Low-cost GNSS sampler based on the beaglebone black SBC

Olesen, Daniel; Jakobsen, Jakob; Knudsen, Per

Published in:
Proceedings of the 8th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)

Link to article, DOI:
10.1109/NAVITEC.2016.8060034

Publication date:
2016

Document Version
Peer reviewed version

Low-cost GNSS Sampler based on the BeagleBone Black SBC

Daniel Olesen, Jakob Jakobsen & Per Knudsen
DTU Space, Dept. of Geodesy
Technical University of Denmark
Email: {danole, jj, pk}@space.dtu.dk

***Abstract*—This paper describes the design of a simple low cost GNSS sampler with integrated storage. The sampler is built from low cost, off-the-shelf components. The proposed design stores digital IF samples from two separate front ends and record the data to a SD card for post-mission processing. The GNSS sampler currently supports GPS and GLONASS signals, but could also be configured for Galileo reception. The design is based on the popular low cost BeagleBone Black Single Board Computer (SBC) and two evaluation kits of the MAX2769 GNSS Radio Frequency (RF) front end. The design is based on two identical coprocessors in the processor of the BeagleBone, known as Programmable Realtime Units (PRU). The sampler has been designed for portability and data acqusition on small Unmanned Aerial Vehicles (UAVs).**

I. Introduction

GNSS Software Defined Receivers (SDRs) are becoming an increasingly important tool in academic research projects. In contrast to commercial GNSS receivers, software implementations offer the ultimate level of control and flexibility for the processing stage. A software based receiver allows for great flexibility, offering support for various modulation techniques, bandwidths and algorithms. In terms of GNSS receivers, reported software implementations have steadily grown in numbers and capabilities over the years. Licensed implementations are offered from commercial suppliers, universities as well as free open-source versions are steadily emerging.

A SDR gives added control and access to parameters at the tracking loop level, which allows for research in more advanced applications such as e.g. Ultra-Tight/Deep Integration of GNSS and Inertial Navigation Systems (INS). Another application, where a SDR is useful is within Space Weather monitoring. Here the two scintillation indices S_4 and σ_ϕ are traditionally used as a measure for the scintillations caused by the ionosphere. Direct access to the correlators and the signal filtering gives control over the quality of these parameters, see [1]. This flexibility and low level access to the hardware is also very useful in other research areas such as multipath mitigation, GNSS based reflectometry, GNSS jamming and spoofing detection.

A prerequisite for any SDR is access to Intermediate Frequency (IF) samples acquired by an RF front end. To this end, there exist a number of available commercial options for RF front-ends to GNSS software receivers. These varies from professional multi-purpose front ends supporting wide frequency ranges, such as the USRP product line by Ettus Research (National Instruments). A number of RF front ends for radio amateurs/hobbyist has furthermore been used for GNSS applications, even a slightly modified TV-Tuner has been demonstrated as a workable GNSS RF front end [2]. A number of more specialized front ends developed specifically for GNSS reception also exist, such as the SiGE GN3S sampler or the Stereo from Nottingham Scientific Limited (NSL).

To the authors best knowledge most dedicated GNSS SDR front ends are designed as streaming devices, which transfer either real or quadrature IF samples in realtime over USB or Ethernet interfaces and thus require a PC for storage or realtime processing. In certain applications, where weight and size is critical, this would not be an optimal approach. Due to this reason, we propose a simple GNSS sampler with integrated storage for subsequent post-processing of data. The system is built using low cost, off-the-shelf components. We are using 2 identical RF front ends to support GPS L1 C/A code and GLONASS L1 C/A code. This is necessary since the used RF front ends have a bandwidth that do not cover the frequency range of both constellations simultaneously.

