SOFT-REC: A LOW-COST GPS RECEIVER FOLLOWING THE SOFTWARE RADIO PARADIGM

Fabio Principe [1] , Carla Terzi [2], Marco Luise [1], and Marco Casucci [2]

[1]*Dipartimento Ingegneria dell'Informazione*
University of Pisa
Via G. Caruso, 56126 Pisa (Italy)
Email: marco.luise@iet.unipi.it, fabio.principe@iet.unipi.it

[2]*INTECS s.p.a.*
Via U. Forti, 5 Loc. Ospedaletto
56121 Pisa (Italy)
Email: terzi@pisa.intecs.it, casucci@pisa.intecs.it

ABSTRACT

This contribution describes an innovative approach to the design of a low-cost GPS/EGNOS receiver. In particular, the architecture of the receiver follows the well-know paradigm of the software radio. Currently, a typical GPS receiver, ranging from the hand-held version to the more sophisticated ones, uses a dedicated hardware circuit to implement all the signal processing algorithms. The SOFT-REC (stands for software-receiver) completely changes this approach, transferring all the ad-hoc hardware processing into a set of software algorithms running in real-time on a standard high-end PC integrated with a COTS (Commodity Off The Shelf) L1 frequency GPS receiver digital front-end.

1. INTRODUCTION

We describe in this paper the implementation of a low-cost GPS/EGNOS software receiver intended for land vehicles (trains). We remark that our receiver is fully re-programmable and re-configurable [2], allowing complete adaptability to several cases, from standard applications to high-performance ones. The receiver architecture can be also regarded as a suited baseline to come to a fully reprogrammable GPS/GALILEO receiver. An analog/digital front-end and a standard PC Unit are the only hardware parts that we need to realize the GPS/EGNOS SOFT-REC (software-receiver).

After this introduction, the *second section* of the paper illustrates the hardware implementation of the SOFT-REC, focusing the attention on the front-end features. The *third section* gives an idea of the real-time software architecture, its features and performances. In the *fourth section* we give a general overview of our signal processing library, showing some simulation results, and final conclusions are presented in the *fifth section.*

2. SOFT-REC GENERAL ARCHITECTURE

The overall architectural scheme for the SOFT-REC prototype is shown in Fig.1. As is seen, the receiver is made up of a GPS/EGNOS antenna, an analog/digital front-end and a Personal Computer (PC). The analog/digital front-end performs a down-conversion from Radio Frequency (RF) to an Intermediate Frequency (IF) and then samples the IF signal on the L_1 carrier. After A/D conversion, the PC receives via a USB port a digital signal that can be processed obtaining GPS/EGNOS data which may be used by a Navigation Library to estimate the user position.

Fig. 2 depicts a detailed scheme of the front-end. The RF filter selects L_1 and cuts off the L_2 carrier. The subsequent mixer performs down conversion from RF to IF of the L_1 carrier. At the output we obtain a bandpass signal at the IF of 15.42 MHz (f_{IF}). The local oscillator frequency (f_{LO}) is thus $f_{LO} = f_{RF} - f_{IF} = 1560$ MHz .

The IF filter has a bandwidth of 2 MHz (B_{IF}). The programmable ADC performs the signal sampling and digital binary coding. To find out the correct value for the sampling frequency (f_S), we observe that from the Band Sampling Theorem, we must have

$$\begin{cases} k \cdot f_S \geq 2 \cdot f_{IF} + B_{IF} \\ (k-1) \cdot f_S < 2 \cdot f_{IF} - B_{IF} \end{cases} \Rightarrow \quad \frac{32.84}{k} < \frac{28.84}{k-1} \quad \Rightarrow \quad k < 8.21 \tag{1}$$

where k is an integer. The resulting ranges for the sampling frequency are shown in Tab. Table 1. If we choose $k = 6$ with 1 bit quantization, we obtain a good trade off between accuracy and the resulting data rate to be handled by the PC. With one-bit quantization and f_s = 5.5 MHz, we get a data rate of 5.5 Mbps that can be easily transferred via a standard USB 1.0 port. Also, the number of samples per chip is $f_S / R_c \approx 5.38$ that is more than adequate for our signal processing functions. We could easily find a COTS GPS front-end complying with these parameters (Accord's "GPS Signal Tap"), but we explicitly remark here that the design of a custom front-end like the one in Fig. 2 is straightforward and very low-cost.

