Hindawi Publishing Corporation
International Journal of Navigation and Observation
Volume 2011, Article ID 979815, 27 pages
doi:10.1155/2011/979815

Review Article

Software-Defined Radio Technologies for GNSS Receivers: A Tutorial Approach to a Simple Design and Implementation

Fabio Principe,[1] Giacomo Bacci,[2] Filippo Giannetti,[2] and Marco Luise[2]

[1] *EM Framework Design Laboratory, Ingegneria dei Sistemi (IDS) S.p.A., 56121 Pisa, Italy*
[2] *Dipartimento di Ingegneria dell'Informazione, Università di Pisa, 56122 Pisa, Italy*

Correspondence should be addressed to Giacomo Bacci, giacomo.bacci@iet.unipi.it

Received 27 January 2010; Revised 2 December 2010; Accepted 17 February 2011

Academic Editor: Letizia Lo Presti

Copyright © 2011 Fabio Principe et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The field of satellite navigation has witnessed the advent of a number of new systems and technologies: after the landmark design and development of the Global Positioning System (GPS), a number of new independent Global Navigation Satellite Systems (GNSSs) were or are being developed all over the world: Russia's GLONASS, Europe's GALILEO, and China's BEIDOU-2, to mention a few. In this ever-changing context, the availability of reliable and flexible receivers is becoming a priority for a host of applications, including research, commercial, civil, and military. *Flexible* means here both easily upgradeable for future needs and/or on-the-fly reprogrammable to adapt to different signal formats. An effective approach to meet these design goals is the *software-defined radio* (SDR) paradigm. In the last few years, the availability of new processors with high computational power enabled the development of (fully) software receivers whose performance is comparable to or better than that of conventional hardware devices, while providing all the advantages of a flexible and fully configurable architecture. The aim of this tutorial paper is surveying the issue of the general architecture and design rules of a GNSS software receiver, through a comprehensive discussion of some techniques and algorithms, typically applied in simple PC-based receiver implementations.

1. Introduction

"*Software-radio is an emerging technology, thought to build flexible radio systems, multi-service, multi-band, multi-standard, reconfigurable, and reprogrammable by software.*"

This definition, borrowed by Buracchini [1], describes very well the key idea of the *software-defined radio* (SDR) design paradigm. More specifically, an ideal SDR transceiver, sketched in Figure 1, is characterized by a commodity off-the-shelf (COTS) front-end (FE) that is connected to a programmable processing unit (PU), which processes the baseband (BB) incoming, or transmitted, signal via flexible software [2–4]. The PU is the core of the system, since it can impart *multiple personalities* to the transceiver by running specific software, depending on the application context and the required functionalities.

The effectiveness and the advantage of a fully SW-defined radio like the one envisaged in Figure 1 with respect to an equivalent fully hardware device were first sketched in the seminal work by Mitola [3] and can be synthesized in three key points.

(i) *Configurability*. Within a specific communication technology, this means that a unique transceiver for different applications can be factory-configured without any hardware change.

(ii) *Updatability/Upgradeability*. Since the core is software based, the receiver can be updated and upgraded in case of new and improved algorithms or amendments to a standard, while maintaining the same hardware components.

(iii) *Flexibility*. The same transceiver, with the same hardware, can be used for different communication technologies, simply by reconfiguring or upgrading the software.

Obviously, these properties are not in general borne to a large extent in fully HW devices, which are usually constrained and confined to particular applications. The direct consequence of this limitation is, on one hand, the need to use multiple HW solutions for different technologies and, on the other, the possible need of HW replacements at the advent of even modest changes in standards or technologies. Still, we must see how the general notion of an SDR can be beneficial to the field of *global navigation satellite systems* (GNSSs).

Satellite navigation has witnessed the advent of a number of new systems and technologies: after the landmark design and development of the *Global Positioning System* (GPS) [5–7], a number of new independent GNSSs were designed and were or are being developed all over the world: Russia's GLONASS [8], Europe's GALILEO [9–11], China's BEIDOU-2 [12–14], and so forth. In addition, *augmentation* systems and technologies [6, 15], like WAAS in the USA, EGNOS in Europe, and MSAS in Japan, have been introduced to increase the availability and accuracy and to predict the integrity of GNSS. This multifaceted environment is a perfect context where an SDR-based navigation receiver can prove its virtues by adapting to the different available formats and/or jointly processing the host of novel as well as legacy positioning signals.

Recently, new features have been studied to improve the performance of the new GNSS, based on the experience of GPS. Multipersonality receivers switch from one satellite system to another and process their signals performing multiconstellation navigation [16, 17], and at the horizon there is also the integration of satellite navigation with different positioning data coming from different, possibly local, sensors (indoor or inertial navigation) [18]. In this framework, an SDR-based receiver has no difficulty in performing the necessary functions of *data fusion* among different sources of information.

All of the above represents, in the authors' opinion, a very well-founded motivation that led the application of the SDR paradigm to the design of GNSS receivers ([10, 19–25]). To this respect, this tutorial intends to provide an overview of the typical techniques used to implement GNSS SDR receivers. To this aim, more details on GNSS evolution are provided in Section 2. The software architecture of a GNSS SDR receiver is illustrated in Section 3. Section 4 focuses on the receiver signal processing techniques and navigation algorithms. Finally, conclusions and future perspectives for the research on SDR for GNSS are drawn in Section 5.

2. GNSS Evolution

In this section, we give a brief overview on GNSS systems and their evolution, proposing the basic concepts that are useful for a good comprehension of the next parts of this tutorial. The main purpose is to show a real context wherein a renovation process is required, which includes very flexible solutions and a coexistence between new and old signal processing techniques.

2.1. Basic Elements. GNSS systems encompass a wide variety of different networks of devices and technologies that are able to provide precise positioning to enabled users every time and everywhere in the world [26]. Specifically, in this wide and complex scenario, the so-called master systems, such as GPS, GLONASS, Galileo, and Compass, are stand-alone systems that estimate the user position without the support of any other external device. Conversely, GNSS *augmentation systems* are satellite- or ground-based systems designed to improve performance (mainly in terms of accuracy and precision) of the master ones, [5, 6].

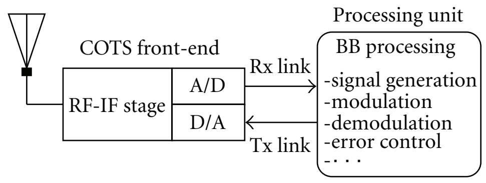


Figure 1: General SDR architecture.

Table 1: Space segment characteristics of the major GNSS systems.

	GLONASS	GPS	Galileo
Number of satellites	24	30	30
Number of orbital planes	3	6	3
Number of satellites per orbit	8	4	10
Orbital inclination	64.8°	55°	56°
Semimajor axis	25,508 km	26,560 km	29,601 km
Period	≈11 h 15 m	≈11 h 58 m	≈14 h 05 m

The overall structure of the master navigation systems, such as GLONASS, GPS, Galileo, and Compass, is made of three main elements, referred to as *segments*. The first one, named the *space segment*, includes a constellation of navigation satellites which transmit positioning signals containing fundamental data about the satellite position (such as ephemerides and almanacs), clock synchronism, and integrity. Such data are essential for typical receiver operations, like prediction of satellite visibility (to be performed at the start-up of the user terminal), computation of satellite positions, and estimation of ionospheric and tropospheric delays.

The constellation is made up of a certain number of satellites orbiting the Earth and is typically divided into orbital planes suitably designed to guarantee the full coverage of the Earth surface. The main parameters of the space segment of the major GNSS systems currently deployed or under deployment (GPS, GLONASS, and Galileo) are reported and compared in Table 1 ([5, 6, 8, 11, 27]). Besides those listed in Table 1, it is worthwhile mentioning also the future Chinese-made Compass navigation system, whose complete constellation (up-to-date information) will be constituted by 27 MEO (medium earth orbit) satellites with a semimajor axis of 27,840 km, 3 IGSO (inclined geosynchronous) satellites, and 5 GEO (geostationary earth orbit) satellites [12, 14, 28].

The second component is the *control segment*, which is in charge of monitoring the space segment by using a set

Table 2: Summary of the main parameters for GLONASS, GPS, and SBAS signals.

Signal parameter of feature		GLONASS	GPS	SBAS
Multiple access technique		FDMA	CDMA	CDMA
Carrier frequency (MHz)	L_1	1598.06 ÷ 1609.31	1575.42	1575.42
	L_2	1242.94 ÷ 1251.69	1227.60	—
Code length (chip)	C/A	511	1023	1023
	P	$5.11 \cdot 10^6$	$2.35 \cdot 10^{14}$	—
Code rate (Mchip/s)	C/A	0.511	1.023	1.023
	P	5.11	10.23	—
Navigation data	data rate (bit/s)	50	50	500 (coded) 250 (uncoded)
	data modulation	BPSK Manchester	BPSK NRZ	BPSK NRZ

of ground stations. More in detail, the tasks of the control segment are

(i) to track satellites and compute navigation data (to be uploaded to space segment),

(ii) to monitor and estimate the drift of the satellite clocks with respect to the system time,

(iii) to monitor and adjust the satellite direction to maintain the right orbit planes,

(iv) to relocate a new satellite in case of failure of an old one.

The control segment carries out then the key functions that allow the navigation signal to be broadcast everywhere and every time on the Earth, so providing worldwide coverage.

Finally, the last component of the system is the *user segment* that includes all the receivers (irrespectively of their specific application) which are enabled to acquire, track, and process the positioning signals provided by a given navigation system (e.g., GPS, GLONASS, etc.).

The augmentation systems are characterized by a large variety of different techniques to improve the performance of master systems, which can be classified into the following two subsets, [5, 6]:

(i) *pseudosatellites*, or *pseudolites* for short,

(ii) *differential GPS* (DGPS).

Specifically, the latter can be further subdivided in terrestrial and satellites systems, namely,

(i) *ground-based augmentation systems* (GBASs), also referred to as *terrestrial DGPS*,

(ii) *satellite-based augmentation system* (SBAS), which encompasses several types of systems such as *Wide Area Augmentation System* (WAAS), *European Geostationary Navigation Overlay System* (EGNOS), and *Multi-functional Satellite Augmentation System* (MSAS).

Each augmentation system operates in a different way for augmenting position accuracy. For instance, pseudolites emulate GNSS satellites on the ground. Conversely, DGPS augmentation systems are made up of known-position stations, called reference stations, which receive positioning signals from satellites, use them to calculate their position, and compare the result with their true position. This allows the reference stations to evaluate a set of corrections (also called *delta parameters*) to be broadcast to enabled user receivers. In the case of GBAS (see Figure 2), the corrections are computed by a *ground segment* made up of a single reference station with known position and broadcast to all receivers present within the station coverage area (generally pointed out as the *local area* of the DGPS reference station). In the case of SBAS (as shown in Figure 3), a *ground segment*, made up of a set of several base stations and one *master control* station, computes the corrections and sends them to a network of repeaters, made of one or more *geostationary satellites*, that cover a wide geographic area.

2.2. GNSS Signals. The main characteristics of GNSS signals, broadcast by current fully operative systems, are summarized in Table 2, which presents a comparison among GLONASS, GPS, and SBAS signals ([7, 8, 11]). Notice that P and C/A stand for *precision* code (i.e., long code yielding high accuracy) and *coarse/acquisition* code (i.e., short code providing low precision), respectively [5–7]. Also L_1 and L_2 are the labels assigned to the carrier frequencies 1575.42 MHz and 1227.6 MHz, whereas BSPK and NRZ stand for *binary phase shift keying* modulation and *non-return-to-zero* pulse shaping, respectively [7]. It is also worthwhile to remark that SBAS systems (like other GNSSs, such as Galileo [11]) use convolutional codes to protect their navigation data from corruptions. Thus, receivers for such systems also include a *Viterbi algorithm* for the maximum-likelihood error correction [29–31].

The first important common feature of the GPS and SBAS signals is the generation process: roughly speaking, they are generated by multiplying positioning data by a *spreading* sequence (also called *pseudorandom noise* (PRN) sequence). The PRN code identifies the satellite that generates the signal and reduces the cochannel interference due to other navigation signals operating at the same carrier frequency ([6, 32–34]). This technique guarantees a multiple

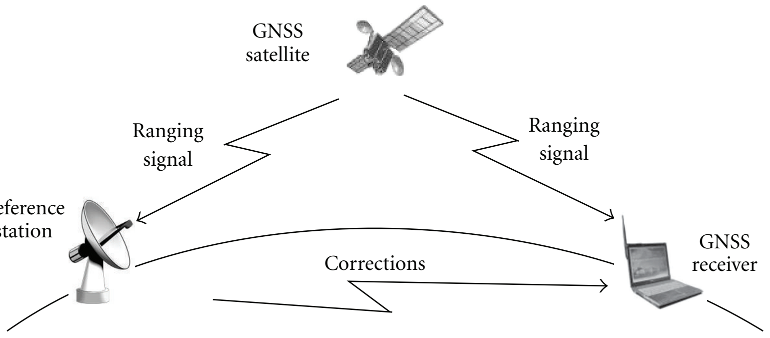


Figure 2: A general scheme of a DGPS.

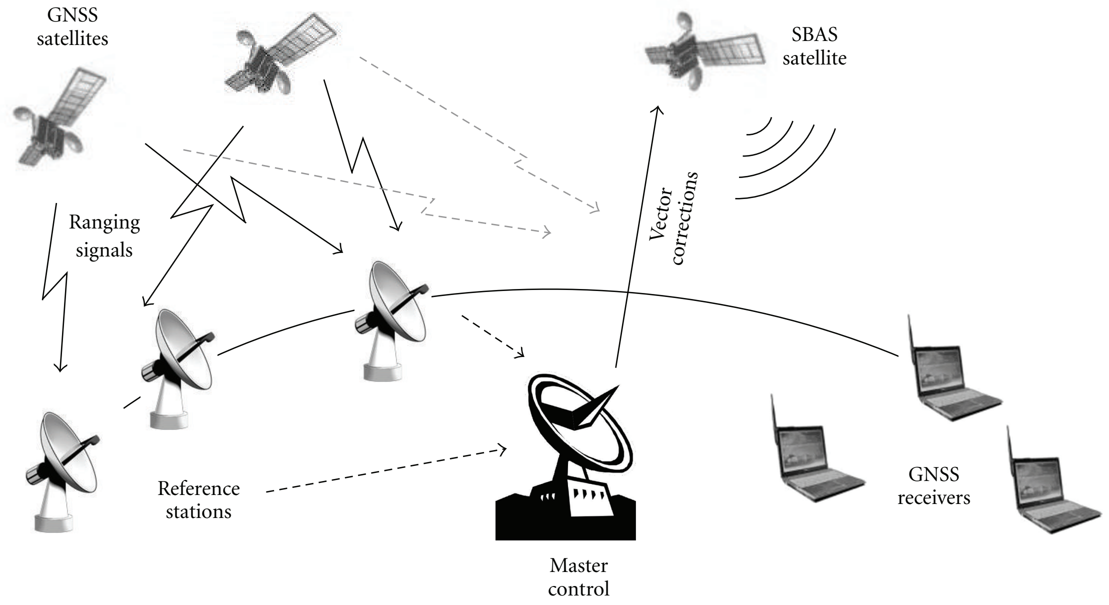


Figure 3: A general scheme of an SBAS.

access exploiting the same carrier frequency and bandwidth, and therefore it is referred to as *code division multiple access* (CDMA) [32–34].

A different approach, based on FDMA (*frequency division multiple access*), is actually adopted in GLONASS, wherein each satellite has its own frequency slot to transmit the ranging signal. Thus, each channel is identified by a specific carrier frequency, and the bandwidth becomes a fundamental parameter to reduce the interchannel interference. Nevertheless, the next evolution of GLONASS system will add a new set of BOC- (*binary offset carrier-*) modulated CDMA signals to the old one, so as to include new services and to improve user position accuracy [35].

An extensive modernization process is also involving GPS with the main goal of adding two civil signals, denoted as L2C (at 1227.6 MHz) and L5 (at 1176.5 MHz). L5 is intended for aeronautical and *safety-of-life* (SoL) applications [5], whereas L2C will be available for non-SoL applications at the L2 frequency.

The birth of a new generation of satellite navigation systems (such as Galileo and Compass) is taking an increasing attention from the scientific community as well. Indeed, even though the design of these systems dips the roots into GPS experience, their signal structure is definitely more complex than that of all the previous systems. For instance, referring to the Galileo *open service signals in space* (OS-SIS) [11], each satellite bears three carriers (E_5, E_6, and E_1), identified by three different *radio frequencies* (RFs), whereas GPS and GLONASS use only two carriers. Then, each Galileo carrier can contain either 3 or 4 channels (E_1 and E_6 have 3 channels, whereas E_5 has 4 channels), and every channel can be made up of a long spreading sequence (generated by multiplying a primary code by a secondary one) modulated by data and BOC-like waves. This elaborate design of Galileo *signals in space* (SIS) improves the accuracy of user-position determination with respect to GPS and GLONASS and provides more services for civil, commercial, and military applications [5, 10, 11].

In the same way, the Chinese satellite navigation system Compass (also denoted as Beidou-2, as it will replace the current Beidou, made up of only 3 operative GEO satellites over China), will transmit 8 ranging signals to cover a large variety of applications from civil to military ones [5, 26, 28, 36, 37]. For the sake of brevity, important projects (such as, the Japanese *Quasi-Zenith Satellite System* (QZSS) [5] and *Indian Regional Navigational Satellite System* (IRNSS) [26]) have not been reported. The interested reader can refer to the large bibliography available on these topics (see, e.g., [5–20, 26–28, 36, 37]).

3. Design of a GNSS SDR Receiver

Before going into the details of the design of a GNSS SDR receiver, it is important to introduce three basic definitions, [3, 4, 38].

(i) **Digital Radio**. After the front-end stage the incoming signal is converted into the digital domain to be processed by a *digital signal processor* (DSP). The processing is often done in a defined, nonreconfigurable way, using hardware and ASICs.

