SOFT-REC: a GPS/EGNOS Software Receiver

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*—In this paper, we describe a new approach to the design of an innovative GPS/EGNOS receiver. In particular, the architecture of our receiver follows the well-know paradigm of the software radio. Currently, typical GPS receivers use a dedicated hardware circuit to implement all the signal processing algorithms. The SOFT-REC (stands for software-receiver) changes this approach, transferring all the 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) L_1 frequency GPS receiver digital front-end.**

I. Introduction

With this paper we want to show the implementation of a new GPS/EGNOS receiver intended for land vehicles (as trains). We remark that our receiver is completely software and so fully re-programmable and re-configurable, allowing to obtain a complete adaptability to several cases: from standard applications to high-performance ones. The only hardware parts that we need to build the SOFT-REC (software-receiver) are an analog/digital front-end and a standard PC Unit.

After the introduction, the *second section* of the paper illustrates the hardware implementation of the SOFT-REC, focusing the attention on the front-end characteristics. The *third section* gives an idea of the real-time software architecture with a short description of its main stages. In the *fourth section* we give a general overview of our signal processing library, showing some simulation results obtained with real GPS signal. Finally the conclusions are presented in the *fifth section.*

II. General Architecture

The overall architectural scheme for the SOFT-REC prototype is shown in Fig.1.

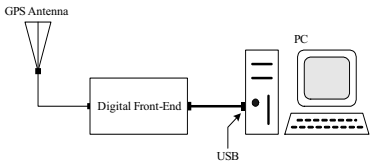

Fig.1. SOFT-REC Hardware Architecture.

As seen, the receiver is made up of a standard 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 the ADC (Analog/Digital Converter), the PC receives via a USB port a digital signal that can be processed obtaining GPS/EGNOS data which may be used by the navigation algorithms to estimate the user position.

A detailed scheme of front-end is depicted in Fig. 2. The RF filter selects L_1 carrier (f_{RF} = 1575.42 MHz) and cuts off the L_2 one (f_{RF} = 1227.6 MHz). The subsequent mixer performs down conversion from RF to IF of the L_1 carrier. Hence we obtain a bandpass signal at the IF of 15.42 MHz (f_{IF}). The local oscillator frequency (f_{LO}) is:

$$f_{LO} = f_{RF} - f_{IF} = 1560 \text{ MHz}.$$

At this point, an IF filter gets in input this signal and cuts off all the spectral components out of its bandwidth (B_{IF} = 2 MHz) and soon after a programmable ADC performs the signal sampling and its quantization.

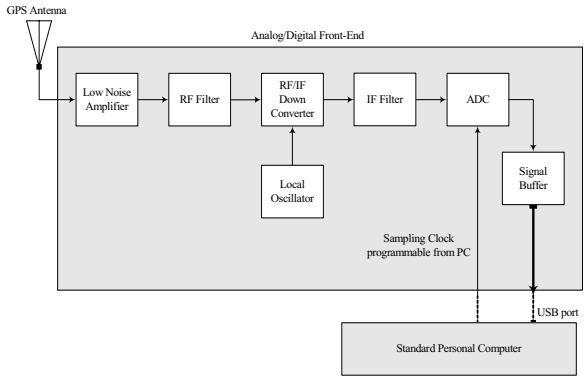

Fig. 2. Analog/Digital Front-End Scheme.

To find out the correct value for the sampling frequency (f_s), we observe that from the Band Sampling Theorem, we must have:

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

where k is an integer. The resulting ranges for the sampling frequency are shown in Table I . 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.

TABLE I
SAMPLING FREQUENCY RANGES

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

III. SOFTWARE DESIGN

In this stage we try to give an idea of the architecture of our software receiver. In particular we will focus the attention on the used techniques to implement it in real-time, describing all its main stages too.

First of all we remark that the SOFT-REC 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 and 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 (Real-Time Linux). Regarding the development tools, it is possible to attain good performances by using the C programming language compiled with early versions of *gcc*.