In this paper, a small, portable GNSS sampler based on the low cost BeagleBone Black Single Board Computer (SBC) and two evaluation kit of the commercial GNSS RF front end, MAX2769 is described. The advantages of this system are low cost, size and flexibility. Often GNSS front end designs are built using a Micro Controlling Unit (MCU) for USB or ethernet communication and programmable logic, such as a CPLD or FPGA for bit manipulation and high bandwidth transfer of IF samples. This approach requires that the designer is experienced with embedded electronics and digital logic design. The setup used for our sampler is besides a few simple electrical connections, configured solely in software and the integration thus only require minimum expertise in hardware. In this configuration, we believe that the proposed setup would be an ideal starting point for researchers who are interested in GNSS SDR applications, but are working on limited budgets and does not necessarily have expertise in hardware engineering.

The sampler was developed with a distinct focus on portability to ensure it can be used in small UAV applications. The authors have previously proposed an embedded realtime GPS SDR design based on the Parallella board and the MAX2769

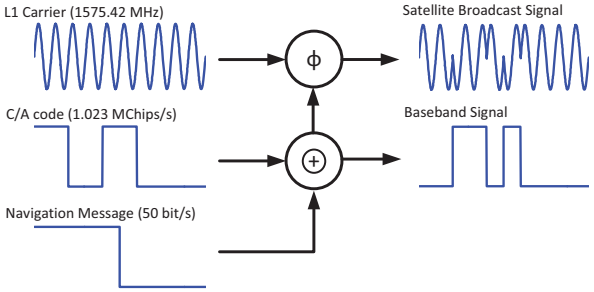

Fig. 1. Principle of GPS CA code modulation

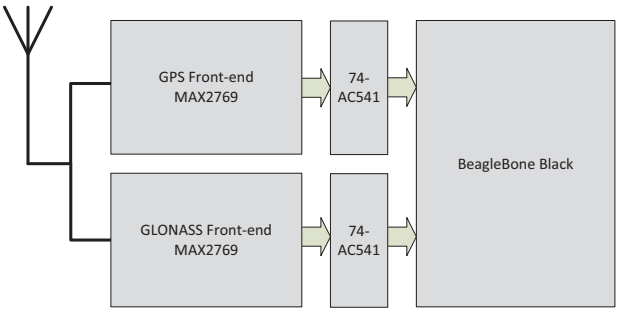

Fig. 2. Hardware Block diagram of GNSS Sampler

RF front end for UAV applications. [3].

II. GPS and GLONASS signals

In this paper we only consider GNSS signals transmitted in the L1 band from the GPS and GLONASS constellations. Both systems have multiple signals in this band, but we will in the following only consider the civilian (non-encrypted) Coarse / Acquisition (C/A) signals.

A. GPS C/A code

The GPS Coarse / Acquisition (C/A) code is a 1023 chip Pseudo Random Noise (PRN) sequence transmitted on the L1 band with a carrier frequency of 1575.42 MHz. The C/A code is modulated on to the carrier by Binary Phase Shift Keying (BPSK). As all GPS satellites are using the same carrier frequency, a Code Division Multiple Access (CDMA) coding scheme is applied on the C/A code. The C/A code is for each satellite a unique PRN code known as a gold code. Gold codes have a guaranteed minimum cross-correlation with other gold codes and hence the satellite transmitting the signal can be identified by correlation with a receiver generated replica. The chiprate of the C/A code is 1.023 MChips/s, corresponding to a code sequence length of 1 ms. The C/A code is modulated with the satellite navigation message (50 bits/s) using a modulo-2 addition, which is commonly also referred to an eXclusive OR (XOR) operation. The navigation message contains the ephemeris, time etc. for the satellite. A visual illustration of how the C/A broadcast signal for GPS satellites are created are shown in Figure 1. For more technical details of the GPS satellite signals, please refer to the GPS Interface Control Document (ICD) [4].