(ii) **Software Radio (SR)**. As technology progresses, the digitization is at (or very near to) the antenna and all of the processing required for the *digital radio* is performed by software residing in high-speed digital signal processing elements.

(iii) **Software-Defined Radio (SDR)**—The RF to IF is done prior to ADC, again followed by a *digital radio* that is performed by software residing on a programmable platform. Thus, the analog front-end remains unchanged, although all processing is fully reconfigurable. Therefore, the SDR solution represents a more realistic implementation of a digital radio.

A basic scheme of the general SDR architecture is sketched in Figure 1. The picture shows a *transceiver* (transmitter-receiver) made up of a front-end, followed by ADC and by the successive signal processing stage. As indicated in Section 1, this tutorial intends to provide an overview of the typical techniques used to design a GNSS SDR receiver, and the rest of this section is thus structured as follows.

(i) Section 3.1 gives a general overview on common SDR architectural designs.

(ii) Section 3.2 provides a brief description of the operation flow that characterizes a standard GNSS receiver.

(iii) Section 3.3 illustrates the general software architecture of a fully software GNSS receiver.

(iv) Finally, Section 3.4 briefly describes two possible software implementation techniques (with both a *single* and a *multi thread* architecture).

Some examples of different implementations and solutions are also given in [39–45].

3.1. Architectural Design of an SDR Receiver. Due to the need for multipersonality receivers mentioned in Section 1 and considering the GNSS signal details sketched in Section 2, ad hoc hardware solutions cannot be considered as viable and efficient solutions for the next generation of GNSS receivers. In this context, a more flexible approach is desired, and so the SDR technology represents a promising way to satisfy the new receiver requirements of low cost and reconfigurability.

There exists a wide range of possibilities for an SDR implementation, but all these architectures are characterized by a common core, represented by a programmable platform the receiver algorithms can be implemented and executed on.

For the sake of clarity, we will introduce three different SDR configurations, focusing on the receiver architecture reported in Figure 4. Five main sections can be identified: the receiving antenna, the analog/digital FE, the digital section with a *field programmable gate array* (FPGA) board and a DSP platform, and a PC board. Depending on the processing tasks allocated to each section (particularly, to the last three blocks), a specific type of SDR receiver is determined.

(i) *Classical (SW-Controlled) HW Architecture*. In this case, the FE stage gets in the RF signal, performs a downconversion to *intermediate frequency* (IF), and digitalizes the IF signal. The FPGA stage converts the incoming stream of samples down to (or near to) baseband, wherein a correlation module performs the despreading operation. The tracking of carrier frequency and phase offsets are also performed at baseband. The remaining baseband and application SW run on the DSP; in particular, the FPGA is programmed to perform basic high-rate operations (such as the downconversion and correlations), whereas the DSP performs all other baseband operations (including acquisition and tracking stages) and the relevant part of navigation SW. The PC board hosts the configuration functions only, including the user-PC interfacing (the man-machine interface (MMI)).

(ii) *Hybrid Architecture*. After the FE stage, the FPGA performs a considerable part of baseband operations (including acquisition), also hosting a high-speed connection to the PC board. The DSP provides the receiver initialization and configuration tasks, whereas the PC board performs the remaining (elaborate) baseband operations and the application SW. In particular, the PC board performs navigation algorithms, integrity algorithms, receiver management functions, and MMI handling.

(iii) *Fully SW*. In this case, the FPGA hosts service functions to transfer the digital samples to the PC, whereas the DSP performs receiver initialization and configuration tasks only; the PC board implements all baseband and navigation functions. Using this approach, the software is actually the core of the receiver, because it accomplishes all the critical tasks, such as processing of ranging signals and navigation data, management and synchronization of all receiver channels and logic stages, and MMI (see also [10, 19, 46–56]).

In the present tutorial, we will focus on *fully SW receivers* only. This means that all signal processing tasks will be performed in the software domain. As already mentioned in the introduction, we select this solution because fully SW architectures can be used when a high degree of flexibility is required. Hybrid HW/SW architectures, despite combining size and power efficiency with field programmability, and thus offering a full algorithmic and architectural reconfigurability, suffer in fact from a lower flexibility and an increased programmability complexity with respect to the fully SW architectures. For the same reason, SW-controlled HW receivers, which show even lower reconfigurability, are not considered throughout this tutorial.

However, it is worth noting that the architecture complexity increases as the number of tasks transferred from the FPGA/DSP digital section to the PC board increases. The fully SW configuration is in fact more demanding in terms of computational realization complexity, and it approaches the state of the art in SDR GNSS terminal design. These drawbacks will be carefully considered for the architectural choices discussed in the following.

3.2. Operations Flow of a GNSS Software Receiver. Hereafter, we will provide a brief description of the main stages of a generic GNSS receiver, so as to give an overview of the overall structure of these devices. To this respect, we will refer to the simplified and quite general flow graph of the SDR operations reported in Figure 5, wherein the software emulates the multichannel architecture of a standard GNSS hardware receiver.

3.2.1. Satellite in Visibility Prediction. At receiver start-up, the *satellite in visibility prediction* is performed exploiting old satellite almanacs, time, and a very rough estimate of the user receiver position. These input data are typically either provided by the user or locally stored at the end of the previous usage of the terminal. This provides a list of probable satellites in visibility to be used for a rapid acquisition and tracking (this is generally labelled as *warm start*). In case of a *cold start* (meaning that no a priori data are available), an exhaustive search during the *acquisition stage* is necessary to detect all satellite in visibility, and this translates into longer acquisition times. If more detailed information is available at the receiver during the start-up phase (namely, the *ephemeris data*, i.e, a sort of refined orbital parameters), the acquisition speed is faster than in the warm start (also called *hot start*).

3.2.2. Acquisition Stage (Coarse Acquisition of Synchronization Parameters). Upon the completion of the preliminary prediction algorithm, a list of satellites in visibility is generated and the *acquisition stage* can be run by emulating a multichannel receiver via software. This is carried out by assigning the PRN code of each incoming signal to a specific channel.

A typical approach in the GNSS software receiver design addresses about 12 channels to process master system signals (such as GPS or Galileo), plus 2 or 3 channels for the augmentation systems (such as EGNOS and WAAS). In the generic jth channel receive section, a local replica of the relevant PRN code is available for despreading. The *acquisition stage* of the jth channel provides a raw estimate for the code delay (also named code phase) and for the carrier frequency offset (mainly due to the Doppler effect, caused by the relative movement of satellites and user receiver, and to the clock drift) [5, 6, 10, 19, 57]. This raw synchronization is obtained by a correlative process that performs a two-dimensional search in the time and the frequency domains. When a correlation peak R crosses a given threshold λ, the acquisition stage ends, otherwise a new test is run with a different time/frequency offset combination.

3.2.3. Tracking Stage. Once a signal (e.g., the PRN-i signal) is acquired, the corresponding channel of the receiver switches to the *tracking stage*. This phase includes the following key operations.

(i) *Tracking of Synchronization Parameters.* In this stage, the code delay and the carrier frequency estimates are refined by a *delay-locked loop* (DLL) and a *frequency-locked loop* (FLL), respectively, whereas a *phase-locked loop* (PLL) is employed to recover the carrier phase offset [5, 6, 10, 19]. In this way, an accurate synchronization between the incoming signal and the local PRN code replica is achieved, which allows the despreading of the received signal and the detection of GNSS raw data to be performed. To check the synchronization, a code lock verification is periodically run by performing a long correlation between the incoming signal and the local replica of its spreading code. If the correlation peak R is larger than the threshold λ, all synchronization parameters are locked, otherwise the receiver channel goes back to the acquisition stage to perform a new estimation of code phase and carrier frequency offset.

(ii) *Frame Detection.* This stage consists in grouping raw data into a unique frame and computing pseudoranges, by exploiting the code phase provided by the DLL [10, 19]. A parity check is also performed, and if this control fails, further tests are generally done before going back to the acquisition stage. If the parity control is passed, the user position can be calculated by the subsequent stage.

(iii) *Further Checks.* Additional tests can be carried out when the parity check control of the frame detection fails. For instance, these may include estimation of the RF signal power-to-noise spectral density ratio (C/N_0) and code lock verification.

3.2.4. Navigation Stage. If the parity control of the frame detection succeeds, the *navigation stage* is run and the user position can be eventually calculated [5, 6]. More in detail, this stage (i) collects all the data provided by every tracked channel, (ii) aligns all received data into a coherent set, and (iii) runs the navigation routines to estimate positions (at least four satellites are necessary) [5–8, 10, 11, 15, 19].

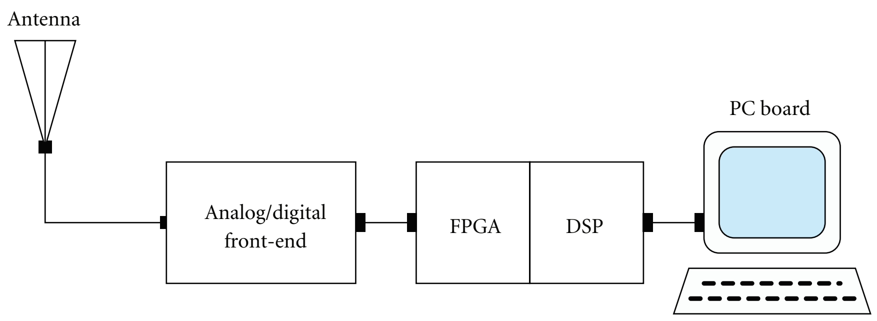


FIGURE 4: Hardware design of an SDR receiver.

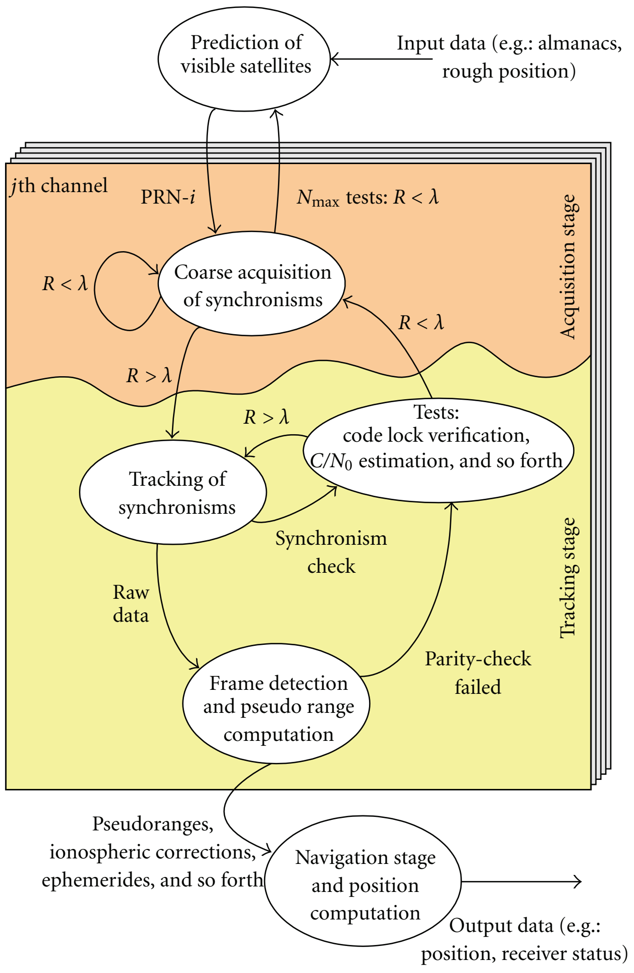

Prediction of visible satellites
Input data (e.g.: almanacs, rough position)
jth channel
PRN-i
$N_{\max}$ tests: $R < \lambda$
Acquisition stage
$R < \lambda$
Coarse acquisition of synchronisms
$R < \lambda$
$R > \lambda$
$R > \lambda$
Tests: code lock verification, C/N_0 estimation, and so forth
Tracking of synchronisms
Synchronism check
Tracking stage
Raw data
Parity-check failed
Frame detection and pseudo range computation
Pseudoranges, ionospheric corrections, ephemerides, and so forth
Navigation stage and position computation
Output data (e.g.: position, receiver status)

FIGURE 5: Main logic stages of a GNSS SDR receiver.

It is worth remarking that the signal processing of each channel is performed independently of the others, whereas the (only) navigation stage of the receiver requires to combine all the data from every channel to solve the mathematical system of positioning equations.

Finally, user position and other pieces of information (e.g., receiver status, satellite sky-plot, real-time C/N_0, etc.) are stored and/or displayed, depending on the final goal of the receiver and on the MMI.

3.3. Layer Architecture of a GNSS SDR Receiver. The signal processing and navigation stages addressed above are actually only a part of the tasks that are accomplished by a GNSS fully software receiver [52, 53]. Indeed, the software section also manages the user-machine interface and all the receiver stages and channels, to avoid conflicts that could cause data loss or, in the worst case, a crash of the *operating system* (OS). In this context, the *multilayer* software architecture, shown in Figure 6, is proposed as a general representation that allows us to highlight the main software stages, each one characterized by its own specific functionalities. This scheme shows three main different layers, or *functional blocks*, described hereafter in the next subsections.

3.3.1. Operating Layer. The *operating layer* (OL) is devoted to processing GNSS signals, to running navigation routines, and to solving positioning equations. Specifically, this level emulates the multichannel architecture of a standard navigation receiver, including the navigation stage, and the driver for the FE management and data buffering. More details will be reported in the next section.

Here, let us just remark that the driver allows the proper flow of digitized data from FE to PC. The time required to fill the *input buffer* (see Figure 6), called T_b, is an important constraint, because it constitutes the temporal interval during which all the receiver processing phases must be accomplished to guarantee real-time execution, avoiding bottlenecks, and consequent data loss. Specifically, T_b depends on the time required to collect N_s digital samples of GNSS signals (this operation is typically performed by the FE) plus T_D, that is, the time to download the stream of data into the input buffer: $T_b = N_s q/R_s + T_D$, where R_s is the bit rate at the sampler output and q is the number of bits per sample (bit/smp). Of course, considering that $T_D = N_s q/R_{\text{out}}$ where R_{out} denotes the FE output rate (in bit/s), the basic condition to avoid bottleneck and crash is $R_s \ll R_{\text{out}}$. (For instance, assuming a front-end with USB 2.0 connection ($R_{\text{out}} \approx 400$ Mbit/s) and a rate $R_s = 5$ Mbit/s, the time to fill a buffer $B \equiv N_s q = 1$ Mbit turns out to be $T_b = (1/5)\text{s} + (1/400)\text{s} = 0.2025\,\text{s} \approx 1/R_s$, since $R_s \ll R_{\text{out}}$.) To comply with the T_b constraint, it is mandatory to execute optimized and fast signal processing algorithms and to plan an ad hoc time schedule that assigns the right *priority* and a specific time slot to each receiver phase. In the SDR design,

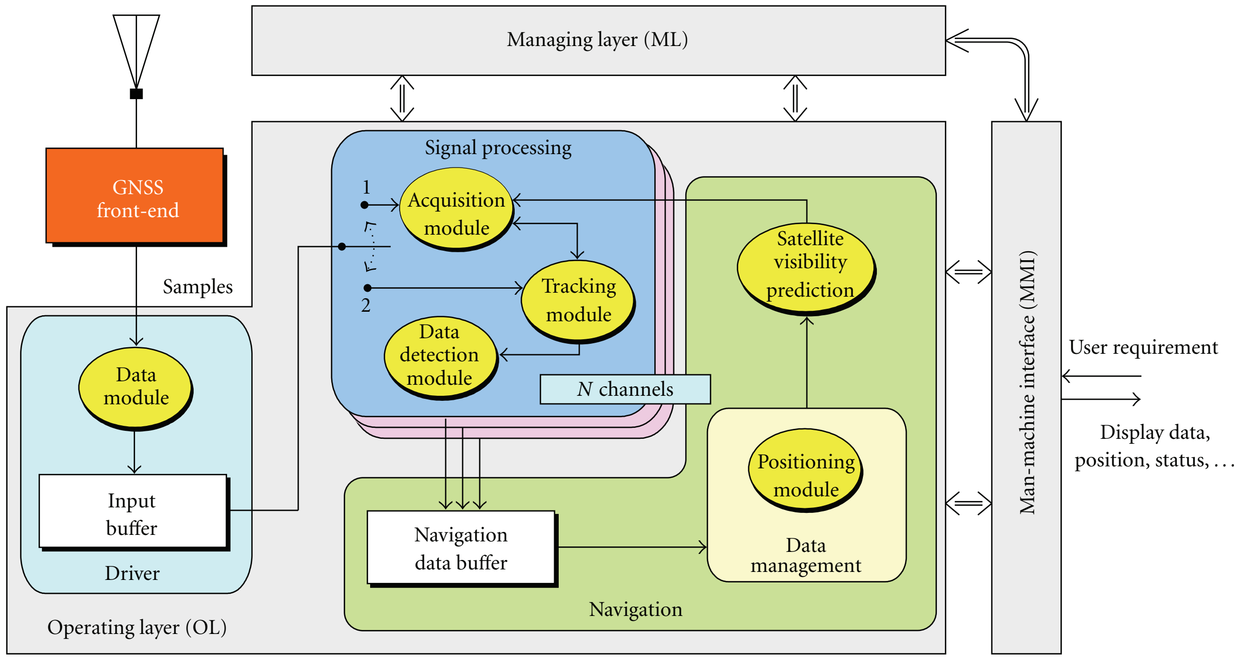


Figure 6: Architectural design of a GNSS software receiver.

this calls for identifying the stages that are more important than others to guarantee the real-time functioning of the receiver. These stages (characterized by strict requirements) shall have the highest priority.