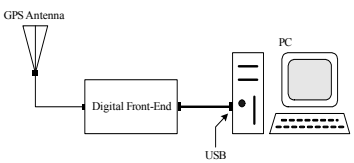

Fig.1. SOFT-REC Hardware Architecture

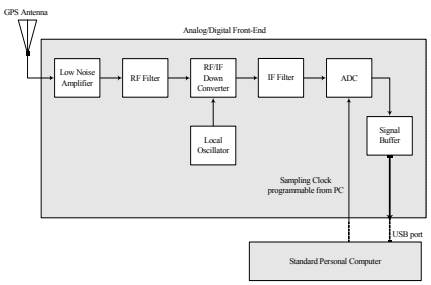

Fig. 2. Analog/Digital Front-End Scheme

Table 1. f_S range.

k	Min f_S (MHz)	Max f_S (MHz)
1	32.84	∞
2	16.42	28.84
3	10.947	14.42
4	8.21	9.6133
5	6.568	7.21
6	5.4733	5.768
7	4.6914	4.8067
8	4.105	4.12

3. SOFTWARE DESCRIPTION

In addition to the standard functions of Man-Machine Interface (MMI) and general housekeeping, the SOFTREC software section has to perform all real-time functions related to signal processing. In particular, the parallel processing of GPS/EGNOS signals implies the management and elaboration of great amounts of data in real time then an underlying management layer, that is constituted by the Operating System primitives, is needed. In order to support the real-time behavior of the receiver, a real-time operating system has been chosen, i.e. RTLinux. Regarding the development tools, good performances were attained by using the C programming language compiled with early versions of *gcc*. The MMI was on the contrary implemented by Tcl/Tk language.

3.1. RTLinux

Real-Time Linux (RTLinux) is a small hard real-time kernel that can run the Linux kernel as its lowest priority thread. RTLinux extends the standard UNIX programming environment to real-time problems. From the programmer's point of view, RTLinux adds a special real-time process to Linux and allows programmers to insert real-time threads and signal handlers into that process. The real-time software can talk to ordinary Linux processes using RT-FIFOs (which are like ordinary pipes), shared memory and signals.

3.2. SOFT-REC software architecture

A high level presentation of the architecture of the SOFT-REC is represented in Fig. 3. The yellow bullets represent software modules, and the green boxes represent data exchange buffers. The GPS/EGNOS Digital Front End is illustrated as an orange box.

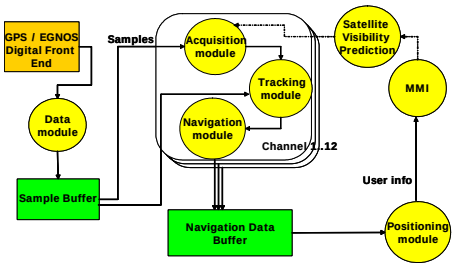

Fig. 3. Software Architecture

Basically, the architecture can be described as a multithreaded and concurrent execution of the channel signal processing algorithms along with the modules responsible for the sampled data acquisition, the MMI, the positioning and the module implementing the satellite visibility prediction, that is the software that calculates the visible satellites starting from the provided rough user position. In the following some details are provided for each module in the figure.

3.2.1. Data Module

The Data Module is in charge of acquiring the samples from the digital front-end into main memory. It has to be synchronized with the rate at which the front-end outputs its data, in order not to lose any of the incoming samples. This component also exports the routines to read or write data samples buffer with bit-wise resolution.