B. GLONASS C/A code

The GLONASS C/A code is transmitted using Frequency Division Multiple Access (FDMA). The GLONASS satellites are assigned different frequency slots in the range $1597.5515-1605.875$ MHz Where each channel/slot has a separation of 562.5 kHz. There is a total of 14 seperate slots (k=-7:+6). As the number of slots are less than the number of satellites in the constellation, some satellites transmits on the same frequency. The assignment of frequency slots are however done in a way, such that satellites sharing the same slot are not visible to a ground based receiver on the same time. The GLONASS C/A code is a 511 chip PRN sequence, which is used for all the satellites. This code has a chip rate of 511 MChips/s, equaling a sequence length of 1 ms. The C/A code is modulated with a 50 bit/s navigation message and a 100 Hz auxilary meander sequence using modulo-2 addition. For more information, refer to the GLONASS ICD [5].

III. System Description

A high level diagram of the system is shown in Figure 2. The two RF front ends uses an RF splitter to connect to a single GNSS antenna, the ADC outputs of both front ends are connected to digital inputs on the BeagleBone Black. The power supply of the front ends are supplied directly from a 3.3V output from the BeagleBone. A simple line-driver/buffer (74AC541) is used between the front ends and the digital inputs on the BeagleBone, this was introduced to prevent clock-jitter due to loading problems experienced from a direct connection.

A. MAX2769 GNSS Front End

The RF front end of our GNSS sampler/data-collector is based on two MAX2769 Integrated Circuits from Maxim Integrated [6]. Incorporated in the chips is a complete RF processing chain which can be configured for GPS L1, GLONASS L1 and Galileo E1 reception. The RF front end features a programmable single-conversion stage and supports both active and passive antennas. The RF front end is available as an evaluation kit (EV-kit) equipped with a number of SMA connectors for evaluation of separate stages of the front end circuitry. The EV-kit comes with a $16.368MHz$ Temperature Compensated Crystal Oscillator (TCXO) but also features an input for use of an external oscillator, which makes it possible to evaluate the system using different grades of reference oscillators. A high level functional diagram of the MAX2769 is shown in Figure 3.

The RF front end is connected to either an active or passive antenna and the signal is amplified though a Low Noise Amplifier (LNA). The received signal then propagates

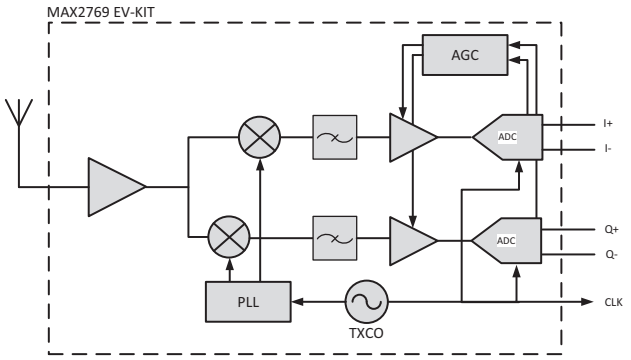

Fig. 3. Functional Diagram of MAX2769 GNSS Front End

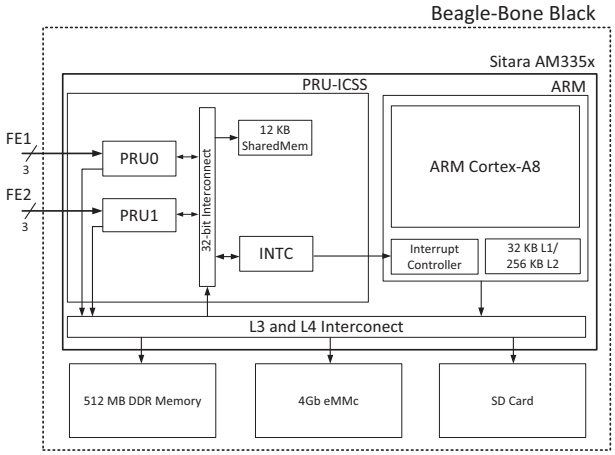

Fig. 4. High level Hardware Diagram of the BeagleBone Black

through a quadrature mixer, where the signal is converted to an Intermediate Frequency (IF).