3.3.2. Managing Layer. The compliance with the planned time schedule is guaranteed by the *managing layer* (ML). A simple example of time schedule is shown in Figure 7, which refers to a software receiver with 6 channels running on a standard PC with a single CPU. Time division of the T_b interval is apparent in the picture. The four channels in the tracking phase are assigned the highest priority to meet the real-time constraint, while maintaining time, phase, and frequency synchronization with the incoming signals. Thus, each pseudorange can be computed, yielding the GNSS raw data used to solve the positioning system. Soon after, two more channels try to acquire further GNSS signals, in order to improve the position accuracy and to guarantee the continuity of position computation in case any of the in-tracking satellites is lost. Afterwards, the driver downloads new samples from the FE. Finally, the navigation stage and other receiver functions, such as results displaying and almanacs and ephemerides storing, are run. Note that the rate to carry out such stages is typically much lower than that of the signal processing stage. Let us also remark that a *safety gap* g_s is scheduled for each time slot in order to avoid bottlenecks in critical situations due to possible overloads of the CPU. As mentioned, the ML activates every receiver phase following the step listed in the planned timetable, thus respecting the scheduling. Furthermore, it allows the change of channel stage from acquisition to tracking and vice versa and synchronizes raw data epochs obtained from different channels during navigation to solve the positioning system.

3.3.3. Man-Machine Interface. The *man-machine interface* (MMI) is the input-and-output interface between user and PC. In detail, the input options allow some different configurations for several operative contexts to be selected, including static or dynamic measures of the user position, off-line/real-time processing, and analysis of GNSS signals. Each choice is translated into an ad hoc setup of signal processing and navigation algorithms. The output is used to report and/or to store results, such as user position, SNR ratio level, satellite health, sky-plot of the GNSS satellite constellation, and receiver status. Typical output devices are monitors or hard drives where *log-files* can be saved.

Additionally, each layer (MMI, OL, and ML) communicates with the others by exchanging information. In this way, we can modify the ML timetable, exchange configuration data, and results between the MMI and the OL, configure signal processing algorithms, and so on.

3.4. Single versus Multithread Design. Finally, we emphasize that a software receiver can be implemented following two different approaches: using either a *single-thread* or a *multithread* design. In the first case, a single program carries out all tasks of the receiver, following a prefixed order. The result is a rigid structure which sequentially runs in loop all the stages (tracking, acquisition, etc.).

Instead, in the multithread approach, each receiver stage is assigned to a specific thread with a given priority level, and all threads are dynamically run in a concurrent manner.

4. Signal Processing Stage

This section illustrates the general receiver architecture and the typical algorithms used to process GNSS signals [5, 6, 10,

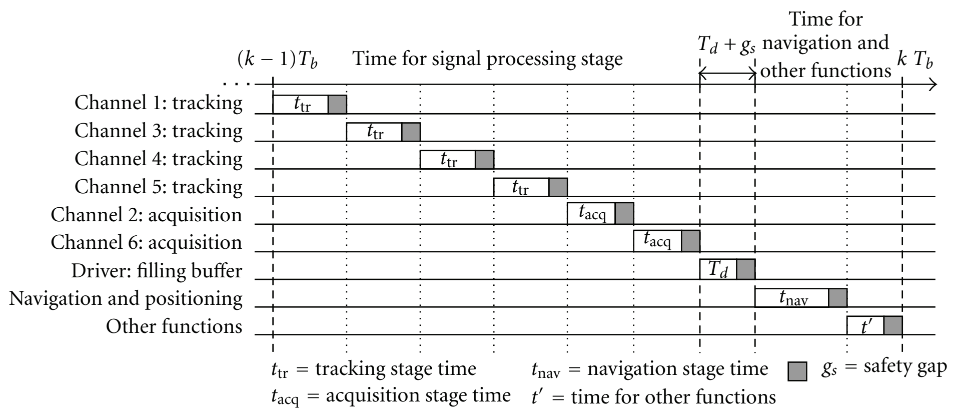


Figure 7: An example of a time schedule of a GNSS software receiver.

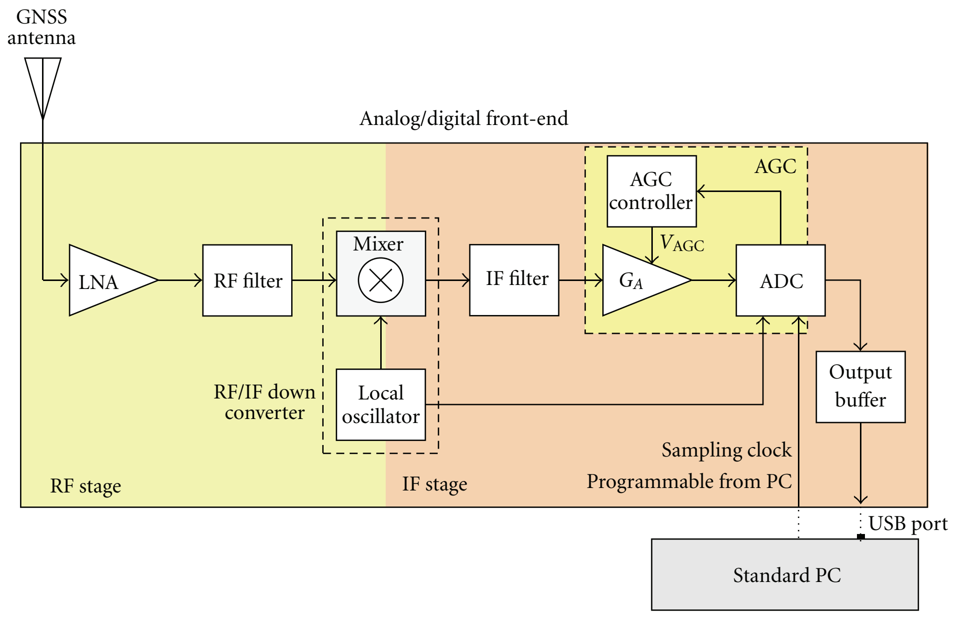


Figure 8: A simplified scheme of a GNSS front-end.

19, 21, 33, 41, 46–55, 57–67]. Referring to the architectural schemes reported in Figures 4 and 6, let us start describing the FE design with its main features, followed by acquisition and tracking stages and related algorithms. Then, the most common navigation algorithms to compute user position will be briefly presented.

To provide a clear and comprehensive description of a GNSS software receiver, the single frequency approach is proposed, considering the GPS L_1 signal as the reference channel. Nevertheless, let us remark that a similar analysis can be easily translated to a generic GNSS signal [41, 49, 51, 53–55, 60, 63, 65–67], by considering its specific characteristics (e.g., bandwidth, subcarrier modulation, carrier frequency, PRN codes, and data frame).

4.1. Front-End Stage. A simplified scheme of the FE architecture is sketched in Figure 8. The FE is clearly divided into two basic blocks, the RF stage and the IF stage, with the RF/IF downconverter acting as the interface between the two (more details are contained in [2, 4, 10, 46, 68, 69]).

4.1.1. Antenna. This first component is designed to generate a voltage from the incoming radio waves (e.g., $L_1 = 1575.42$ MHz). Thus, its frequency response must be similar to a band-pass filter centered in L_1. In case of a dual frequency antenna (e.g., L_1 and $L_2 = 1227.6$ MHz), the response is typically characterized by two narrow bands, centered on the two desired frequency ranges. In this way, the interferences due to the presence of signals between the two carriers are reduced. Other two parameters, typically used to characterize an antenna, are the *polarization* and the *pattern gain*. Specifically, the polarization indicates the electric field orientation of the received signal. GNSS signals are *right-hand circularly polarized* (RHCP). Hence, GNSS antennas are designed to receive incoming RHCP signals. We can effectively suppress *left-hand circular polarization*

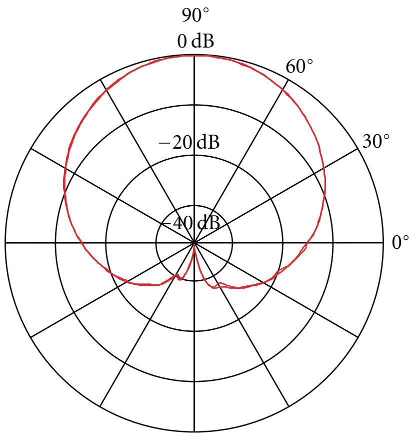


Figure 9: The antenna pattern of NovAtel GPS-533 at GPS L_1 [74].

(LHCP) rays (e.g., due to the reflection of RHCP signals), thus reducing the multipath effect. Note that multipath attenuation is mandatory, since it is the first cause of errors in the user position computation of GNSS receivers [6, 57, 70–73].

The pattern gain refers to the directivity of an antenna. An example of a pattern gain designed as a function of satellite visibility and multipath rejection is shown in Figure 9. A quasiuniform gain toward all GNSS satellites above a specific elevation angle is a desirable feature. Generally, this elevation mask is chosen around 5 ÷ 10 degrees above the horizon. Therefore, the antenna pattern is characterized by a low back lobe (Figure 9), in order to minimize the multipath effect due to the ground reflection and interferences which are present at low elevation [10, 19].

Commercial antennas are divided into two categories: *active* and *passive* antennas. In particular, the latter class of antennas are passive elements directly connected to the FE chain, whereas active antennas are featured by an integrated *low-noise amplifier* (LNA), with a very low *noise figure* (typical values are around 2 ÷ 3 dB).

4.1.2. RF Stage (LNA and RF Filter).

Referring to Figure 8, the RF stage is made up of a cascade of an LNA followed by an *RF filter*. The well-known *Friis formula* provides the overall noise figure of the FE [6]:

$$F_T = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \cdots + \frac{F_N - 1}{G_1 G_2 \cdots G_{N-1}}, \quad (1)$$

where F_i and G_i ($i = 1, \ldots, N$) are the noise figure and the gain of each component of the FE, respectively. If the first component is an LNA with a high gain G_1, F_T is dominated by the noise figure of the first element present in the FE chain, so that F_T is about F_1 (2 ÷ 3 dB). Possible impairments to this approach are the strong spectral components due to interferers in the amplifier bandwidth, which can saturate the LNA and thus generate spurious frequencies. This drawback can be circumvented by placing the RF filter before the LNA, in order to attenuate the out-of-band spectral components, thus avoiding saturation. This approach is particularly suited in the case of strong interfering signals adjacent to the L_1 carrier [10, 19]. However, this solution yields an overall noise figure which is typically 2 ÷ 3 dB higher than in the previous configuration, due to the *insertion loss* I_L of the filter that reduces the overall gain G_1 of the first stage (inclusive of RF filter plus LNA).

Narrow-band filters are obviously desirable to guarantee high frequency selectivity. Nevertheless, due to technical constraints, integrated narrow-band filters are scarcely available. Actual values for the filter quality factor $Q = f_c/B_W$, where f_c is the central frequency and B_W the bandwidth, turn out to be $Q \le 100$. Thus, assuming that $f_c = L_1$, the filter bandwidth results in $B_W \ge 20$ MHz, whereas the GPS C/A-band is only 2.046 MHz wide. So, the received signal, centered around RF frequency f_c, must be downconverted to a lower IF, to allow an easier spectrum filtering and guarantee a low output rate from the FE to the PC.

4.1.3. RF/IF Stage (Downconverter and IF Filter).

The RF/IF downconverter consists of an oscillator generating a local reference wave, at the frequency f_{LO}, that is multiplied with the incoming signal (e.g., GPS L_1 signal with $f_c = 1575.42$ MHz) by a mixer. Thus, assuming that the input signal is $s(t) = d(t)\cos(\omega_c t)$, with $\omega_c = 2\pi f_c$, the mixer output is

$$s(t)\cos(\omega_{\mathrm{LO}} t) = \frac{d(t)}{2}\{\cos[(\omega_c - \omega_{\mathrm{LO}})t] + \cos[(\omega_c + \omega_{\mathrm{LO}})t]\}, \quad (2)$$

where $\omega_{\mathrm{LO}} = 2\pi f_{\mathrm{LO}}$. Neglecting some mixer parameters such as conversion loss, intermodulation, and isolation, (2) produces a low-frequency $(\omega_c - \omega_{\mathrm{LO}})$ component and a high-frequency one $(\omega_c + \omega_{\mathrm{LO}})$. The difference frequency is the desired IF, whereas the other is cut off by the subsequent IF filter (Figure 8), typically a narrow band filter (e.g., with bandwidth 2 MHz) centered on the IF. This represents a single direct downconversion stage from RF to IF. In the practice, it is very common to design receiver FEs with two or more downconversion stages, which are more effective in rejecting *image frequencies*. More details on this aspect can be found in [4–6, 68, 75].

4.1.4. Automatic Gain Control and Analog-to-Digital Conversion.

The IF section may also contain an *automatic control gain* (AGC) circuit to adjust the signal level at the input of the ADC. Indeed, this second amplifier is controlled by a variable gain piloted by a feedback signal resulting from the ADC processing (thus, named *automatic gain control*). The gain G_A can be expressed as a function of the related AGC control voltage V_{AGC} [5]:

$$G_A = \alpha \cdot e^{\beta V_{\mathrm{AGC}}}, \quad (3)$$

where α is the AGC gain coefficient and β is the AGC control voltage coefficient (see Figure 8).

Roughly speaking, based upon monitoring the digitized data stream at the ADC output, the AGC increases or decreases the signal level at the ADC input, aiming at

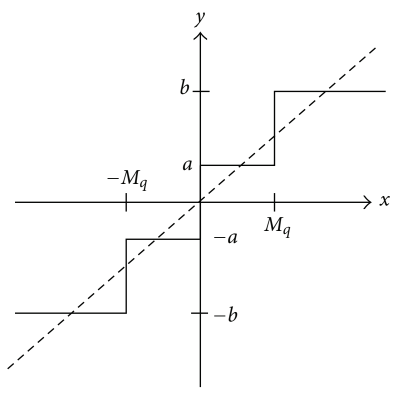


FIGURE 10: Uniform 2 bit quantization.

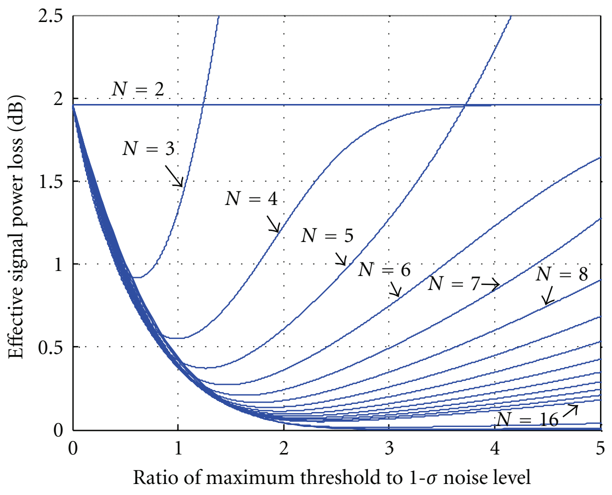


FIGURE 11: Postcorrelation C/N_0 degradation using an N-level quantization [77].

TABLE 3: Minimum degradation due to quantization.

	1 bit	2 bits	3 bits	4 bits
Optimal k_q	—	0.986	1.731	2.291
Min. degradation (dB)	1.96	0.537	0.159	0.014

exploiting the entire ADC dynamics, and ensuring thus the use of all quantization levels without incurring in saturation.

For GNSS receivers, in which the useful signal power is below that of the thermal noise floor, the AGC is driven by the ambient noise rather than the signal power. In such context, let us assume the absence of any interference. Considering the common case of uniform quantization [76], we can show that the degradation due to the digitization procedure is a function of the ratio of the maximum quantization threshold M_q (e.g., Figure 10) to the noise standard deviation, σ (e.g., see Figure 11 [26, 77]).

Some optimal values of $k_q = M_q/\sigma$ that minimize the degradation are given in Table 3 [26, 78]. Thus, the AGC regulates the Gaussian noise standard deviation at the input of the ADC, to ensure an optimal ratio. Therefore, this unit

TABLE 4: An example of f_s computation.

k	Minimum f_s (MHz)	Maximum f_s (MHz)
1	32.84	∞
2	16.42	28.84
3	10.94	14.42
4	8.21	9.61
5	6.57	7.21
6	5.47	5.77
7	4.69	4.81
8	4.11	4.12

is implemented as a feedback loop that uses the ADC output samples to form metrics used to adjust the AGC gain (3).

Another function of the AGC unit is to avoid the ADC saturation, when RF interferences occur. In this case, the AGC rapidly reduces its gain to maintain an optimal RMS level (at the ADC input) that can minimize the degradation. Therefore, when remarkable variations of the amplifier gain occur with respect to the nominal case, the AGC can also become a useful tool to detect and assess interferences. More details on this topic are given in [5, 78].

Finally, it is worth noting that, if a 1-bit ADC is used, no AGC is required because the ADC is just a limiter. This simplify the front-end architecture, but this solution results extremely vulnerable to continuous wave (CW) interferences [5]. In case of 2 (or more)-bit ADC, the AGC is necessary to reduce the degradation due to the quantization.

In the FE chain, the AGC is followed by an *analog-to-digital converter* (ADC) that digitizes the input signal, outputting a stream of bits that are grouped in the *signal buffer* to be sent to a PC. The main figures that characterize an ADC are the *sampling frequency* and the number of *quantization bits*. Specifically, the sampling frequency f_s can be calculated using the band-pass sampling theorem [79]:

$$\begin{cases} k \cdot f_s \geq 2f_{\mathrm{IF}} + B_W, \\ (k-1) \cdot f_s \leq 2f_{\mathrm{IF}} - B_W. \end{cases} \tag{4}$$

This system of simultaneous inequalities relates f_s to the IF frequency f_{IF} and the band-pass bandwidth B_W, as a function of the integer factor k, which can assume values in the range (see [79]):

$$1 \leq k \leq \left\lfloor \frac{f_{\mathrm{IF}} + (B_W/2)}{B_W} \right\rfloor \tag{5}$$

where $\lfloor x \rfloor$ denotes the integer part of the argument x. Let us give an example of f_s computation: supposing to use a local oscillator with frequency $f_{\mathrm{LO}} = 1560$ MHz, then $f_{\mathrm{IF}} = f_{L1} - f_{\mathrm{LO}} = 15.42$ MHz. The range of the sampling frequency as a function according to (4) is listed in Table 4 ($B_W = 2$ MHz). A pictorial representation of the band-pass theorem is displayed in Figure 12, with $k = 3$ and $f_s = 12$ MHz.