3.2.2. Channel Module

The Channel Module is responsible for implementing the GPS-EGNOS channels signal processing algorithms. It is the core of the SOFT-REC application, and certainly the heaviest part of the system as regards the computational burden. In optimal conditions, up to 12 GPS channels plus 2 EGNOS channels shall be tracked, and therefore the real-time properties of the channel modules have to be precisely defined and implemented, as regards the real-time interaction with the various data exchange buffers and the time and computational power allocation.

3.2.3. Acquisition Module

The Acquisition module is in charge of finding the signal of a certain satellite. It operates on a block of samples to extract the C/A code start and the carrier Doppler frequency, information used subsequently by the tracking software in order to demodulate the signal.

3.2.4. Tracking module

The Tracking Module is responsible for demodulating the GPS-EGNOS signal. It uses the data provided by the Acquisition module in order to track the signal of a certain satellite and the variations of some of its parameters, such as the variation of the carrier frequency due to the Doppler effect.

3.2.5. Navigation Module

This component implements Navigation procedures, on the basis of outputs coming from Tracking algorithms. The relevant subframes are processed and ephemeris data is extracted as well as clock and satellite health information.

3.2.6. Positioning

The Positioning Module calculates an estimation of the user position, given the data provided by the Navigation Modules, read from the Navigation Data Buffer. R/T FIFO queues are opened towards the user-space MMI graphical interface as communication resources.

3.2.7. MMI Module

The MMI is responsible for displaying all the SOFT-REC output and system state data to the user. Moreover, it is in charge of acquiring the user inputs and commands. This component runs at user level, using the time slices left by R/T tasks. It periodically reads from R/T FIFO queues provided by Positioning module and shows information about the current SOFT-REC operational mode.

3.2.8. Satellite Visibility Prediction Module

This component also runs in user-space and provides the routines needed to evaluate expected satellites in visibility and a coarse estimation of the Doppler shift basing on the rough current user position. After having executed (successfully or not in determining the needed information) it hands over the control to RTLinux module which is in charge of initializing the whole software tasks.

3.2.9. Software Start-up and Modules Characterization

As above mentioned, regarding the Operating System characterization, the signal processing components have to be implemented as RTLinux modules i.e. object files containing routines and resources, whereas the MMI and the Prediction of Visible Satellites have not strict real-time constraints and they can be run under Linux user space. The latter are implemented as common executable files. In particular, at start-up time the correct module activation order has to be respected. After the Satellite Visibility Prediction has executed an initialization module takes the control. The system start-up policy is to load each module into the kernel, before starting it. When loading a module, its public resources are accessible to the others, including, where applicable, the code of the thread implementing the peculiar module functionality. In this way, loading modules does not imply starting real-time threads. The initialization component creates all threads referring to the relevant public code and starts all of them assigning the right order and priority.

4. SIGNAL PROCESSING LIBRARY

All signal processing functions needed to perform user position estimation were implemented through real-time C procedures integrated in the general software described in the previous section. To be more specific, we will start from a description of the baseband equivalent of the IF received signal:

$$s_{L_1}(t+\tau) = \mathrm{Re}\left\{ s_{BB}(t+\tau) \cdot e^{-j\left[2\pi(f_{IF}+f_D)\cdot(t+\tau)+\phi\right]} \right\} + n(t) = \\ = \mathrm{Re}\left\{ \left[r_R(t+\tau) + j \cdot r_I(t+\tau) \right] \cdot e^{-j\left[2\pi(f_{IF}+f_D)t+\theta\right]} \right\} + n(t) \tag{2}$$

where $\theta = 2\pi(f_{IF} + f_D)\tau + \phi$, $n(t)$ is AWGN (Additive White Gaussian Noise), τ is the propagation delay f_D, is the carrier Doppler Shift, and $s_{BB}(t)$ is the (filtered, baseband) GPS signal. The main cause of the Doppler Shift is satellite motion. As in [1]-[2], the carrier Doppler Frequency range ranges from -5000 Hz to 5000 Hz, while the C/A code Doppler Shift is about 3.2 Hz.