A. RTLinux

Real-Time Linux (RTLinux) is a small hard real-time kernel that can run the Linux kernel as its lowest priority thread. It 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.

B. 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 analog/digital front-end is illustrated as an orange box.

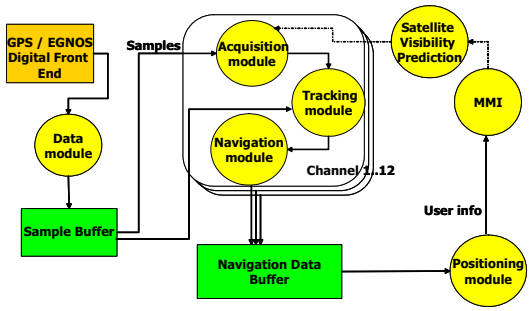

Fig. 3. General Software Architecture.

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. In the following some details are provided for each module in the figure.

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.

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 the heaviest part of the system as regards the computational burden. In optimal conditions, up to 12 GPS channels plus 1 EGNOS channel shall be tracked, and therefore the real-time properties of the channel modules have to be precisely defined and implemented.

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 by the tracking stage in order to demodulate the signal.

4) Tracking module

The Tracking Module is responsible for demodulating and detecting of GPS/EGNOS subframes. It uses the rough information provided by the acquisition module in order to track the signal of a certain satellite.

5) Navigation Module

This component implements Navigation procedures, on the

basis of outputs coming from Tracking algorithms. The relevant subframes are processed, the pseudoranges are measured and ephemeris data is extracted.

6) Positioning

The Positioning Module calculates an estimation of the user position, using the data provided by the Navigation Modules, read from the Navigation Data Buffer.

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. It is implemented by Tcl/Tk language.

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, it hands over the control to RTLinux module which is in charge of initializing the whole software tasks.

9) Software Start-up and Modules Characterization

As above mentioned, 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.

IV. Signal Processing Algorithms

All signal processing functions needed to detect GPS/EGNOS signals and 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:

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

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 for land vehicles. These considerations are very important for the right implementation of the SOFT-REC architecture.

Now we give a description of our signal processing algorithms. In particular we can identify two main stages, called Acquisition and Tracking Stage. The first one is dedicated to the rough acquisition of GPS/EGNOS signals, while, during the tracking stage, it is possible to recover all the signal synchronisms (as code timing, Doppler shift, and carrier phase) and then decode GPS/EGONOS subframes to evaluate user position.

A. 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 obtained form the satellite almanacs. 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.

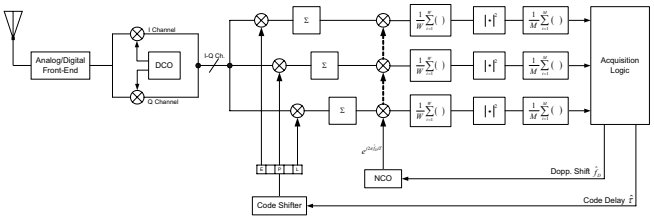

Fig. 4. Acquisition Stage Channel.

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. At this point it is possible to identify the visible satellites. This operation is done using stored almanacs that can be periodically downloaded from Internet or from satellites during receiver functioning.

2) Baseband Module

Soon after the RF/IF down-conversion and the signal sampling, the Baseband Module is inserted. This unit 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 performs:

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

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

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

Using the expression reported in (3) it is possible to executed several bit operations in parallel way reducing the processing time (i.e. 16 or 32 bit operations per PC clock tick).

3) Despreading/Accumulation Module

In this stage despreading with ranging code and a preliminary accumulation are executed. In particular, as Fig. 4 shows, we have three branches, each one with its own shifted code replica: called early, prompt and late (e/p/l) versions. The baseband received signal is thus correlated with these three replicas and then accumulated (integrated) on a 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) 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 pre-compensation 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 have to remark that, performing the acquisition correlations with three shifted code replicas together, we can reduce the acquisition time.