Generally speaking, the selection of f_s is done in accordance with the desired number of quantization bits,

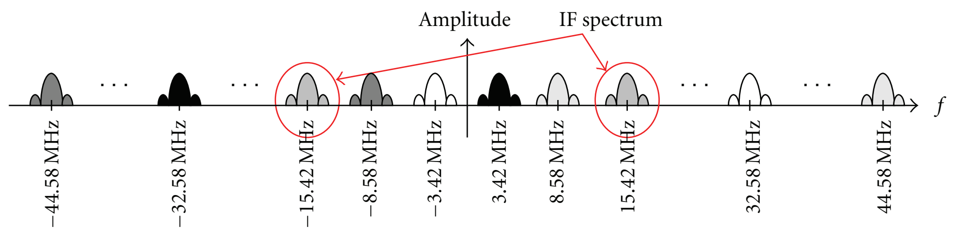


FIGURE 12: An example of the band-pass sampling theorem ($k = 3$ and $f_s = 12\,\text{MHz}$).

yielding an FE output rate which, to avoid bottleneck, must be lower than the maximum connection rate. For instance, considering the USB 2.0 maximum rate $R_M \approx 480\,\text{Mbit/s}$ and referring to the results reported in Table 4, a good choice is $k = 6$, that points to $f_s = 5.5\,\text{MHz}$, and a quantization of 2 bit/smp, and the output rate results $R_{\text{out}} \approx 11\,\text{Mbit/s} < R_M$.

The quantization operation is a nonlinear transformation which assigns a number (typically represented by a set of digits $\{0, 1\}$), belonging to an alphabet $\mathcal{A}$, to a real number, according to a specific quantization curve [76]. The quantization process degrades the input SNR, because the hard approximation of a soft input sample adds a random error, called *quantization noise*. For the sake of clarity, a very simple communication system (with infinite bandwidth, which includes just one GPS satellite, an AWGN channel, and no interferers) is simulated to compare the *bit error rate* (BER) obtained by an analog receiver to that of 1- and 2-bit quantization receivers. The results, reported in Figure 13, show that the loss due to uniform quantization is roughly 2 dB for the 1-bit receiver and that it decreases to 0.5 dB for the 2-bit receiver (as remarked in [6, 10, 19]).

It is worth noting that this result complies with the more general analysis proposed by Hegarty in [77] and shown in Figure 11. The picture reports signal losses for different uniform quantizer levels N as functions of maximum input threshold levels of the quantizer. Thus,

(i) with $N = 2$ (1 bit), the loss is about 1.96 dB,

(ii) with $N = 4$ (2 bits), the minimum loss (optimum threshold) is about 0.54 dB.

Hence, for standard applications, an SDR receiver is typically designed assuming 1-bit quantization, while the 2-bit quantization is recommended for better performance. More details on this topic can be found in [2, 6, 76].

4.1.5. Signal Buffer. The last block in the FE chain is the *signal buffer*. Here the bits output by the ADC are grouped into frames, to be completed with the necessary bits that satisfy the USB protocol (including a *packet identifier* (PID) and a *cyclic redundancy check* (CRC)). This message is then transmitted to the PC, as shown in Figure 8.

The signal buffer is detailed in Figure 14. The picture shows a data buffer (or equivalently, a stack of memory), placed right after the ADC, that is used to temporarily hold data while they are being moved from the front-end to the PC. *First in first out* (FIFO) is the rule used for the data transfer.

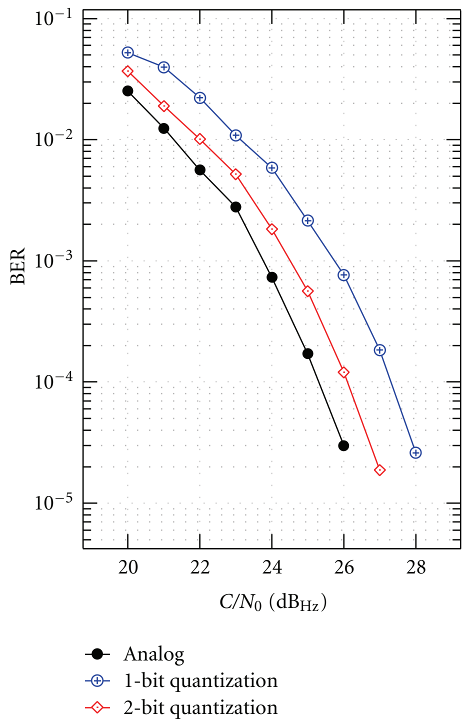


FIGURE 13: Analog versus quantized receiver BER.

The input rate, R_i, is imposed by the ADC (e.g., with a sampling frequency of 5.5 MHz and a 2-bit quantizer, $R_i = 11\,\text{Mbit/s}$), and the consequent output rate, R_o, must be larger than R_i to avoid buffer filling. If this occurs, a data loss event may happen.

The maximum output rate depends on the communication between the FE and the PC. In this kind of applications, the USB (*universal serial bus*) has become the standard connection method. Indeed, the USB 2.0 maximum rate is about 480 Mbit/s, that is more than enough to avoid the signal buffer filling and the subsequent data loss.

At the PC side, the communication is managed by the driver (see also Section 3.3.1) that receives data and packs them in arrays which are processed by the signal processing stage, as illustrated in the next sections.

The hardware stage ends here, so the next stages are fully developed in software, using high-level languages such as C/C++, MATLAB, SciLab. It is worth emphasizing that real-time software receivers require accurate design of the signal processing stages to cope with real-time constraints

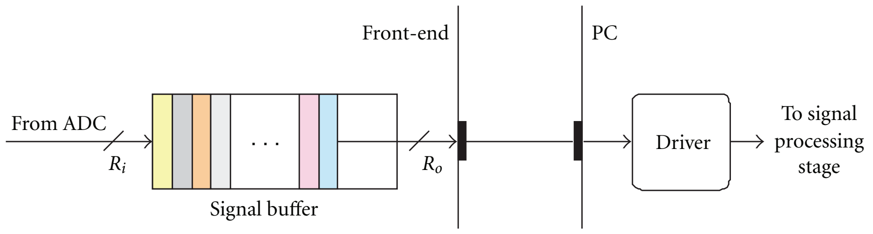


FIGURE 14: Signal buffer details.

[24, 54, 60]. More in detail, particularly critical are the acquisition engine, the local code generation, and the tracking loop architecture. The next subsection will provide more details on this issue.

4.2. Software-Based Signal Processing Stages. The perfect synchronization between the incoming positioning signal (in a CDMA format) and the local reference waves is a necessary requirement to correctly process GNSS signals. The despreading operation can be effectively performed only by ensuring the correct alignment of the received signal with the local replica of the PRN code (for time synchronization) and with the local carrier (for frequency synchronization). This is achieved in two steps [5, 6, 32, 33].

(a) First, the *acquisition stage* gets a preliminary coarse estimation of code timing and carrier frequency offset.

(b) Then, the *tracking stage* exploits this rough result to refine the synchronization and track the signal.

4.2.1. Acquisition Stage. The acquisition stage is the first step to get a preliminary coarse synchronization, in both frequency and time domains, that will be used in the subsequent tracking stage. CDMA signal detection is generally accomplished carrying out a two-dimensional search in time and frequency domain. Specifically, the time search is needed to synchronize the received PRN code with its local replica and consequently to despread the input signal. This step provides the necessary code gain to improve the signal SNR (for navigation data processing) by estimating the so-called *code timing* (or, equivalently, the *code phase*). Nevertheless, this time-alignment is not enough to correctly process a spreading signal, as motivated in the following. A despreading operation is basically a long correlation performed in the baseband domain between the received signal and a local replica of its PN code. Thus, remembering that the FE provides a digital signal at IF, the *baseband downconverter* becomes the first module to be run before the *correlator*. The BB signal is obtained by multiplying the received IF signal with a couple of local waves that are IF-synchronized. So, the result is a BB-replica of the input signal plus some high-frequency components which are filtered out. Of course, the consequence of any frequency misalignment produces a *frequency offset* in the output signal, which could cause a catastrophic power loss during despreading operation. Therefore, it is mandatory to accurately estimate any frequency deviation from the nominal IF value.

The main sources of frequency deviations are the Doppler shift, caused by the relative movement between GNSS satellites and user receiver, and possible frequency drifts, due to low-precision FE clocks. The latter is typically a small residual component that can be easily tracked using an FLL or a 2nd-order PLL during the tracking stage. Conversely, in the worst case, for example, in the case of the GPS receiver onboard an aircraft [6], the unknown Doppler shift can range from -10 kHz to 10 kHz, thus producing significant carrier frequency deviations. Therefore, this large interval cannot be covered by the limited dynamic of an FLL/PLL module. Hence, it needs to be firstly recovered performing an exhaustive search in the frequency domain [5, 6, 19]. The residual offset will be compensated by an FLL/PLL in the tracking stage.

Let us assume acquiring a GPS L_1 signal. In this case, the Doppler effect shifts the nominal carrier frequency ($f_{L1} = 1575.42$ MHz) by f_D with $f_D \in [-10, 10]$ kHz. This means that the RF becomes $f_0 = f_{L1} + f_D$ and, consequently, IF becomes $f_1 = f_{\mathrm{IF}} + f_D$. Thus, the BB downconverter (that is IF-synchronized) translates the input signal from f_1 to near-BB, and the carrier frequency offset that remains to be assessed (called carrier *Doppler shift*) is $\Delta f = f_1 - f_{\mathrm{IF}} = f_D$, with $\Delta f \in [-10, 10]$ kHz. Hence, the acquisition stage must provide a rough estimation of the carrier frequency offset Δf together with the code timing to process the incoming signal. The estimated figures during this phase are then used in the tracking stage to lock and track the signal.

The simplest algorithm used to acquire a spread-spectrum signal (such as a GNSS signal) is the *simple serial search* [32, 33, 80–82]. This technique performs an *exhaustive search* (also defined *brute search*) in the frequency and time domain till the incoming signal is detected. This happens when a prefixed threshold λ is crossed, otherwise a new frequency offset and/or a new code phase is tested.

A basic scheme for the simple serial algorithm is given in Figure 15, where the main modules are (i) the BB downconverter, that performs IF-to-BB conversion, (ii) the correlation module that despreads the input signal, (iii) a squarer that removes the carrier phase dependence, (iv) a decision unit that checks the signal detection using the threshold λ, and (v) a logic unit that shifts the local code and/or changes the tentative frequency in case of no acquisition.

For the sake of a better understanding of this algorithm, let us provide more details on the acquisition of a GPS

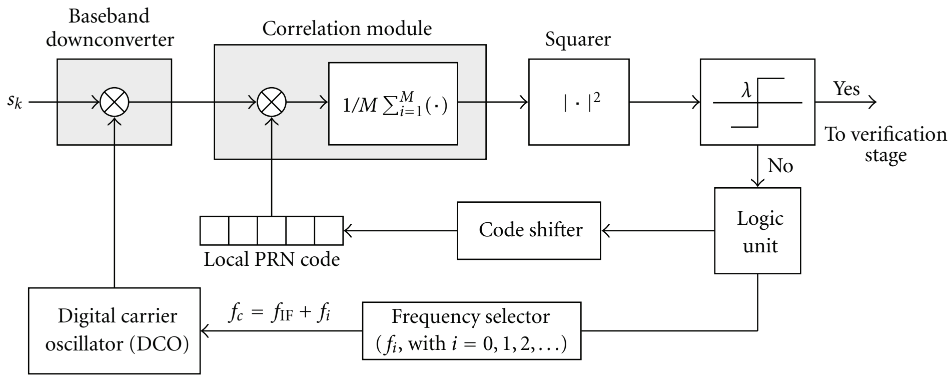

Baseband downconverter
Correlation module
Squarer
s_k
$1/M \sum_{i=1}^{M}(\cdot)$
$|\cdot|^2$
λ
Yes
To verification stage
No
Logic unit
Code shifter
Local PRN code
Digital carrier oscillator (DCO)
$f_c = f_{\mathrm{IF}} + f_i$
Frequency selector (f_i, with $i = 0, 1, 2, \ldots$)

FIGURE 15: Simple serial search architectural scheme.

L_1 signal. As illustrated in Section 4.1 (neglecting the quantization and the USB link), the incoming signal at the end of the FE chain becomes

$$s_k = s(k \cdot T_s) = \Re\left\{s_b(k \cdot T_s) \cdot e^{j[2\pi(f_{\mathrm{IF}}+f_D)kT_s+\varphi]}\right\}, \tag{6}$$

where f_{IF} is the IF, $T_s = 1/f_s$ is the sampling time in compliance with the band-pass sampling theorem [79], f_D is the Doppler shift, φ is the carrier phase, and $s_b(t) = i(t) + j \cdot q(t)$ is the L_1 signal in the BB domain. More in detail, $i(t) = p(t) \cdot d(t)$ is the in-phase signal composed of the P code, $p(t)$, and the GPS data sequence, $d(t)$, whereas $q(t) = \sqrt{2} \cdot c(t) \cdot d(t)$ is the in-quadrature signal, composed of the C/A code, $c(t)$, with the correspondent GPS data sequence (see also [7]). For the sake of simplicity, some additional contributes (such as, thermal noise, multipath, and interferences) have been neglected in (6).

Looking at the acquisition scheme (Figure 15), the first operation is the IF-to-BB downconversion that outputs the in-phase I_k and quadrature Q_k components as follows:

$$\begin{aligned} I_k &= s_k \cdot 2 \cdot \cos[2\pi \cdot (f_{\mathrm{IF}} + f_i) \cdot k \cdot T_s + \theta], \\ Q_k &= s_k \cdot 2 \cdot \sin[2\pi \cdot (f_{\mathrm{IF}} + f_i) \cdot k \cdot T_s + \theta], \end{aligned} \tag{7}$$

where f_i is a tentative estimate of the Doppler shift and θ is the initial phase. After some manipulations, (7) becomes

$$w_k = I_k + j \cdot Q_k = (i_k + j \cdot q_k)e^{-j(2\pi f_\Delta kT_s+\varphi_\Delta)} + h_k, \tag{8}$$

where $i_k = i(kT_s)$, $q_k = q(kT_s)$, $f_\Delta = f_D - f_i$, $\varphi_\Delta = \varphi - \theta$, and h_k represents the high-frequency components (at $2f_{\mathrm{IF}}$) produced by the downconversion operation.

Now, despreading is done by correlating w_k with the C/A PRN sequence c_n (GPS P code is encrypted for military application). Thus, we get

$$r_k(\delta) = \frac{1}{M}\sum_{n=k}^{M+k-1} w_n \cdot c_{n+\delta}. \tag{9}$$

Since the despreading process with the local C/A code cancels the contributes of P code and the averaging operation filters the high-frequency components, (9) becomes

$$r_k(\delta) \approx j\frac{e^{-j\varphi_\Delta}}{M}\sum_{n=k}^{M+k-1} q_n \cdot c_{n+\delta} \cdot e^{-j2\pi f_\Delta nT_s}. \tag{10}$$

Assuming that no data transition occurs during the MT_s samples and that $f_\Delta T_s \ll 1/(2\pi M)$, then $r_k(\delta)$ can be written as

$$r_k(\delta) \approx j\sqrt{2} \cdot e^{-j(2\pi f_\Delta kT_s+\varphi_\Delta)} \cdot d_k \cdot \gamma_k(\delta), \tag{11}$$

where $d_k = d(kT_s)$ and

$$\gamma_k(\delta) \triangleq \frac{1}{M}\sum_{n=k}^{M+k-1} c_n \cdot c_{n+\delta} \tag{12}$$

is the normalized correlation function of the C/A PRN sequence [6, 10, 32].

The two previous assumptions are satisfied if the following conditions are verified:

(a) the correlation period $T_I \triangleq MT_s$ is much lower than the data duration T_d to reduce the event of data transition;

(b) the condition $f_\Delta T_s \ll 1/(2\pi M)$ is satisfied when the tentative Doppler frequency f_i is a good approximation of the actual Doppler shift ($f_i \approx f_D$) in accordance with correlation period length.

The *power loss* (PL) due to an eventual residual frequency in the correlation module can be evaluated as

$$\mathrm{PL} \cong -10\log_{10}\left|\frac{\sin(\pi \cdot f_\Delta \cdot T_I)}{\pi \cdot f_\Delta \cdot T_I}\right|^2 (\mathrm{dB}). \tag{13}$$

Going through the acquisition chain shown in Figure 15, the correlation module is followed by a squarer, which deletes any carrier phase contribute. The output of the squarer is then compared with a threshold λ, to detect whether a rough alignment with the local replica of the C/A code is achieved or not. If λ is crossed, a coarse synchronization occurs and a verification stage is typically run to check the signal detection, otherwise a new acquisition is tried. The latter case can be caused either by a large code misalignment (larger than the chip time) or by a catastrophic degradation of the signal power due to a bad estimation of the Doppler shift. Thus, the next acquisition is done by either shifting the local code phase or updating the tentative frequency.

The threshold λ should be selected to maximize the detection probability and to minimize the acquisition time and the false alarm probability. In such sense, optimal values can be also derived by simulations. We can also use an adaptive threshold instead of a fixed one [83–85]. In this case, we need a coarse estimation of the SNR to select a specific value of λ.

In the discussion above, we intentionally neglected multipath distortions, interferences, and thermal noise, effects that must be considered in the design of a practical acquisition algorithm. These effects induce a degradation of the C/N_0 ratio, which increases the occurrence of false alarm and missed detection events. In the case of the GPS L_1 signal, the typical range for C/N_0 is $35 \div 55\,\mathrm{dB_{HZ}}$ [57]. Hence, when $C/N_0 < 35\,\mathrm{dB_{HZ}}$, the detection of positioning signals becomes critical. More details on the detection of CDMA signals can be also found in [6, 32, 33, 80–86].