We now give a short description of our signal processing library. In particular we describe the two main stages, namely signal acquisition and tracking, and we discuss the main related processing algorithms that we implemented.

4.1. Signal Acquisition

When the SOFT-REC receiver is turned on, it runs an algorithm for the prediction of satellite in visibility based on a very rough indication of the receiver position. In this way it is possible to load the right codes and to have a coarse estimation of the Doppler Shift, avoiding an exhaustive search during the acquisition stage in the time and frequency domains.

When the prediction phase is over, coarse code acquisition stage is started. Fig. 4 depicts the processing architecture of a single SOFT-REC channel (each channel is identified by its own C/A code). From this picture it is easy to recognize the following modules.

4.1.1. Prediction of Satellites in Visibility

This satellite prediction algorithm is run when the SOFT-REC is turned on. The user inputs its coarse position, date and time, and the visible satellites are identified. This is done using stored almanacs than can be periodically downloaded from the Internet.

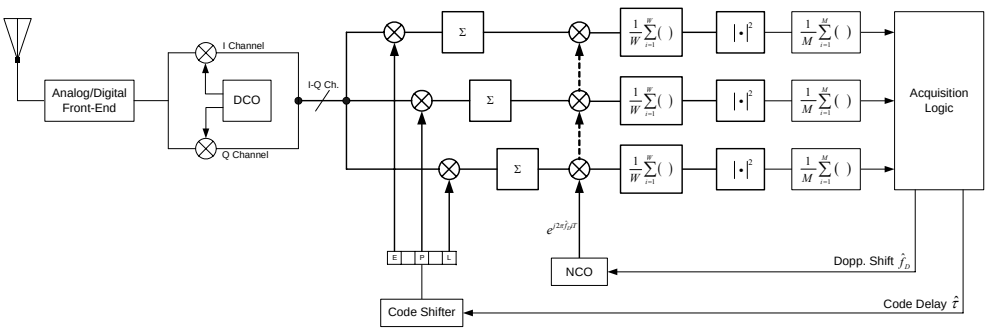

Fig. 4. Acquisition Stage

4.1.2. Baseband Module

This module implements a down-conversion from IF to near-baseband of the digital signal. From (2) and remembering that the front-end output is a sampled and 1 bit quantized signal, the baseband down-converter implements

$$\begin{cases} r_R'(kT_s) = \overline{\mathrm{u}\left[s_{IF}(kT_s+\tau)\right] \otimes \mathrm{u}\left[\cos(2\pi f_{IF} kT_s)\right]} \\ r_I'(kT_s) = \overline{\mathrm{u}\left[s_{IF}(kT_s+\tau)\right] \otimes \mathrm{u}\left[\sin(2\pi f_{IF} kT_s)\right]} \end{cases} \tag{3}$$

where $k = 0,1,2,\ldots$, $\otimes$ is XOR operator, T_s is sampling time and

$$\mathrm{u}[\mathrm{k}] = \begin{cases} 1, & k \geq 0 \\ 0, & k < 0 \end{cases}. \tag{4}$$

4.1.3. Despreading/Accumulation Module

In this stage we perform despreading with ranging code and a preliminary accumulation. In particular we have three branches, with a shifted code replica each: the early, prompt and late (e/p/l) versions. The baseband received signal is thus correlated with these three replicas on an accumulation (integration) period equal to 16 or 32 samples. This preliminary integration allows to decrease the processing rate of the signal with basically no impairments due to Doppler shift.

4.1.4. Bidimensional Serial Code Acquisition Module

The last module is a serial code acquisition unit [5]. This unit performs triple correlation of the received signal with the e/p/l codes as above, but also performs Doppler Shift precompensation with a 1 kHz step, allowing a two-dimensional search: in the time and frequency domains. This allows to hand over to the tracking stage with a residual Doppler Shift smaller than 500 Hz. We show in Fig. 5 a pictorial representation of the bidimensional search result in the form of a plot of the correlation results as a function of time and frequency (delay and Doppler shift).

