyquist–Shannon_sampling_theorem

[25] Wikipedia: Information about Frequency mixer. http://en.wikipedia.org/wiki/Frequency_mixer

[26] Rubiola, Enrico. *"Tutorial on the double balanced mixer".* arXiv:physics/0608211. August 21, 2006.

[27] Wikipedia: Information about diode. http://en.wikipedia.org/wiki/Diode

[28] Wikipedia: Information about Voltage controlled oscillator. http://en.wikipedia.org/wiki/Voltage-controlled_oscillator

[29] Johansson, Fredrik et. al. "*GPS Satellite Signal Acquisition and Tracking*". *http://www.sm.luth.se/csee/courses/sms/019/1998/navstar/navstar.pdf.* August 21, 1998.

[30] Wikipedia: Information about Fast Fourier transform. http://en.wikipedia.org/wiki/Fast_Fourier_transform

[31] Wikipedia: Information about Fourier transform. http://en.wikipedia.org/wiki/Fourier_transform

[32] Wikipedia: Information about Convolution. http://en.wikipedia.org/wiki/Convolution

[33] Roddier, François. *Distributions et transformation de Fourier: à l'usage des physiciens et des ingénieurs*. Paris: McGraw-Hill, 1982.

[34] Wikipedia: Information about Attenuator. http://en.wikipedia.org/wiki/Attenuator_(electronics)

[35] Wikipedia: Information about Robustness. http://en.wikipedia.org/wiki/Robustness

[36] Matlab central: GPS C/A code generator. http://www.mathworks.com/matlabcentral/fileexchange/14670-gps-c-a-code-generator/content/cacode.m

[37] Cabodevila, Gonzalo. *"Automatique lineaire échantillonée".* Book for the 1st semester of the 1st year in École Nationale Supérieure de Mécanique et des Microtechniques.

Appendix

Appendix A. C/A code phase assignment

Satellite ID number and GPS PRN signal number	Code phase selection G_2	Code delay chips	First 10 chips octal
1 1	2 ⊕ 6	5	1440
2 2	3 ⊕ 7	6	1620
3 3	4 ⊕ 8	7	1710
4 4	5 ⊕ 9	8	1744
5 5	1 ⊕ 9	17	1133
6 6	2 ⊕ 10	18	1455
7 7	1 ⊕ 8	139	1131
8 8	2 ⊕ 9	140	1454
9 9	3 ⊕ 10	141	1626
10 10	2 ⊕ 3	251	1504
11 11	3 ⊕ 4	252	1642
12 12	5 ⊕ 6	254	1750
13 13	6 ⊕ 7	255	1764
14 14	7 ⊕ 8	256	1772
15 15	8 ⊕ 9	257	1775
16 16	9 ⊕ 10	258	1776
17 17	1 ⊕ 4	469	1156
18 18	2 ⊕ 5	470	1467
19 19	3 ⊕ 6	471	1633
20 20	4 ⊕ 7	472	1715
21 21	5 ⊕ 8	473	1746
22 22	6 ⊕ 9	474	1763

23 23	1 ⊕ 3	509	1063
24 24	4 ⊕ 6	512	1706
25 25	5 ⊕ 7	513	1743
26 26	6 ⊕ 8	514	1761
27 27	7 ⊕ 9	515	1770
28 28	8 ⊕ 10	516	1774
29 29	1 ⊕ 6	859	1127
30 30	2 ⊕ 7	860	1453
31 31	3 ⊕ 8	861	1625
32 32	4 ⊕ 9	862	1712
— 33	5 ⊕ 10	863	1745
— 34	4 ⊕ 10	950	1713
— 35	1 ⊕ 7	947	1134
— 36	2 ⊕ 8	948	1456
— 37	4 ⊕ 10	950	1713

Table 1: C/A code phase assignment [6]

Appendix B. C/A code octave script generator (cacode(sv,fs))

```
function g=cacode(sv,fs)
% function G=CACODE(SV,FS)
% Generates 1023 length C/A Codes for GPS PRNs 1-37
% g: nx1023 matrix- with each PRN in each row with symbols 1 and 0
% sv: a row or column vector of the SV's to be generated
%         valid entries are 1 to 37
% fs: optional number of samples per chip (defaults to 1), fractional samples allowed, must be 1 or greater.
%
% For multiple samples per chip, function is a zero order hold.
%
% For example to generate the C/A codes for PRN 6 and PRN 12 use:
% g=cacode([6 12]),
% and to generate the C/A codes for PRN 6 and PRN 12 at 5 MHz use
% g=cacode([6 12],5/1.023)
% For more information refer to the "GPS SPS Signal Specification"
```

```
% http://www.navcen.uscg.gov/pubs/gps/sigspec/default.htm
%
% Dan Boschen 12-30-2007
% boschen@loglin.com

% Revision History
% rev 1.0 Dan Boschen 4-15-2007  Initial Release
%
% rev 1.1 Dan Boschen 7-15-2007 Corrected error with taps for PRN30, should be [2,7] was
%                               incorrect as [1 7]. Thank you Jadah Zak for finding this.
%
%
% rev 1.2 Dan Boschen 12-26-2007 Fixed column index error when ceil ~ L
%                               Thank you Jared Meadows for finding this.
%
% rev 1.3 Dan Boschen 12-30-2007 Changed comment "first order hold" to
% "zero order hold".
%
% rev 1.4 Dan Boschen 6-1-2010   Updated email address in comments

if nargin<2
        fs=1;
end

if (max(sv)>37) || (min(sv)<1) || (min(size(sv))~=1)
        error('sv must be a row or column vector with integers between 1 and 37\n')
end

if fs<1
        error('fs must be 1 or greater\n')
end

% force integers
testint=round(sv)-sv;
if testint ~= 0
        warning('non-integer value entered for sv, rounding to closest integer\n');
        sv = round(sv);
end

% table of C/A Code Tap Selection (sets delay for G2 generator)
tap=[2 6;
```

```
    3 7;
    4 8;
    5 9;
    1 9;
    2 10;
    1 8;
    2 9;
    3 10;
    2 3;
    3 4;
    5 6;
    6 7;
    7 8;
    8 9;
    9 10;
    1 4;
    2 5;
    3 6;
    4 7;
    5 8;
    6 9;
    1 3;
    4 6;
    5 7;
    6 8;
    7 9;
    8 10;
    1 6;
    2 7;
    3 8;
    4 9
    5 10
    4 10
    1 7
    2 8
    4 10];

% G1 LFSR: x^10+x^3+1
s=[0 0 1 0 0 0 0 0 0 1];
n=length(s);
g1=ones(1,n); %initialization vector for G1
L=2^n-1;
```

```
% G2j LFSR: x^10+x^9+x^8+x^6+x^3+x^2+1
t=[0 1 1 0 0 1 0 1 1 1];
q=ones(1,n);   %initialization vector for G2

% generate C/A Code sequences:
tap_sel=tap(sv,:);
for inc=1:L
    g2(:,inc)=mod(sum(q(tap_sel),2),2);
    g(:,inc)=mod(g1(n)+g2(:,inc),2);
   g1=[mod(sum(g1.*s),2) g1(1:n-1)];
   q=[mod(sum(q.*t),2) q(1:n-1)];
end

%upsample to desired rate
if fs~=1
        %fractional upsampling with zero order hold
        index=0;
        for cnt = 1/fs:1/fs:L
                index=index+1;
                if ceil(cnt) > L   %traps a floating point error in index
                        gfs(:,index)=g(:,L);
                else
                        gfs(:,index)=g(:,ceil(cnt));
                end
        end
        g=gfs;
end
```