In the following, we show a sample implementation of the acquisition stage in a GPS/SBAS software receiver (more examples can be found in [10, 19, 46, 48, 51–53]). The relevant architectural scheme is reported in Figure 16. This block diagram (like that depicted in Figure 15) shows an FE using 1-bit digitized data followed by a digital IF-to-BB downconverter.

From (6), the single sample can be written as

$$s_{\mathrm{IF}}[k] = u[s_k + n_k], \tag{14}$$

where n_k is the noise sample (generally modeled as a Gaussian process) and $u[\cdot]$ is the *step function* ($u[x] = 1$ if $x \geq 0$ and $u[x] = 0$ if $x < 0$). Thus, the downconversion (7) can be implemented as

$$\begin{aligned} I_k &= \overline{s_{\mathrm{IF}}[k] \oplus u[\cos(2\pi f_{\mathrm{IF}} \cdot k \cdot T_s)]}, \\ Q_k &= \overline{s_{\mathrm{IF}}[k] \oplus u[\sin(2\pi f_{\mathrm{IF}} \cdot k \cdot T_s)]}, \end{aligned} \tag{15}$$

where $\oplus$ is the XOR operator, $\theta = 0$, and $\overline{x}$ is the NOT operation on the logical variable x. This formulation minimizes the processing of the PC, since every Boolean operation on 32- or 64-bit (according to the PC architecture) words is done within exactly one clock tick. Therefore, 32 or 64 samples (32/64 bits) grouped in 32- or 64-bit words can be downconverted during two clock ticks (due to the XOR and NOT operations). Additionally, instead of computing every sample of $u[\cos(2\pi f_{\mathrm{IF}} \cdot k \cdot T_s)]$, it is preferable to implement a look-up table containing one period of $u[\cos(2\pi f_{\mathrm{IF}} \cdot k \cdot T_s)]$ that is also representative of the sine component. This approach does efficiently reduce the computation of local waves, saving both time and memory.

Going back to Figure 16, the next module (called correlation module) splits the incoming signal on three parallel branches, labeled *early*, *prompt*, and *late*. Each branch correlates the input signal with a shifted local replica of the received spreading code. More specifically, the correlation is done with three different shifts of the local code that are in advance (early branch) and in late (late branch) with respect to the prompt code phase. In this way, three different code phases can be tested in a parallel way. (A single-processor PC can simulate the parallel architecture of a GNSS receiver by sequentially performing a large part of the processing.) Furthermore, the complex wave $\exp\{j2\pi f_i kT\}$ is used to compensate for the Doppler shift. This technique is generally termed *hybrid search* (see also [32, 86]).

The key points of the software implementation are listed below.

(a) The local code is digitized using 1 bit and then stored into a preloaded look-up table. Three pointers are used to indicate early (E), prompt (P), and late (L) code phases. This technique allows any code shift to be translated into an equivalent pointer shift.

(b) The correlation is carried out on words of 32 (or 64) bits. More specifically, the multiplication is implemented by combining XOR and NOT operations as in (15), whereas the sum is equal to the difference $N_1 - N_0$ between the number of bits that are 1 (represented by N_1) and the number of bits that are 0 (indicated by N_0). The final result is a reduction of the input sampling rate from R_s to $R_o = R_s/32$ in case of 32-bit processor (or $R_s/64$ in case of 64-bit processor). To further minimize the processing time, the operation $N_1 - N_0$ (done on words of 32 or 64 bits) can be computed during the initialization of the software receiver and permanently stored in a look-up table.

(c) The correlation output is multiplied by the complex wave $\exp\{j2\pi f_i kT\}$, where $T = 32/R_s$ and f_i denotes the tested frequency sample, and the result is grouped into an array of W samples to be averaged. This step allows the input noise to be attenuated, also reducing the processing rate.

Note that, at the output of the first correlators of the correlation module (before the multiplication with $\exp\{j2\pi f_i kT\}$), the maximum power loss due to unresolved Doppler shift can be evaluated using (13). For instance, focusing on a GPS signal, the worst Doppler shift is $|f_D| = 10\,\mathrm{kHz}$ [6]. The period of correlation is $T_I = T = N_{\mathrm{bit}}/R_s$, where $N_{\mathrm{bit}} = 32$ or 64 bits, and the sampling frequency R_s can be extracted from Table 4. Using $R_s = 5.5\,\mathrm{MHz}$, the loss is

$$\mathrm{PL} \approx \begin{cases} 0.05\,\mathrm{dB} & \text{if } N_{\mathrm{bit}} = 32, \\ 0.19\,\mathrm{dB} & \text{if } N_{\mathrm{bit}} = 64. \end{cases} \tag{16}$$

The PL is low, and it can be neglected in the case of a static receiver, for which $|f_D| \leq 5\,\mathrm{kHz}$ [6].

Concerning Step (c), the value of W should be chosen large enough to minimize the noise contribute. Nevertheless, we should also consider the frequency step used to estimate the Doppler shift, because large values of W can cause strong degradations of C/N_0, due to the presence of a possible residual frequency offset. Again, (13) can be used to evaluate its impact in terms of PL (e.g., let us assume

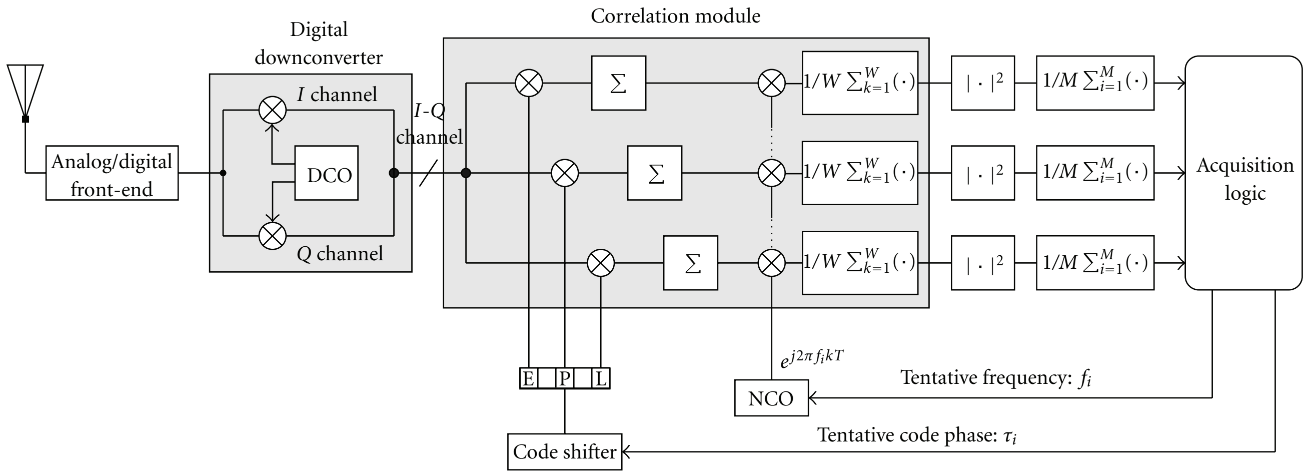


FIGURE 16: Acquisition stage channel of a software receiver.

$N_{\text{bit}} = 32$, $W = 100$ and a frequency step of 1 kHz, the residual is $f_\Delta \leq 500$ Hz and PL ≤ 1.24 dB, while with a frequency step of 500 Hz, the residual is $f_\Delta \leq 250$ Hz and PL ≤ 0.30 dB).

The next modules square the incoming samples, to eliminate the dependence on carrier phase, and average over M samples to filter spurious peaks due to residual noise. The outcome is then compared to a threshold to detect the signal. The acquisition is achieved when the local code replica is roughly aligned with the incoming spreading code and the frequency estimation well approximates the Doppler shift.

An example of the acquisition of a real GPS signal is described below using a GPS/SBAS software receiver [52, 53], that makes use of the approach illustrated above. A static scenario is assumed so that the Doppler shift belongs to the interval $[-5, 5]$ kHz. The GPS PRN-09 signal is detected and the result is plotted in Figure 17. The 3D plot contains the acquisition function as a function of the code timing (sampled at $f_s = 5.5$ MHz) and of the frequency offset. The frequency step is 1 kHz and the code phase step is 1 chip. The maximum peak is achieved at a code shift of about $2500/f_s$(max code phase offset $\leq 1/2$ chip) and roughly at $f_D = 1000$ Hz (max frequency residual ≤ 500 Hz). This rough synchronization can be refined during the tracking stage, as shown in the next section.

To sum up, the algorithm described above is a simple and basic approach to effectively acquire GNSS signals. Nevertheless, it results to be a very time-consuming technique, because it carries out exhaustive searches in time and frequency domain. Therefore, faster algorithms based on Fourier transform are typically used [5, 10, 19, 34, 87]. Like the conventional serial search methods, these acquisition algorithms are based on correlating the incoming signal with a local replica of the received spreading code. However, this is performed in the frequency domain instead of in the time domain. This implies a parallel computation of the correlation function that sensibly reduces the acquisition time, since it eliminates the code phase search, as shown in the remaining part of this section.

In the discrete time domain, the circular cross-correlation definition of two functions, x_n and y_n, is

$$r_k(\delta) = \frac{1}{M} \sum_{n=k}^{M+k-1} x_n \cdot y_{n+\delta}. \tag{17}$$

If the *discrete Fourier transform* (DFT) $\mathcal{F}\{\cdot\}$ is performed on $r_k(\delta)$, and neglecting the normalization factor, we get

$$\begin{aligned} R_k(\ell) &= \mathcal{F}\{r_k(\delta)\} \\ &= \sum_{\delta=0}^{M-1} r_k(\delta) e^{-j2\pi\ell\delta/M} \\ &= \sum_{\delta=0}^{M-1} \left[\sum_{n=k}^{M+k-1} x_n \cdot y_{n+\delta} \right] e^{-j2\pi\ell\delta/M} \\ &= \sum_{n=k}^{M+k-1} x_n \left[\sum_{\delta=0}^{M-1} y_{n+\delta} e^{-j2\pi\ell(n+\delta)/M} \right] e^{+j2\pi\ell n/M} \\ &= Y(\ell) \cdot \sum_{n=k}^{M+k-1} x_n e^{+j2\pi\ell n/M} \\ &= Y(\ell) \cdot X^*(\ell), \end{aligned} \tag{18}$$

where $Y(\ell) = \mathcal{F}\{y_n\}$, $X^*(\ell)$ is the complex conjugate of $X(\ell) = \mathcal{F}\{x_n\}$, and x_n is a real function. Thus, the correlation becomes

$$r_k(\delta) = \mathcal{F}^{-1}\{R_k(\ell)\} = \frac{1}{M} \sum_{\ell=0}^{M-1} R_k(\ell) \cdot e^{+j2\pi\ell\delta/M}, \tag{19}$$

where $\mathcal{F}^{-1}\{\cdot\}$ is the inverse Fourier transform operation.

To apply this result in the acquisition of a GPS signal, it is sufficient to slightly modify (9) and to include the frequency estimation. Thus, (9) becomes

$$r_k(\delta) = \frac{1}{M} \sum_{n=k}^{M+k-1} w_n \cdot e^{j2\pi f_i (n+\delta) T_s} \cdot c_{n+\delta}, \tag{20}$$

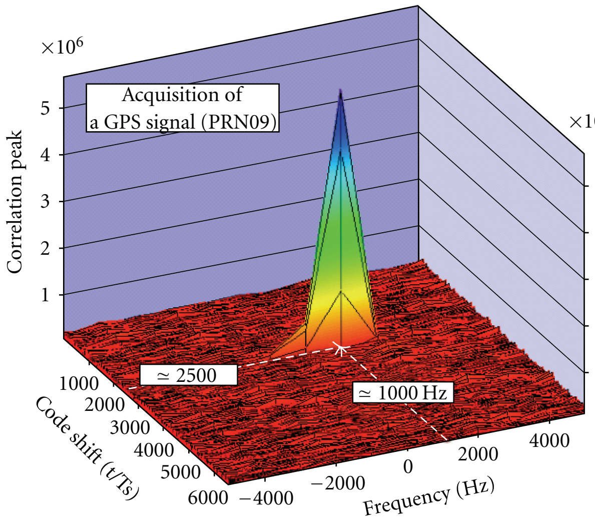


FIGURE 17: Time/frequency acquisition of a GPS signal.

where w_n is the near-baseband incoming complex signal, c_n is the local replica of the received spreading sequence, and $\exp\{j2\pi f_i(n+\delta)T_s\}$ is a complex wave that attempts the Doppler compensation with the tentative frequency f_i. Thus, from (17), we obtain

(i) $y_{n+\delta} = c_{n+\delta} \cdot \exp\{j2\pi f_i(n+\delta)T_s\}$,

(ii) $x_n = w_n$,

and, consequently, we can apply (18) and (19) to parallelize the computation of the correlation function.

The DFT-based acquisition scheme is shown in Figure 18. This picture contains a block diagram of the *parallel code phase search algorithm* [10, 19] and the algorithm is briefly summarized as follows:

(a) a chunk with length 1 ms of the incoming digital signal s_n is downconverted near-baseband (the output is w_n),

(b) this sequence is Fourier-transformed and complex-conjugated (the output is $X^*(\ell)$),

(c) the PRN code $c_{n+\delta}$ (offline generated) is multiplied by the complex wave $\exp\{j2\pi f_i(n+\delta)T_s\}$,

(d) the result is a complex input sequence, to be Fourier-transformed (the output is $Y(\ell)$),

(e) the two sequences are multiplied to compute the correlation in the frequency domain ($R_k(\ell) = X^*(\ell) \cdot Y(\ell)$),

(f) the correlation $R_k(\ell)$ is transformed in the time domain (the output is $r_k(\delta)$),

(g) the maximum of the correlation amplitude, $\max[|r_k(\delta)|^2]$, is compared to the threshold λ,

(h) if the threshold is crossed, the acquisition ends and the code phase is identified by the index δ and the Doppler estimation is f_i,

(i) otherwise, a new f_i is selected and the algorithm restarts from Step (a).

A buffer of 1 ms of signal has been selected because it corresponds to one complete period of the GPS C/A code, so it is a good trade-off between the acquisition performance and computational complexity. (A signal buffer length less than 1 ms involves a correlation with an incomplete code, whereas a length higher than 1 ms increases the computational complexity.)

To give an idea of the difference between the DFT acquisition algorithm and the serial search in terms of acquisition time, it is sufficient to compare the total number of possible acquisition tentatives. This number corresponds to the total number of combinations code phase c_n and Doppler frequency f_i. Let us assume a frequency step of 1 kHz (so, to compensate a Doppler from −10 kHz to 10 kHz, 21 frequency bins are needed) and 1023 different code phases:

(i) in case of serial search the number of combinations is $1023 \cdot 21 = 21483$,

(ii) in case of DFT, all code phases are computed in parallel, so the number of combinations is equal to the number of frequency bins, that is 21.

This basic observation shows that the serial search can be very longer with respect to the DFT acquisition technique.

The reduction of the acquisition time is paid in terms of computational complexity. Indeed, the complexity of the parallel code phase search is clearly higher than the conventional serial search acquisition, because, for each acquisition tentative, it requires the computation of one DFT and one DFT^{-1} (the DFT of the local code can be done once off-line) and, consequently, more sum and product operations than a single correlation in the time domain. It is thus apparent that the computational efficiency of the DFT acquisition algorithm depends on the implementation of the DFT transformations. Thus, efficient algorithms, called FFTs (*fast Fourier transforms*), are commonly used to calculate the DFT and its inverse, [88]. Indeed, the DFT complexity is of order $\mathcal{O}(M^2)$, whereas the FFT complexity is $\mathcal{O}(M \cdot \log_2(M))$ [88]. Thus, the gain in terms of reduction of the computational complexity increases as M increases.

The most common FFT is the Cooley-Tukey algorithm, but some different variants of this algorithm exist. All these methods are based on the same approach that subdivides the initial Fourier transform in a sequence of length M of Fourier transforms which have a smaller and smaller dimension. Therefore, they require input buffers that are integer multiples of $2 : M = 2^p$, where p is an integer. More details on the implementation of FFT algorithms are given in [88].

To conclude, we cite the results reported by [10]. Referring to a (MATLAB implemented) software receiver run on a Pentium 4 2.8 GHz, it is shown that the serial search is about 87 times slower than the FFT acquisition algorithm, but, of course, this last technique bears a higher computational complexity.

4.2.2. Tracking Stage. Though mandatory for the detection and locking of incoming GNSS signals, the rough synchronization achieved by the acquisition stage is not sufficient to directly track the incoming signals and to allow

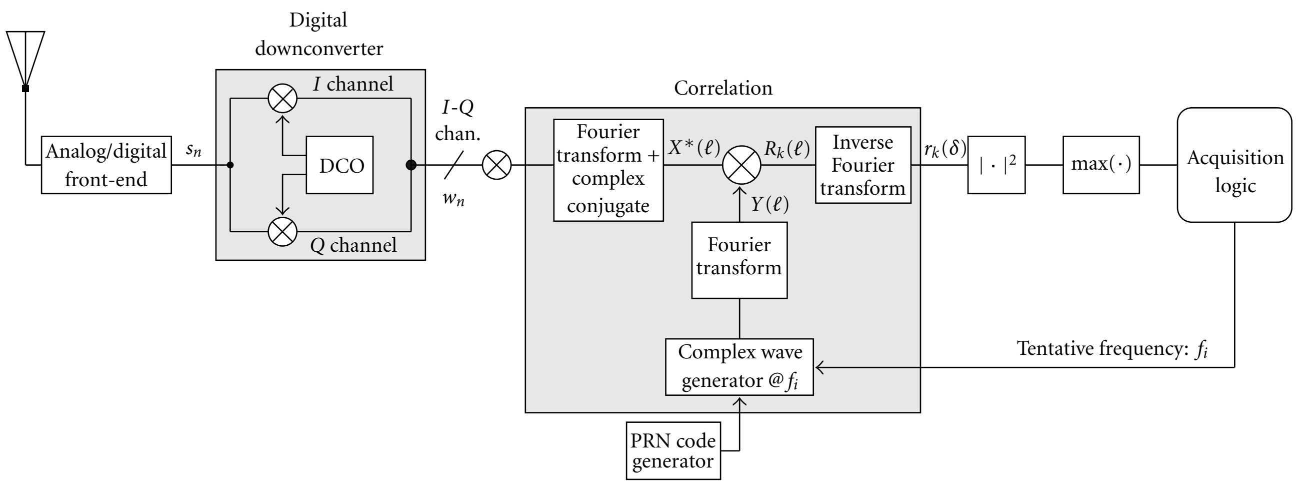


Figure 18: FFT-based acquisition algorithm.