4.2. Tracking Stage

When the acquisition stage is over, tracking phase is executed. The channel architecture changes its structure, it is shown in Fig. 6. Practically in this stage the receiver has to track the C/A code and to recovery frequency residual offset and carrier phase. These operations are implemented respectively by a 1st order DLL (Delay Locked Loop) [3] and FLL (Frequency Locked Loop) [4] and 2nd order PLL (Phase Locked Loop) [4].

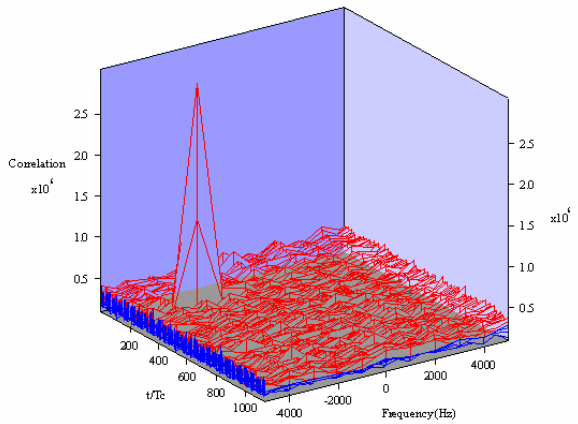

Fig. 5. Bidimensional Coarse Code Acquisition

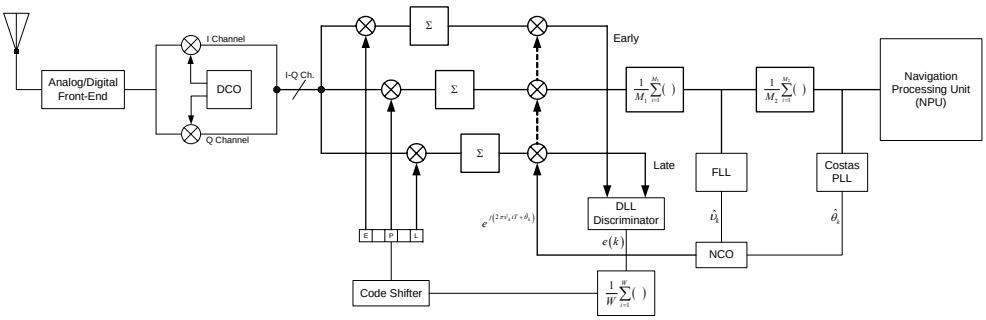

Fig. 6. Tracking Stage.

4.2.1. Delay Locked Loop (DLL)

We used a standard 1st order non-coherent early-late DLL, whose sample output is shown in Fig. 7.

4.2.2. Carrier Frequency Locked Loop (FLL)/ Phase-Locked Loop (PLL)

This module implements a 2nd order PLL aided by a 1st order FLL. In particular this unit estimates and compensates for the residual Doppler Shift, and phase-locks onto the carrier phase to perform coherent data detection of the BPSK navigation data. The frequency loop is pretty standard, with a digital frequency discriminator given by

$$e(k) = \mathrm{Im}\left\{x_k^* \cdot x_{k-1}\right\} \tag{5}$$

Fig. 8 shows an example of frequency estimation with 1 Hz Loop Bandwidth (B_L).
The carrier phase locking loop is the standard 2nd order decision-aided Costas PLL shown in Fig. 9. The phase error is computed as follows:

$$e(n) = w_I(n) \cdot \mathrm{sgn}\left[w_Q(n) \right] \tag{6}$$

where $w_I(n)$ and $w_Q(n)$ are the in-phase and quadrature components of $w(n)$, respectively.

4.2.3. Navigation Library

During tracking, this library allows to decode the GPS subframe and to perform navigation. In particular, the user position is estimated through a standard iterative minimum mean-square error solution of the positioning equations starting from the pseudorange measurements combined with corrective parameters taken from GPS ephemeris.