We report in Fig. 5 a pictorial representation of the bi-dimensional search result in the form of a 3D plot of the correlation results as a function of time and frequency (code delay and Doppler shift).

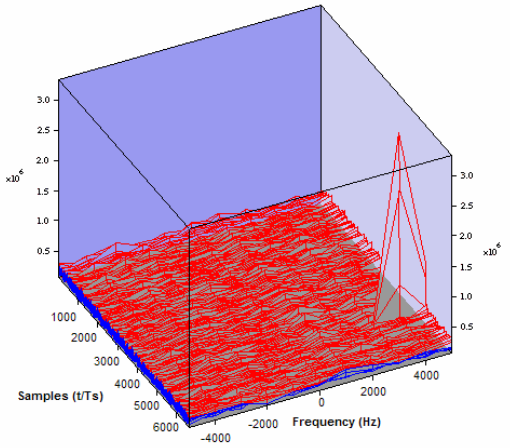

Fig. 5. Bidimensional Coarse Code Acquisition

This result is obtained trying to acquire a real GPS signal, in particular the 30 C/A code. As this picture shows the coarse Doppler shift is 4000 KHz.

B. Tracking Stage

When the acquisition stage is over, the tracking phase is executed. The channel architecture changes its structure, as Fig. 6 reports. Practically in this stage the receiver has to track the C/A code and to recover frequency residual offset, and carrier phase.

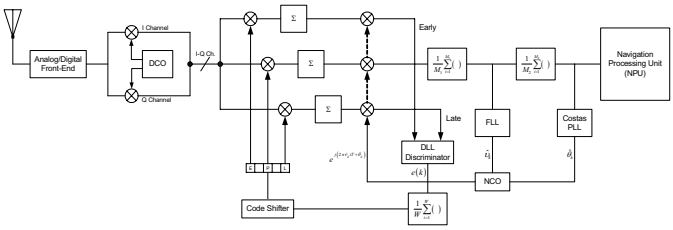

Fig. 6. Tracking Stage Channel.

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]. This stage is very important to detect the GPS/EGNOS data and then decode their messages during the Navigation Stage.

1) Delay Locked Loop (DLL)

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

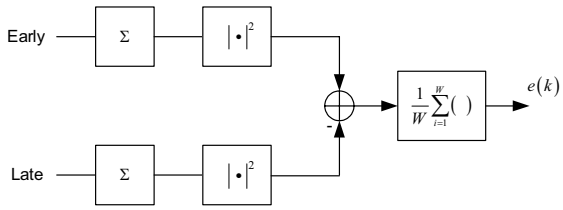

Fig. 7. DLL Discriminator.

This unit compare the correlation obtained on the early branch with the late one. In this way, we can measure a timing error that can be used to track the channel code and estimate its delay, according the following formula:

$$\tau_{k+1} = \tau_k - \gamma \cdot e_k \quad (5)$$

where γ it is DLL step-size [3], e_k is the discriminator output, τ_k is the code delay, and with k we point out the temporal index.

We report an example of code timing estimation in Fig. 8. As shown, we can track a real GPS code (30 C/A) and we can recover its code Doppler shift.

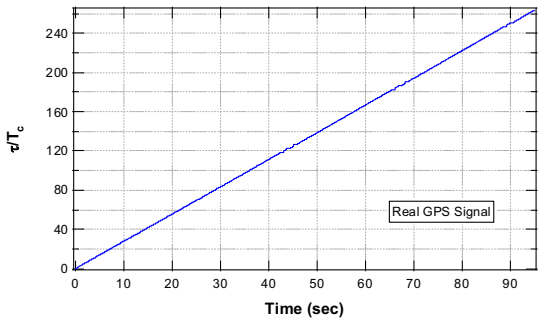

Fig. 8. Code Timing Estimation.