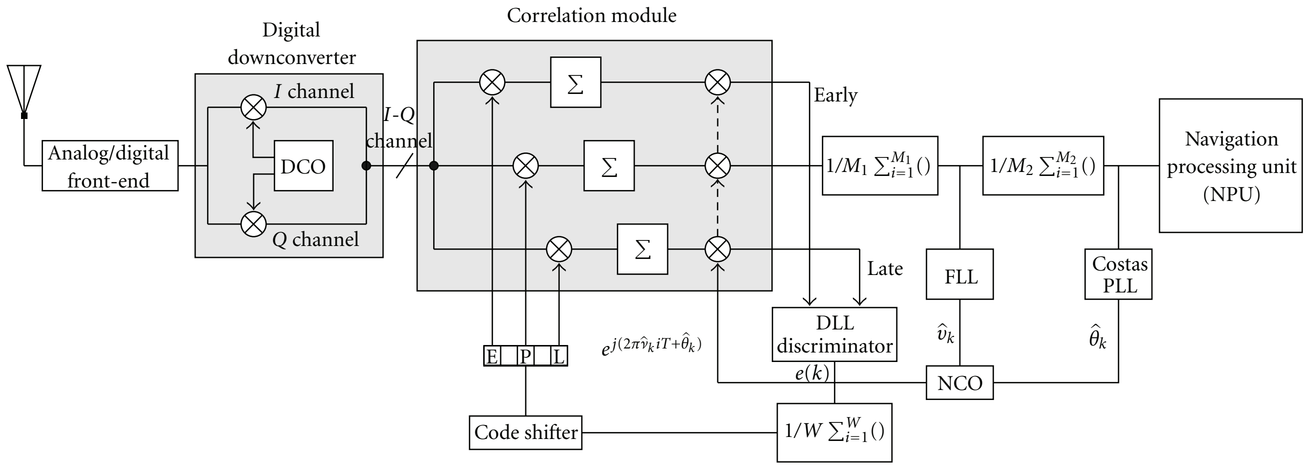


Figure 19: Tracking stage channel of a software receiver.

data processing during the navigation stage. Therefore, the tracking stage, which follows the acquisition, must perform the two important tasks listed below [5, 6, 19, 32, 33].

(a) Refine the synchronization provided by the acquisition stage, up to a fine lock of the received signal.

(b) Maintain the locking of the signal by tracking changes of its code delay and carrier frequency/phase. It is then possible to exploit the entire code gain (by code despreading) to detect and decode the navigation data.

An example of the tracking stage implementation in a real-time software receiver is given in Figure 19 (see also [10, 19]). More in detail, the first part of this architecture replicates the same design of the acquisition stage. The chain, made up of the BB downconverter followed by the correlation module, is composed of a threefold stream of samples, named "*early*" ($E_i = I_{E,i} + j \cdot Q_{E,i}$), "*prompt*" ($P_i = I_{P,i} + j \cdot Q_{P,i}$), and "*late*" ($L_i = I_{L,i} + j \cdot Q_{L,i}$). Early and late samples are used in a timing loop to estimate the received code delay and, finally, to lock the received PRN code. Prompt samples feed frequency and phase loops to finely estimate carrier frequency and phase and to track their changes. This latter stream is also used to detect the GNSS data frames and to decode them for the next navigation stage.

The algorithms used to recover and track the synchronisms of a GNSS signal are listed below:

(i) a DLL refines the code phase estimation, thus allowing the received PRN sequence to be locked and tracked;

(ii) a FLL is used to lock and track the carrier frequency;

(iii) a PLL is used to lock and track the carrier phase.

There exists a large bibliography for these techniques and their theoretical aspects (e.g., [5, 6, 19, 29, 30, 32–34, 89–92]). This section proposes a practical approach, focusing on implementation aspects. The interested reader is referred to the bibliography for the theoretical details.

All the aforementioned units have a similar structure which is displayed in Figure 20. Specifically, an initial error

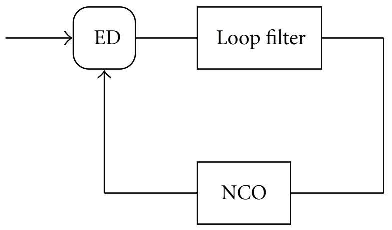


Figure 20: General scheme of a D/F/PLL.

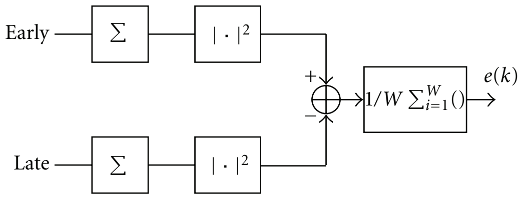


Figure 21: DLL discriminator.

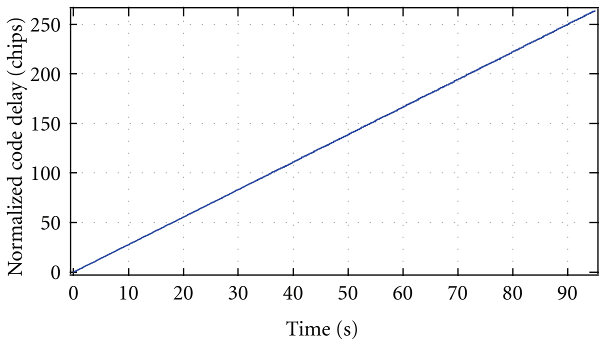


Figure 22: C/A code tracking (GPS PRN-30).

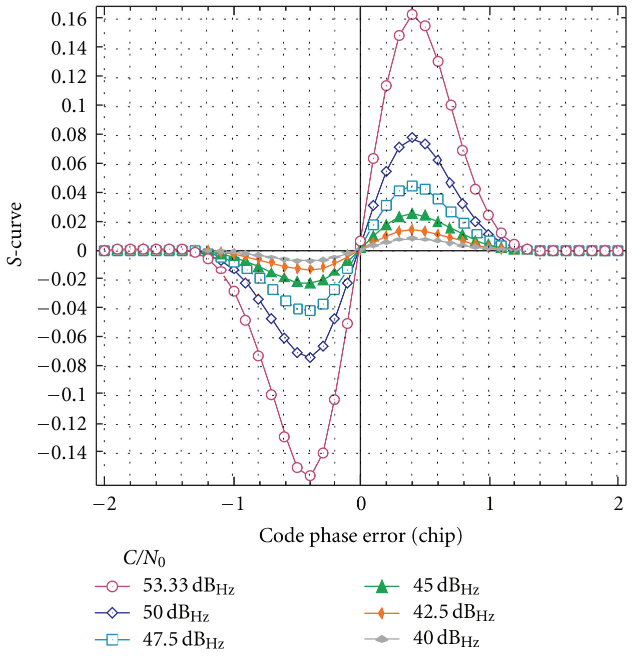


Figure 23: DLL S-curves at different C/N_0 ratios.

discriminator (or *error detector* (ED)) evaluates the offset between the input parameter, to be assessed (e.g., code phase, carrier frequency, etc.), and its estimation. Then, this error feeds the internal loop filter which updates the estimation of the input figure, and, finally, the evaluated parameter is used by an NCO (*numerically controlled oscillator*) or a look-up table to generate a local wave synchronized with the received input reference.

It is important to remark that the error, detected by discriminator, must belong to a specific locking range (which typically depends on the kind of discriminator) to allow the lock of the desired input parameter. This is why the acquisition stage should be designed with a synchronization accuracy compatible with the locking range of DLL and FLL discriminators (i.e., the maximum error that can be recovered). For instance, in case of the DLL, the code misalignment cannot be larger than one chip, or, equivalently, the expected code phase offset should belong to the locking range $[-T_c, +T_c]$, where T_c is the chip time [29].

Regarding the DLL, a very common noncoherent discriminator is the *early-minus-late power* [6, 10, 19, 58] reported below:

$$e_i = |E_i|^2 - |L_i|^2, \tag{21}$$

where the code phase offset, e_i, is the difference of the square modulus of early (E_i) and late (L_i) samples. A block scheme of the DLL discriminator is given in Figure 21. This approach has the advantage to guarantee a timing error that is not sensitive to the data and the carrier phase but just depends on the alignment between the received PRN code and its local replica. Then, the assessed timing error feeds the first-order recursive filter $\tau_{i+1} = \tau_i - \gamma \cdot e_i$ to update the code phase estimation, τ_i. The γ-factor is called *step-size*, and its value should be a trade-off between the rapid convergence of the algorithm and its accuracy. Finally, the estimated code delay is exploited by a code shifter to synchronize the local code phase with that of the incoming sequence.

An example of tracking of a real GPS signal is given in Figure 22. The picture shows the PRN-30 code tracked by the software receiver illustrated in [53] (using the tracking stage scheme reported in Figure 19).

To evaluate the DLL locking range, it is sufficient to run an *open-loop simulation* that evaluates the discriminator response at each alignment between the received PRN sequence and its local replica. The outcome is the S-curve displayed in Figure 23 ([6, 29]). The picture shows six S-curves at different C/N_0 ratios, and, as expected, the locking range is about one chip in advance and late. This means that, if $|e_i| > T_c$, the incoming PRN code cannot be locked.

The design of a first-order FLL is very similar to that of a DLL, but it exploits the rough frequency estimation output by the acquisition stage to track the carrier frequency. Thus, the discriminator (*frequency error detector* (FED)) used

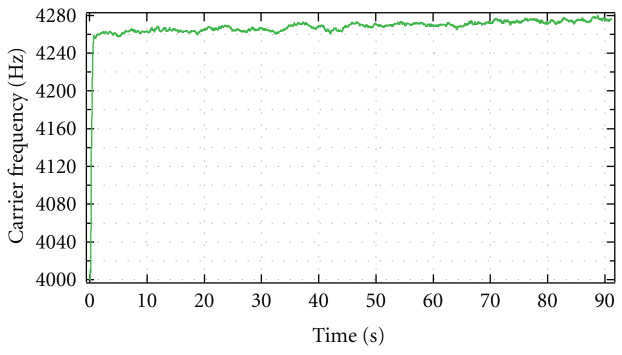


FIGURE 24: Carrier frequency tracking (GPS PRN-30).

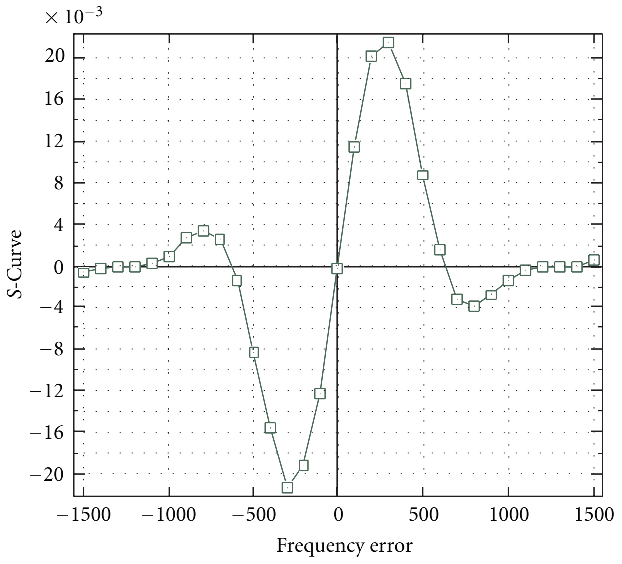


FIGURE 25: FLL S-curve ($C/N_0 = 45\,\mathrm{dB_{HZ}}$).

to detect the frequency offset in the simple case of a BPSK modulation (as GPS signals) is

$$e_i = \Im\{P_i^* \cdot P_{i-1}\}, \tag{22}$$

where P_i is a prompt sample. Of course, the frequency offset feeds the recursive filter $\nu_{i+1} = \nu_i + \gamma \cdot e_i$ to update the assessed carrier frequency, ν_i (γ is the step-size). Then, the NCO, fed by ν_i, generates a wave synchronized with the received carrier frequency. Figure 24 displays an example of tracking for the PRN-30 carrier frequency.

As shown in the DLL, the FED S-curve can be obtained evaluating e_i at different frequency alignments [29]. An example is given in Figure 25 that shows the capability to lock frequency offsets larger than 500 Hz, which is in compliance with maximum frequency error output by the discussed acquisition scheme.

Analogous approaches can be considered for the design of a PLL. The error detector (or *phase error detector* (PED)) in case of a BPSK signal is

$$e_i = \Im\{P_i \cdot \mathrm{sgn}[P_i^*]\}. \tag{23}$$

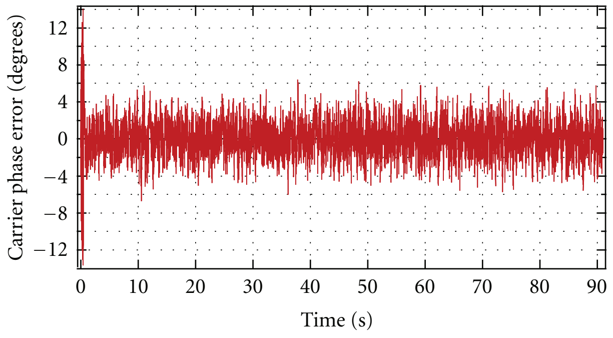


FIGURE 26: Carrier phase tracking (GPS PRN-30).

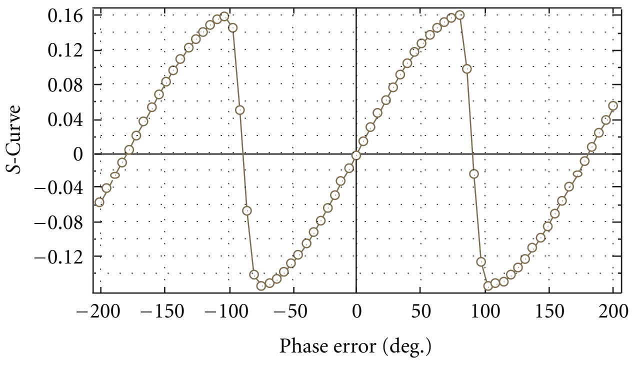


FIGURE 27: PLL S-curve ($C/N_0 = 45\,\mathrm{dB_{HZ}}$).

The error e_i feeds the second-order recursive filter reported below:

$$\begin{aligned}\mu_i &= \mu_{i-1} + \gamma \cdot (1+\rho) \cdot e_i - \gamma \cdot e_{i-1},\\ \theta_{i+1} &= \theta_i + \mu_i,\end{aligned} \tag{24}$$

where both the parameters ρ and γ are related to the loop noise bandwidth and to the damping factor (more details are in [29]). The output is the estimation of the carrier phase, θ_i.

The use of a second-order filter has the advantage to support the tracking of the carrier frequency jointly with the phase.

An example of phase tracking is reported in Figure 26. The picture shows the phase error jitters around 0 degrees with peaks of ±6 degrees. The S-curve is given in Figure 27.

When the tracking steady state is achieved, and after the received data are processed by the Viterbi algorithm for error correction (this is the case of Galileo and SBAS data that are encoded by convolutional codes [11, 15]), the message frame is detected and pseudoranges are computed. Then, navigation algorithms are run to calculate user positions.

4.3. Navigation Stage

The tracking stage is followed by the navigation stage, which is in charge of running ad hoc algorithms to calculate the user position. This implies the evaluation of four unknown parameters, namely, the three components of the receiver position $\mathbf{r}_u$ and the clock bias b, by solving a system of (at least) four equations (one for every tracked satellite). The generic equation relevant to the

ith satellite ($i = 1, 2, \ldots, N_s$, with $N_s \geq 4$) can be expressed as [5, 70]

$$\rho_i = \|\mathbf{r}_i - \mathbf{r}_u\| + c \cdot b + \varepsilon_i, \tag{25}$$

where $\rho_i \approx c(T_{rx} - T_{tx})$ is the pseudorange or, in other words, the coarse distance between the ith satellite and the receiver (the reason for the use of the symbol $\approx$ will be clarified in the following), $(T_{rx} - T_{tx})$ is the difference between the signal reception and the transmission time, $\mathbf{r}_i$ is the satellite position, b is the clock bias, and ε_i is an additive error, whose origin shall be detailed hereafter.

The system (25) can be solved by linearizing the equations with a Taylor expansion around an initial approximate estimate of the user position and by using an iterative technique. In general, the number of equations can be larger than four (as, usually, more than four satellites can be locked), and therefore a least-square solution of the corrections to the initial estimates shall be calculated as illustrated in [5, 19].

Following this approach, the required inputs to compute the user position are the following:

(a) the *ephemerides* are sets of parameters broadcast by the satellite to all the enabled receivers, to let them compute the satellite position according to the Keplerian laws [5, 6];

(b) the *pseudorange*, indicated in (25) as ρ_i, is the raw measure of the distance from a satellite to the receiver (whereas the true satellite-receiver distance is instead denoted as the *range*), which is affected by many error sources ε_i, such as satellite clock instability, receiver noise, multipath, atmospheric delays, and ephemeris mismodelling, that severely degrade its precision [5, 6, 70];

(c) the *pseudorange corrections* (typically broadcast by the satellite itself or by an additional augmented system, such as SBAS and LAAS) are used to compensate the measurement errors, yielding finer pseudoranges to calculate a more precise user position [5, 6].