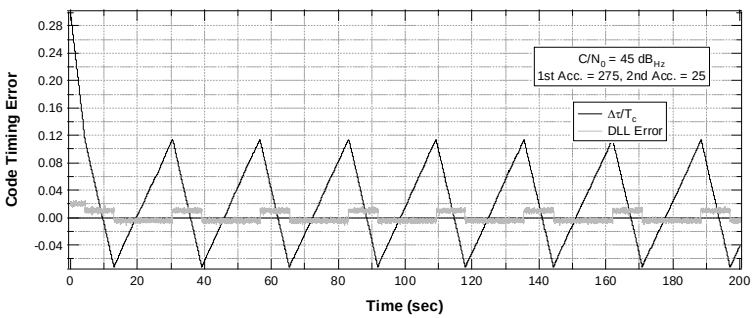

Fig. 7. Code Timing Estimation Sample

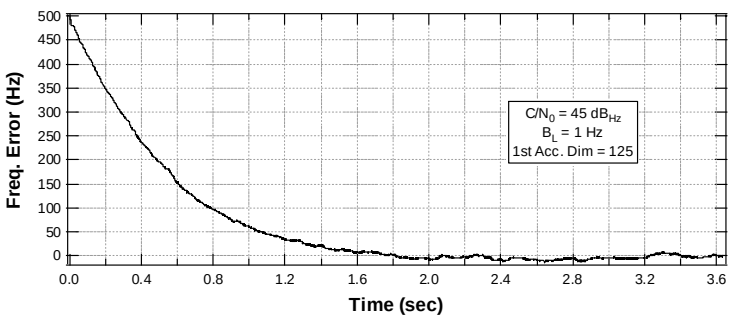

Fig. 8. Carrier Frequency Acquisition Sample

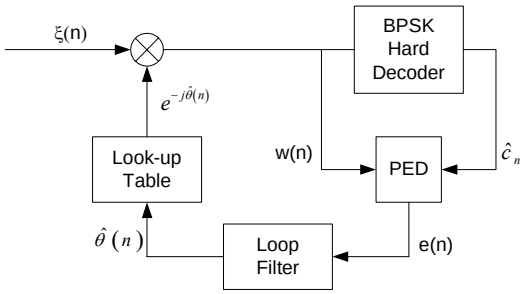

Fig. 9. Carrier PLL Architecture

5. CONCLUSIONS

The aim of this paper is showing how to implement a low-cost GPS/EGNOS Software Receiver, with characteristic of full re-programmability and re-configurability. The analog front-end used here is just an off-the-shelf component intended for research, but what was shown here is that the overall design is actually independent of the front-end characteristics. Rather, the design focus was on the architecture and the optimization of the real-time software to perform the different signal processing functions for positioning. Custom realization of the front-end would lead to a very low-cost implementation of the whole receiver, and this design approach paves the way for the extension of the receiver to GALILEO signals.

REFERENCES

[1] E. Kaplan, “Understanding GPS: Principles and Applications”, *Artech House*, 1996.

[2] J. Bao-Yen Tsui, “Fundamentals of Global Positioning System Receivers a Software Approach”, *Wiley-Interscience*, 2000.

[3] R. De Gaudenzi, M. Luise, R. Viola, “A Digital Chip Timing Recovery Loop for Band-Limited Direct-Sequence Spread-Spectrum Signals”, *IEEE Transactions on Communications*, Vol. 41, No. 11, November 1993.

[4] U. Mengali, A. N. D'Andrea, "Synchronization Techniques for Digital Receivers (Applications of Communications Theory)", *Plenum Publishing Corporation*, 01 November, 1997

[5] M.K. Simon, J. K. Omura, R. A. Scholtz, B. K. Levitt, “Spread-Spectrum Communications Handbook”, *McGraw-Hill*, 1994.