2) Carrier Frequency Locked Loop (FLL)

This module implements a 1st order FLL. This unit is dedicated to recover the residual Doppler shift and to compensate it, as Fig. 6 shows. In fact, we remark that from the acquisition stage we obtain a coarse frequency estimation because it performs a frequency search with 1 KHz of frequency step. The result is a residual Doppler smaller than 500 Hz. This unit is dedicated to recover this offset.

The frequency loop is pretty standard, with a digital frequency discriminator given by:

$$e(k) = \mathrm{Im}\left\{x_k^* \cdot x_{k-1}\right\}. \tag{6}$$

This formula is known as *Delay and Multiply*.

Fig. 9 shows an example of frequency estimation with 1 Hz Loop Bandwidth (B_L). In particular, it reports the frequency recovery of 30 C/A acquired in Fig. 5. As seen, the result is a fine frequency estimation: from 4 KHz (its coarse value) to 4.27 KHz (the fine figure).

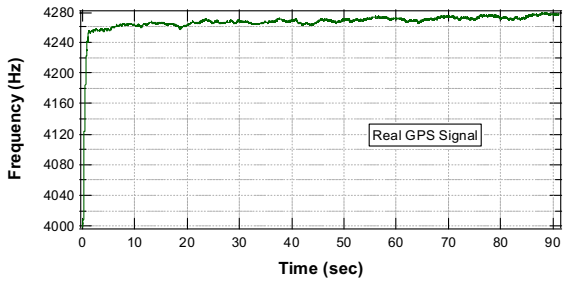

Fig. 9. Carrier Frequency Estimation.

3) Phase-Locked Loop (PLL)

This module implements a 2nd order PLL. It is the standard 2nd order decision-aided Costas PLL, shown in Fig. 10.

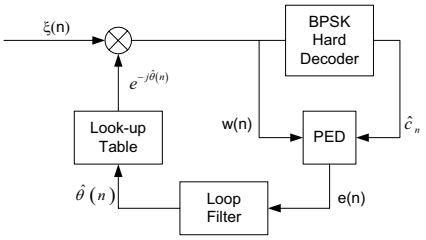

Fig. 10. Carrier PLL Architecture.

It can estimate and compensate the carrier phase with a residual carrier frequency uncompensated by FLL. In this way it is possible to lock carrier phase and frequency to perform coherent data detection of the BPSK navigation data.

The phase error is computed as follows:

$$e(n) = w_I(n) \cdot \mathrm{sgn}\left[w_Q(n)\right] \tag{7}$$

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

Fig. 11 depicts a carrier phase recovery of a real GPS signal. In particular it shows the phase error. As seen, at the end of a preliminary transitory, this figure jitters around zero degree.

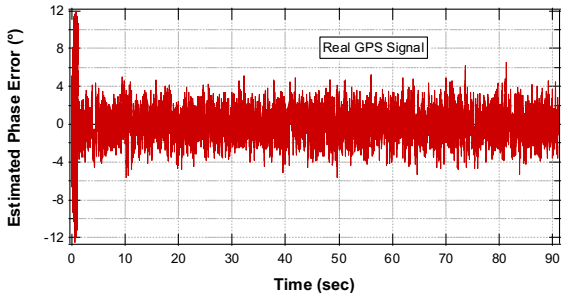

Fig. 11. Carrier Phase Estimation.

4) Navigation Library

During tracking stage, this library allows to decode the GPS subframe and to execute the navigation algorithms. 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. The pseudoranges are evaluated measuring the signal reception time obtained adding the code timing estimation from DLL unit with the time to lock the GPS data.

V. CONCLUSIONS

The aim of this paper is showing how to design a low-cost GPS/EGNOS software receiver, with characteristic of full re-programmability and re-configurability. In particular we remark that the proposed architecture is very simply and the used signal processing algorithms are standard and with a very low complexity.

We also underline that the analog/digital 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", *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.