Ephemerides and pseudorange corrections, together with other essential data (e.g., transmission time, almanacs, satellite health, etc.), are included in the satellite navigation message broadcast by each satellite. These messages are structured in subframes of binary data, each one made of a preamble, a subframe ID, a data field, and a parity check field (for more details, see the *GPS interface control document*, (ICD) [7]). This kind of arrangement allows a simple search and a reliable detection of the received navigation subframe, by using very standard algorithms. Details on the implementation of these algorithms and on the data decoding are strictly related to the design of the desired navigation message and, therefore, are reported in the related ICDs (see [7] for GPS, [11] for Galileo, [15] for SBAS, and [8] for GLONASS).

The approach used to compute pseudoranges turns out to be quite elaborate. Actually, from a purely theoretical point of view, assuming a free-space transmission, the pseudorange is simply the product of the speed of light by the difference between the reception time T_{rx} and the transmission time T_{tx} (which is derived from the received message). In the case of an SDR receiver, there is no absolute time reference to evaluate the reception time, and the sampling frequency is the only reference. Thus, the approach to be pursued consists in evaluating, through a time measurement, a "relative pseudorange" which is the distance between two reference points. Let us consider the case of a GPS signal. At the receiver side, we are interested in identifying the beginning of the frame of the message, assumed to be the reference instant and tagged as "subframe 1." Apart from the satellite clock bias, such beginning point is actually transmitted by all the satellites simultaneously, since there exists a control segment that keeps the satellite transmitters aligned. Anyway, the subframe 1 is received at different instants for each satellite, due to the different distance between the transmitter and the receiver. Thus, by measuring the time difference between a couple of received signals, the receiver can obtain the distance difference from the two relevant satellites. A pictorial representation of this approach is given in Figure 28, where Δ_j, Δ_n, and Δ_m are the relative pseudoranges of PRN-j, PRN-n, and PRN-m with respect to PRN-i. Note that the maximum time distance between any couple of GPS satellites is less than 19 ms [19], as follows from constellation and orbital parameters. This is much shorter than a GPS subframe period, equal to 30 s [6, 7, 19]. Hence, setting a time window of 20 ms at the receiver allows those simultaneously transmitted subframes 1 to be discriminated.

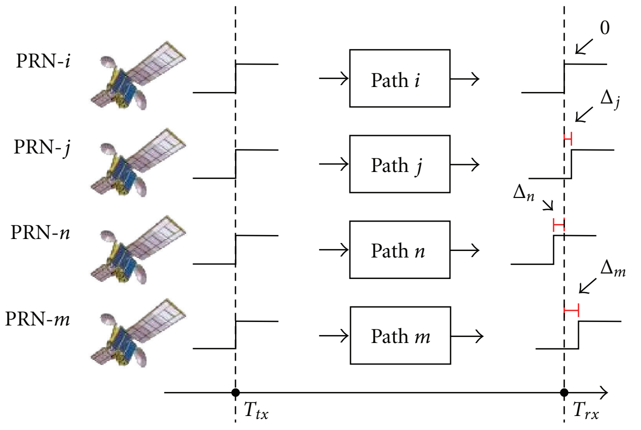


FIGURE 28: Relative pseudoranges.

A relative pseudorange is computed by measuring the time to detect the reference point (e.g., the beginning of the subframe 1) from the input digitized data. To this aim, an SDR receiver must consider the following three contributes [19]:

(a) the initial code phase,

(b) the beginning of the first navigation data,

(c) the beginning of the reference subframe 1.

Thus, still adhering to the example of a GPS receiver, an ideal scheme of the proposed technique is sketched in

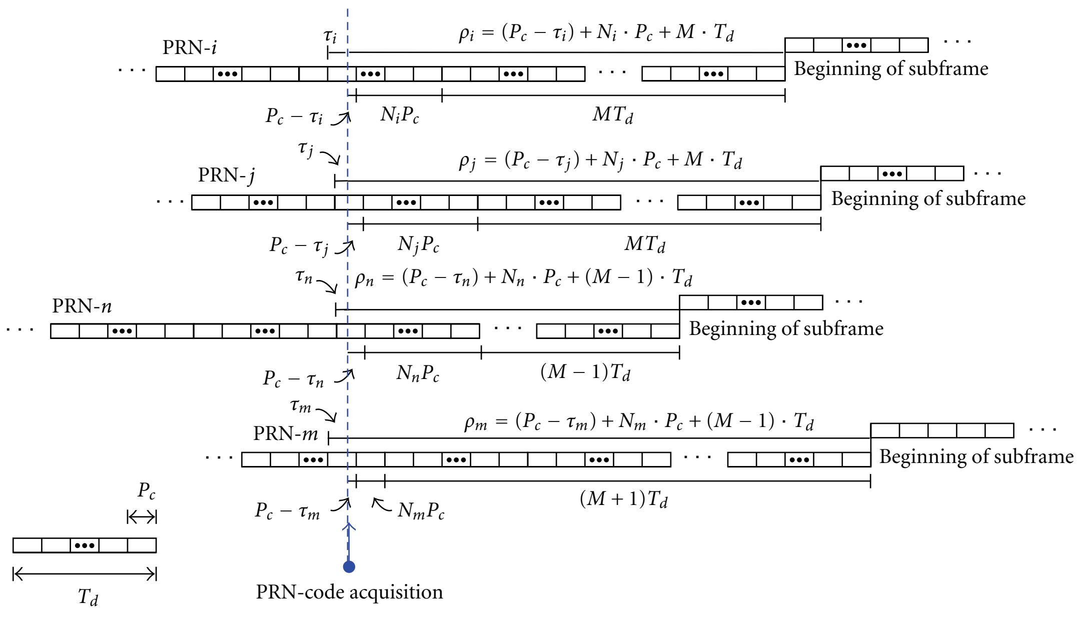


Figure 29: Computation of relative pseudoranges.

Figure 29, where P_c is the duration (period) of a PRN-code and T_d is the data duration. The reference point is marked by a step. Thus, referring to the PRN-i ranging signal, at the end of the acquisition, the tracking stage finely synchronizes the local code with the received one and, at the same time, provides the initial code phase τ_i. After N_i code periods, the beginning of the first data field is obtained, and, finally, after M data intervals, the beginning of subframe 1 is found (checking some control parameters). The same method is used to calculate the time to get the reference point of all other locked signals, and the differences with respect to the PRN-i yield the relative pseudoranges. This approach can also produce negative pseudoranges, so that any constant selected from 67 to 86 ms (which is the range of the time delay of a GPS satellite to the receiver) is further added, in order to make all pseudoranges positive.

The initial pseudorange can be computed by using the raw synchronization provided by the acquisition stage which measures the code phase to the sample resolution [10, 19]. This means that, for example, when $f_s = 5.5$ MHz, the accuracy is about $c/f_s \approx 54.5$ m, whereas the accuracy becomes 7.5 m when $f_s = 40$ MHz. Of course, it is desirable to improve the initial code phase accuracy, because it has a direct effect on the final position accuracy. This result is achieved during the tracking stage, because the initial rough estimation of the code phase is refined by the DLL unit. Indeed, the outputs of the correlation module (early and late) are used by the DLL to accurately determine the code phase residual of the incoming signal and to better synchronize the local replica. Consequently, the refined estimation of the code phase can be also used to improve the pseudorange accuracy, and the remaining measurement noise can be significantly reduced by averaging (e.g., see Figure 21).

Coming back to the navigation stage, the overall scheme is given in Figure 30, which depicts one channel for every tracked satellite. Each raw pseudorange is preliminarily refined by compensating both the iono- and the tropodelays, the satellite clock bias, the relativistic effects, and the group delay. Then, each corrected pseudorange is ready to be used to compute the user position by solving the N_s-equation system. An example of a position error plot measured by a GPS/EGNOS software receiver is shown in Figure 31 [53].

Further details can be found in the huge bibliography addressing navigation algorithms from the subframe detection to the user position computation (e.g., [5, 6, 10, 19]).

Position accuracy is traditionally measured by multiplying the standard deviation of the pseudorange error σ_ρ by a factor that is a function of the satellite geometry PDOP (*position dilution of precision*), yielding the RMS value of the position error ε_{RMS} as [5, 6, 70]

$$\varepsilon_{\text{RMS}} = \sigma_\rho \cdot \text{PDOP}. \qquad (26)$$

High PDOPs indicate poor satellite geometry that implies low position accuracy. The PDOP evaluation algorithm is well illustrated in [5, 6, 70].

5. Conclusions

In the evolution process of communication systems, the high *configurability* and *flexibility*, together with the simplicity to *update/upgrade*, are determining the success of the emerging SDR technology. An evident example is given by the evolution of the GNSS scenario, in which new satellite systems and innovative techniquesshould coexist and cooperate with

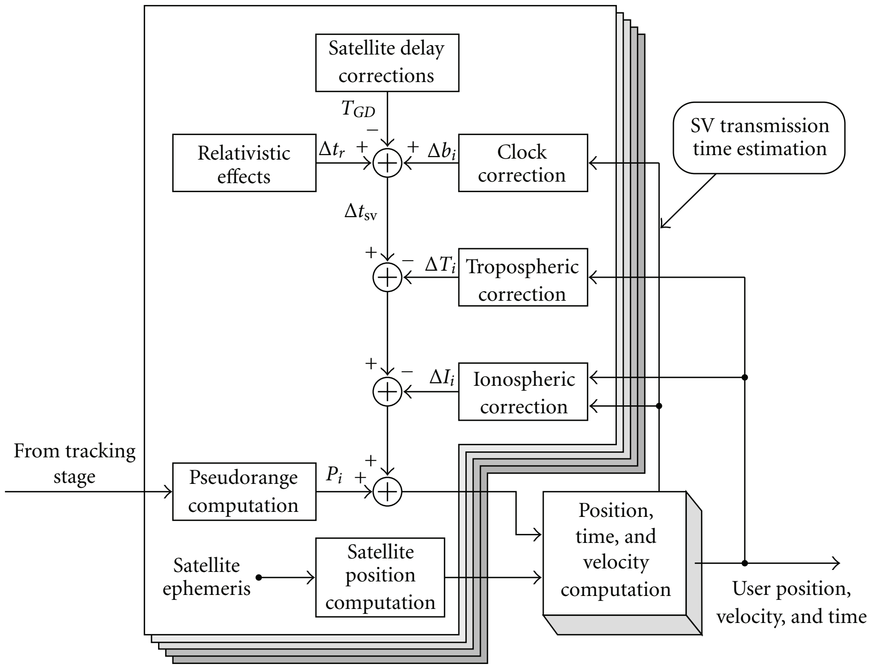


FIGURE 30: Navigation stage.

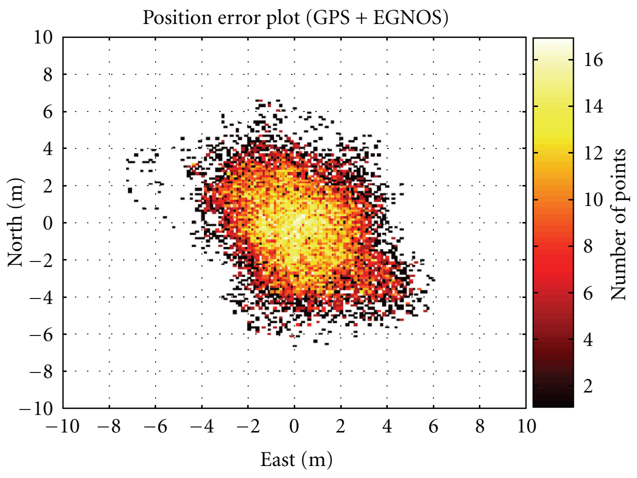


FIGURE 31: An example of a horizontal plot.

the preexisting systems. In this context, traditional hardware receivers are characterized by clear limitations in terms of configurability, flexibility, and capacity to be upgraded. The SDR paradigm, supported by the increasing computational power of new processors, seems to be the right answer to these limitations.

This tutorial gives an overlook to the SDR world through the window opened by the GNSS evolution. To this aim, the overall architecture of a basic software GNSS receiver is illustrated in all its aspects. The proposed analysis illustrates the following key issues of the SDR technology:

(i) implementation solutions (SW-controlled HW, hybrid HW/SW, and fully SW architectures),

(ii) operation flow of a GNSS receiver,

(iii) software architecture (multilayer structure with the managing, MMI, and operating layers),

(iv) front-end architecture and sampling frequency selection,

(v) signal processing and navigation algorithms.

The final result is a simple methodology to design and implement a low-cost GNSS SDR receiver.

Future directions of research will focus on designing high-performance GNSS SDR receivers for specific applications, such as monitoring of positioning signals in a local area, safety of life applications, or aircraft navigation. The reason lies in the huge versatility of the SDR paradigm, which makes this technology very attractive for a large field of commercial applications.

List of Acronyms

ADC:	Analog-to-digital converter
AGC:	Automatic gain control
AltBOC:	Alternative BOC
ASIC:	Application-specific integrated circuit
AWGN:	Additive white Gaussian noise
BB:	Baseband
BER:	Bit error rate
BOC:	Binary offset carrier
BM:	Base model
CBOC:	Composite BOC
CDMA:	Code division multiple access
CEP:	Circular error probability
COTS:	Commodity off-the-shelf
CNR:	Code-to-noise ratio
CPU:	Central processing unit
D/F/PLL:	Delay/frequency/Phase-locked loop
DGPS:	Differential GPS
DSP:	Digital signal processor
E.g.:	Exempli gratia
EGNOS:	European Geostationary Navigation Overlay System
FDMA:	Frequency division multiple access
FPGA:	Field programmable gate array
FE:	Front-end
F/PED:	Frequency/phase error detector
FFT:	Fast Fourier transform
GBAS:	Ground-based augmentation system
GEO:	Geostationary earth orbit
GIS:	Geographic information system
GLONASS:	Global Navigation Satellite System
GNSS:	Global Navigation Satellite Systems
GPS:	Global Positioning System
GSO:	Geosynchronous orbit satellite
HW:	Hardware
ICD:	Interface control document
IF:	Intermediate frequency
LAAS:	Local Area Augmentation System
LNA:	Low-noise amplifier
L/RHCP:	Left/right hand circularly polarized
MBOC:	Multiplexed BOC
MEO:	Medium Earth Orbit
ML:	Managing layer
MMI:	Man-machine interface
MSAS:	Multi-Functional Satellite Augmentation System
N-GSO:	Nongeosynchronous orbit satellite
NRZ:	Non-return-to-zero
OL:	Operating layer
PC:	Personal computer
PDOP:	Position dilution of precision
PL:	Power loss
PRN:	Pseudorandom noise
RF:	Radio frequency
RMS:	Root mean square
SA:	Selective availability
SBAS:	Satellite-based augmentation system
SDR:	Software-defined radio
SIS:	Signals in space
SNR:	Signal-to-noise ratio
SoL:	Safety of life
SR:	Software radio
SV:	Space vehicle
SW:	Software
TDMA:	Time division multiple access
USB:	Universal serial bus
WAAS:	Wide area augmentation system.

Acknowledgment

The authors wish to acknowledge the activity of the Network of Excellence in Wireless COMmunications NEWCOM++ of the European Commission (contract no. 216715) that motivated this work.

References

[1] E. Buracchini, "Software radio concept," *IEEE Communications Magazine*, vol. 38, no. 9, pp. 138–143, 2000.

[2] J. Mitola III and Z. Zvonar, *Software Radio Technologies: Selected Readings*, IEEE Press, New York, NY, USA, 2001.

[3] J. Mitola, "Software radio architecture," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, 1995.

[4] W. Tuttlebee, *Software Defined Radio: Enabling Technologies*, John Wiley & Sons, New York, NY, USA, 2002.

[5] E. D. Kaplan and C. J. Hegarty, *Understanding GPS: Principles and Applications*, Artech House, London, UK, 2nd edition, 2006.

[6] B. W. Parkinson and J. J. Spilker, *Global Positioning Systems: Theory and Applications*, American Institute of Aeronautics and Astronautics, Washington, DC, USA, 1996.

[7] NAVSTAR GPS Space Segment/Navigation User Interface, ICD-GPS-200 Rev. C-005R1, ARINC Research Corporation, El Segundo, Calif, USA, 2003.

[8] "Global Navigation Satellite System Interface Control Document," GLONASS ICD v. 5.0, Moscow, Russia, 2002.

[9] G. Gibbons, "Galileo's new era—step by step the program advances," *Inside GNSS*, vol. 3, no. 7, 2008.

[10] K. Borre, D. Akos, N. Bertelsen, P. Rinder, and S. Jensen, *A Software- Defined GPS and Galileo Receiver: A Single-Frequency Approach*, Birkhäauser, Boston, Mass, USA, 2007.

[11] "Galileo open service, Signal in space interface control document," OSSIS-ICD d. 1, European Space Agency/European GNSS Supervisory Authority, Brussels, Belgium, 2008.

[12] G. X. Gao, A. Chen, S. Lo, D. De Lorenzo, and P. Enge, "GNSS over China: the compass MEO satellite codes," *Inside GNSS*, vol. 2, no. 5, 2007.

[13] F. Boon, F. Wilms, J.-M. Sleewaegen, and W. De Wilde, "More compass points tracking China's MEO satellites on a hardware receiver," *Inside GNSS*, vol. 2, no. 5, 2007.

[14] T. Grelier, J. Dantepal, A. de Latour, A. Ghion, and L. Ries, "Compass MEO satellite signals: initial observations and analysis," *Inside GNSS*, vol. 2, no. 4, 2007.

[15] "Minimum operational performance standards for Global Positioning System/Wide Area Augmentation System airborne equipment," RTCA/DO 229C, RTCA Inc., Washington, DC, USA, 2001.

[16] O. Julien, P. Alves, M. E. Cannon, and G. Lachapelle, "Triple frequency GPS/GALILEO for ambiguity resolution," *European Journal of Navigation*, vol. 2, no. 2, pp. 51–57, 2004.

[17] A. Smimsky, “Three’s the charm. Triple-frequency combinations in future GNSS,” *Inside GNSS*, vol. 1, no. 5, 2006.

[18] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global Positioning Systems, Inertial Navigation, and Integration*, John Wiley & Sons, New York, NY, USA, 2001.

[19] J. B.-Y. Tsui, *Fundamentals of GPS Receivers: A Software Approach*, John Wiley & Sons, New York, NY, USA, 2005.

[20] G. Hein, J.-H. Won, and T. Pany, “GNSS software defined radio: real receiver or just a tool for experts?” *Inside GNSS*, vol. 1, no. 5, 2006.

[21] F. Macchi and M. G. Petovello, “Development of a one channel Galileo L1 software receiver and testing using real data,” in *Proceedings of the 20th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS ’07)*, vol. 2, pp. 2256–2269, Fort Worth, Tex, USA, 2007.

[22] S. A. Nik and M. Petovello, “Multichannel dual frequency GLONASS software receiver,” in *Proceedings of the 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS ’08)*, pp. 614–624, September 2008.

[23] M. G. Petovello, C. O’Driscoll, G. Lachapelle, D. Borio, and H. Murtaza, “Architecture and benefits of an advanced GNSS software receiver,” *Positioning*, vol. 1, no. 1, pp. 66–78, 2009.

[24] M. Fantino, A. Molino, and M. Nicola, “N-Gene: a complete GPS and Galileo software suite for precise navigation,” in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ITM ’10)*, vol. 2, pp. 1245–1251, San Diego, Calif, USA, 2010.

[25] D. F. M. Cristaldi, D. Margaria, and L. Lo Presti, “A multi-frequency low-cost architecture for GNSS software receivers,” in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ITM ’10)*, vol. 2, pp. 849–857, San Diego, Calif, USA, 2010.

[26] C. J. Hegarty and E. Chatre, “Evolution of the global navigation satellite system (GNSS),” *Proceedings of the IEEE*, vol. 96, no. 12, pp. 1902–1917, 2008.

[27] J. Börjesson, *GLONASS contributions to space geodesy*, Ph.D. dissertation, Chalmers University of Technology, Göteborg, Sweden, 2000.

[28] Y.-S. Huang and M.-L. Tsai, “The impact of compass/Beidou-2 on future GNSS: a perspective from Asia,” in *Proceedings of the 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS ’08)*, vol. 2, pp. 675–686, Savannah, Ga, USA, 2008.

[29] U. Mengali and A. N. D’Andrea, *Synchronization Techniques for Digital Receivers*, Plenum Press, New York, NY, USA, 1997.

[30] J. G. Proakis, *Digital Communications*, McGraw-Hill, New York, NY, USA, 4th edition, 2001.

[31] G. D. Forney Jr., “The viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[32] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications Handbook*, McGraw-Hill, New York, NY, USA, 2002.

[33] J. K. Holmes, *Spread-Spectrum Systems for GNSS and Wireless Communications*, Artech House, London, UK, 2007.

[34] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, “Theory of spreadspectrum communications—a tutorial,” *IEEE Transactions on Communications*, vol. 30, no. 5, pp. 855–884, 1982.

[35] G. Gibbons, “GLONASS—a new look for the 21st century,” *Inside GNSS*, vol. 3, no. 4, 1982.

[36] News Update, “China adds details to Compass (Beidou II) signal plans,” *Inside GNSS*, vol. 3, no. 6, 2008.

[37] G. W. Hein, J.-A. Avila-Rodriguez, S. Wallner et al., “MBOC: the new optimized spreading modulation recommended for GALILEO L1 OS and GPS L1C,” in *Proceedings of the IEEE Position Location and Navigation Symposium*, pp. 883–892, San Diego, Calif, USA, 2006.

[38] E. M. Wolmarans and A. J. Truter, “Software radio: implementation aspects,” in *Proceedings of the EUROCOMM*, Munich, Germany, 2000.

[39] B. Sauriol and R. Landry, “FPGA-based architecture for high throughput, flexible and compact real-time GNSS software defined receiver,” in *Proceedings of the Institute of Navigation National Technical Meeting (NTM ’07)*, pp. 708–717, San Diego, Calif, USA, January 2007.

[40] G. Bacci, M. Calamia, L. Foglia, F. Longo, and P. Mulassano, “SWAN: software systems for navigation applications—an Italian project,” in *Proceedings of the GNSS Vulnerabilities and Solutions Conference*, Baška, Croatia, 2008.

[41] S. Peng and B. M. Ledvina, “A real-time software receiver for the GLONASS L1 signal,” in *Proceedings of the 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS ’08)*, vol. 2, pp. 1152–1163, 2008.

[42] J. Raasakka, H. Hurskainen, T. Paakki, and J. Nurmi, “Modeling multi-core software GNSS receiver with real time SW receiver,” in *Proceedings of the 22nd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS ’09)*, vol. 1, pp. 468–473, Savannah, Ga, USA, 2009.

[43] D. A. Godsoe, M. E. Kaye, and R. B. Langley, “A framework for real-time GNSS software receiver research,” in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ITM ’10)*, vol. 2, pp. 892–905, San Diego, Calif, USA, 2010.

[44] A. Knežević, C. O’Driscoll, and G. Lachapelle, “Co-processor aiding for real-time software GNSS receivers,” in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ITM ’10)*, vol. 2, pp. 837–848, San Diego, Calif, USA, 2010.

[45] S. Jeon, H. So, H. Noa, T. Lee, and C. Kee, “Development of realtime software GPS receiver using Windows Visual C++ and USB RF front-end,” in *Proceedings of the International Technical Meeting of the Institute of Navigation*, San Diego, Calif, USA, 2010.

[46] D. M. Akos, *A software radio approach to GNSS receiver design*, Ph.D. dissertation, Ohio University, Athens, Ohio, USA, 1997.

[47] B. M. Ledvina, S. P. Powell, P. M. Kintner, and M. L. Psiaki, “A 12-channel real-time GPS L1 software receiver,” in *Proceedings of theInstitute of Navigation National Technical Meeting (ION NTM ’03)*, Anaheim, Calif, USA, 2003.

[48] C. Ma, G. Lachapelle, and M. E. Cannon, “Implementation of a software GPS receiver,” in *Proceedings of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS ’04)*, pp. 956–970, Long Beach, Calif, USA, 2004.

[49] B. M. Ledvina, M. L. Psiaki, D. J. Sheinfeld, A. P. Cerruti, S. P. Powell, and P. M. Kintner, “A real-time GPS civilian L1/L2 software receiver,” in *Proceedings of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS ’04)*, pp. 986–1005, Long Beach, Calif, USA, September 2004.

[50] P. Rinder and N. Bertelsen, *Design of a single frequency GPS software receiver*, M.S. thesis, Aalborg University, Aalborg, Denmark, 2004.

[51] G. MacGougan, P.-L. Normark, and C. Ståhlberg, "Attitude determination using a real-time GPS/Galileo software receiver," in *Proceedings of the ESA Workshop on Satellite Navigation User Equipment Technologies (NAVITEC '04)*, Noordwijk, The Netherlands, 2004.

[52] F. Principe, C. Terzi, M. Luise, and M. Casucci, "SOFT-REC: a lowcost GPS receiver following the software radio paradigm," in *Proceedings of the ESA Workshop on Satellite Navigation User Equipment Technologies (NAVITEC '04)*, Noordwijk, The Netherlands, 2004.

[53] G. Bacci, F. Principe, M. Luise, C. Terzi, and M. Casucci, "SOFTREC: a GPS real-time software receiver with EGNOS augmentation," in *Proceedings of the ESA Workshop on EGNOS Performance and Applications*, Gdynia, Poland, 2005.

[54] P.-L. Normark and C. Ståhlberg, "Hybrid GPS/Galileo real time software receiver," in *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS '05)*, vol. 2005, pp. 1906–1913, Long Beach, Calif, USA, 2005.

[55] P. Crosta, G. A. Vecchione, and L. Marradi, "A real-time GALILEO/GPS L1 software receiver," in *Proceedings of the ESA Workshop on Satellite Navigation User Equipment Technologies (NAVITEC '06)*, Noordwijk, The Netherlands, 2006.

[56] A. Di Cintio, O. Montenbruck, L. L. Presti, R. Lesca, and E. Guyader, "The GREHDA project: Galileo software receiver for high dynamic applications," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS '07)*, vol. 2, pp. 2376–2387, 2007.

[57] M. S. Braasch and A. J. Van Dierendonck, "GPS receiver architectures and measurements," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 48–64, 1999.

[58] P. Fenton, B. Falkenberg, T. Ford, K. Ng, and A. J. V. Dierendonck, "Novatel's GPS receiver the high performance OEM sensor of the future," in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '91)*, Albuquerque, NM, USA, 1991.

[59] L. Dong, C. Ma, and G. Lachapelle, "Implementation and verification of a software-based if GPS signal simulator," in *Proceedings of the Institute of Navigation National Meeting (NTM '04)*, pp. 378–389, San Diego, Calif, USA, January 2004.

[60] B. M. Ledvina, M. L. Psiaki, T. E. Humphreys, S. P. Powell, and P. M. Kintner, "A real-time software receiver for the GPS and Galileo L1 signals," in *Proceedings of the 19th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '06)*, pp. 2321–2333, Fort Worth, Tex, USA, September 2006.

[61] E. Vinande and D. Akos, "Improvements to "a software-defined GPS and Galileo receiver: Single-frequency approach"," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS '07)*, vol. 2, pp. 2230–2236, Fort Worth, Tex, USA, 2007.

[62] M. Anghileri, T. Pany, D. S. Güixens et al., "Performance evaluation of a multi-frequency GPS/Galileo/SBAS software receiver," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS '07)*, pp. 2749–2761, Fort Worth, Tex, USA, September 2007.

[63] A. Di Cintio, O. Montenbruck, L. L. Presti, R. Lesca, and E. Guyader, "The GREHDA project: Galileo software receiver for high dynamic applications," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS '07)*, vol. 2, pp. 2376–2387, Fort Worth, Tex, USA, 2007.

[64] C. O'Driscoll, M. G. Petovello, and G. Lachapelle, "Software receiver strategies for the acquisition and re-acquisition of weak GPS signals," in *Proceedings of the National Technical Meeting*, vol. 2, pp. 843–854, San Diego, Calif, USA, 2008.

[65] M. G. Petovello and G. Lachapelle, "Centimeter-level positioning using an efficient new baseband mixing and despreading method for software GNSS receivers," *Eurasip Journal on Advances in Signal Processing*, vol. 2008, Article ID 194276, 12 pages, 2008.

[66] K. Muthuraman, "Theoretical bounds and reliable C/N0 estimation for modernized GPS signals," in *Proceedings of the 22nd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '09)*, vol. 6, pp. 3730–3740, Savannah, Ga, USA, 2009.

[67] K. Muthuraman, D. Borio, R. Klukas, and G. Lachapelle, "Adaptive data/pilot carrier phase tracking for modernized GNSS signals," in *Proceedings of the 22nd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '10)*, San Diego, Calif, USA, 2010.

[68] P. B. Kenington, *RF and Baseband Techniques for Software Defined Radio*, Artech House, London, UK, 2005.

[69] D. M. Akos and J. B. Y. Tsui, "Design and implementation of a direct digitization GPS receiver front end," *IEEE Transactions on Microwave Theory and Techniques*, vol. 44, no. 12, pp. 2334–2339, 1996.

[70] P. Misra, B. P. Burke, and M. M. Pratt, "GPS performance in navigation," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 65–85, 1999.

[71] J. M. Kelly and M. S. Braasch, "Validation of theoretical GPS multipath bias characteristics," in *Proceedings of the IEEE Aerospace Conference*, pp. 31317–31325, Big Sky, Mont, USA, March 2001.

[72] M. S. Braasch, "Performance comparison of multipath mitigating receiver architectures," in *Proceedings of the IEEE Aerospace Conference*, pp. 31309–31315, Big Sky, Mont, USA, March 2001.

[73] J. M. Kelly, M. S. Braasch, and M. F. DiBenedetto, "Characterization of the effects of high multipath phase rates in GPS," *GPS Solutions*, vol. 7, no. 1, 2003.

[74] Novatel website, L1/L2 Antennas, http://www.novatel.com/Documents/Papers/L1L2antennas.pdf.

[75] F. Piazza and Q. Huang, "A 1.57-GHz RF front-end for triple conversion GPS receiver," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 2, pp. 202–209, 1998.

[76] A. Gersho, "Principles of quantization," *IEEE Transactions on Circuits and Systems*, vol. 25, no. 7, pp. 427–436, 1978.

[77] C. J. Hegarty, "Analytical model for GNSS receiver implementation losses," in *Proceedings of the 22nd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '09)*, vol. 6, pp. 3205–3218, Savannah, Ga, USA, 2009.

[78] F. Bastide, D. Akos, C. Macabiau, and B. Roturier, "Automatic gain control (AGC) as an interference assessment tool," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS '03)*, Portland, Ore, USA, 2003.

[79] R. G. Vaughan, N. L. Scott, and D. R. White, "The theory of bandpass sampling," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 1973–1984, 1991.

[80] A. Polydoros and C. L. Weber, "A unified approach to serial search spread-spectrum code acquisition—part I: general theory," *IEEE Transactions on Communications*, vol. 32, no. 5, pp. 542–549, 1984.

[81] A. Polydoros and C. L. Weber, "A unified approach to serial search spread-spectrum code acquisition—part II: a matched filter receiver," *IEEE Transactions on Communications*, vol. 32, no. 5, pp. 550–560, 1984.

[82] S. S. Rappaport and D. M. Grieco, "Spread-spectrum signal acquisiton: methods and technology," *IEEE Communications Magazine*, vol. 22, no. 6, pp. 6–21, 1984.

[83] C. J. Kim and H. J. Lee, "Adaptive acquisition of PN sequences for dsss communications," *IEEE Transactions on Communications*, vol. 46, no. 8, pp. 993–996, 1998.

[84] S. G. Glisic, "Automatic decision threshold level control (ADTLC) in direct-sequence spread-spectrum systems based on matched filtering," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 519–527, 1988.

[85] S. Chung, "New serial search acquisition approach with automatic decision threshold control," in *Proceedings of the IEEE 45th Vehicular Technology Conference*, pp. 530–536, Chicago, Ill, USA, July 1995.

[86] W. Zhuang, "Noncoherent hybrid parallel PN code acquisition for CDMA mobile communications," *IEEE Transactions on Vehicular Technology*, vol. 45, no. 4, pp. 643–656, 1996.

[87] R. A. Scholtz, "Multiple access with time-hopping impulse modulation," in *Proceedings of the IEEE Military Communications Conference (MILCOM '93)*, pp. 447–450, Boston, Mass, USA, 1993.

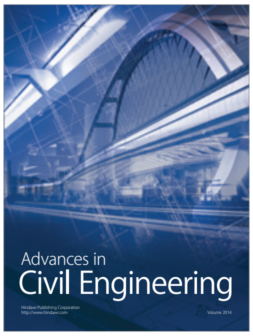
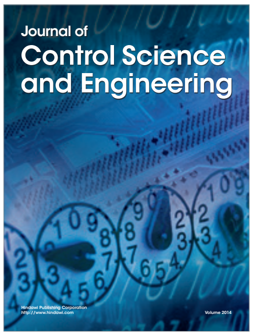
[88] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1999.

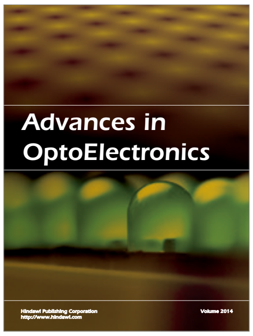
[89] R. De Gaudenzi, M. Luise, and R. Viola, "Digital chip timing recovery loop for band-limited direct sequence spread-spectrum signals," *IEEE Transactions on Communications*, vol. 41, no. 11, pp. 1760–1769, 1993.

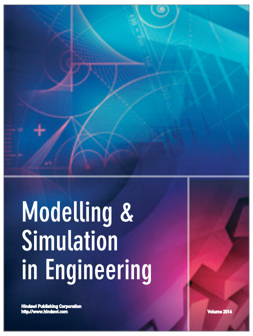
[90] F. Gardner, *Phaselock Techniques*, John Wiley & Sons, New York, NY, USA, 3rd edition, 2005.

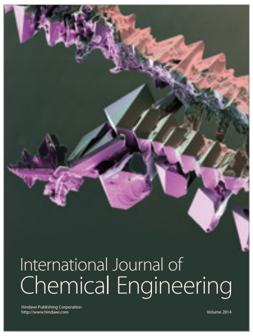
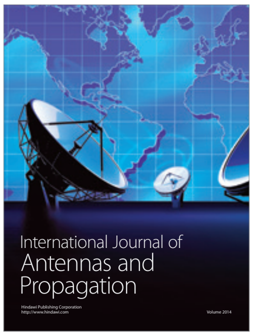
[91] A. N. D'Andrea, U. Mengali, and R. Reggiannini, "Modified Cramer-Rao bound and its application to synchronization problems," *IEEE Transactions on Communications*, vol. 42, no. 2, pp. 1391–1399, 1994.

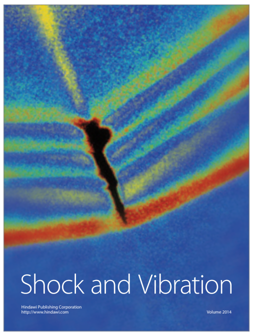
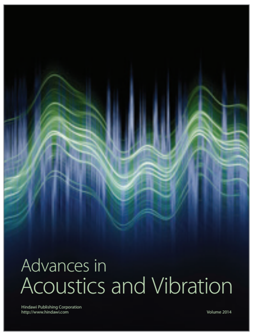
[92] G. C. Hsieh and J. C. Hung, "Phase-locked loop techniques—a survey," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 6, pp. 609–615, 1996.





Hindawi

Submit your manuscripts at
http://www.hindawi.com