The local oscillator (PLL Synthesizer) for the mixing-stage can be programmed to any given frequency in the range 1550-1610 MHz with 40 Hz separations.

After down-mixing the IF signal passes through a configurable filter, which can be set to either Bandpass or Lowpass operation. After the filter, the I and Q signals are sent through an Programmable Gain Amplifier (PGA) before AD conversion. The RF front end is equipped with an Automatic Gain Control (AGC) circuit, which adjust the PGA's for ideal saturation of the ADC. The ADC has a configurable sample rate of up to 50 MSamples/s. The ADC has the ability to quantize the signal with up to 3 bits precision for real samples and (2+2) bit precision for I/Q sampling. The configuration of the mixer, filters etc. is done using a Serial Peripheral Inteface (SPI). The MAX2769 also have a number of preconfigured device-states, which can be activated by enforcing static logic levels to the individual signals of the SPI bus. The two sections below, describes how the two MAX2769 RF front end was configured for GPS and GLONASS reception.

1) GPS Reception: The carrier frequency for GPS L1 is $F_{RF} = 1575.42MHz$. To configure the front end for GPS reception, a low side injection oscillation frequency of $F_{OSC} = 1571.328MHz$ has been chosen. This gives an IF Frequency of $F_{IF} = F_{RF} - F_{OSC} = 1575.42 - 1571.328 = 4.092MHz$ The IF bandpass filter can be programmed to have a bandwidth of 2.5 MHz, 4.2 MHz, 8 MHz and 18 MHz. In addition the filter can be implemented as a 3rd or 5th order polyphase filter. We have selected the bandwidth to 2.5 MHz using the 5th order filter setting. For GPS reception, we use real sampling with 2 bits resolution and a sample frequency of 16.368 MHz.

2) GLONASS Reception: In order to capture all the channels we require the bandwidth of the front end to be $1605.375MHz - 1598.0625MHz + 2 \times 0.562MHz = 14.4365MHz$. In order to achieve this, the Local oscillator was tuned to $F_{OSC} = 1600.995MHz$. The IF filter was configured to a 5th order polyphase lowpass, with a dual sided bandwidth of $18MHz$. The PGA's was set in a static gain operation. In order to capture the negative side of the spectrum, we use quadrature/complex sampling with 2 bits precision and a sample frequency of 16.368 MHz.

B. BeagleBone Black SBC

In order to store the IF samples, a BeagleBone Black SBC has been used for data storage. This platform can be purchased for approximately $50 and has a number of hardware interfaces and GPIOs for external peripherals.

The transfer of the IF samples from the ADC's is done using a parallel interface on the MAX2769 front ends and 8 digital inputs on the BeagleBone Black.

1) System Description: The BeagleBoard Black consists of a Texas Instruments Sitara AM335x Processor [7]. This features an ARM Cortex-A8 core running 1GHz. In addition the processor is also equipped with a Programmable Real-Time Unit SubSystem and Industrial Communication Subsystem (PRU-ICSS) [8]. This module consist of two 32-bit RISC cores which runs with a clock-frequency of 200 MHz. Each core has 8 Kb of Instruction Memory and 8 Kb Data memory. In addition the PRU-ICSS also includes 12 Kb of shared memory. The PRU-ICSS features an interrupt controller which is directly connected to the interrupt controller of the host-system (ARM). The PRU-ICSS is especially useful in timing-critical applications, as they operate independently from the linux operating system on the ARM processor. In our design we have utilized both cores to transfer data from the two connected front ends. A block diagram of the core-components and peripherals used for the system is shown in Figure 4.

The PRU cores is configured to have access to a shared portion of the system memory and stores the samples in that range. Two adjacent circular reception buffers for each PRU are created to ensure continuous operation, such that one portion of the buffer can be filled while the other is

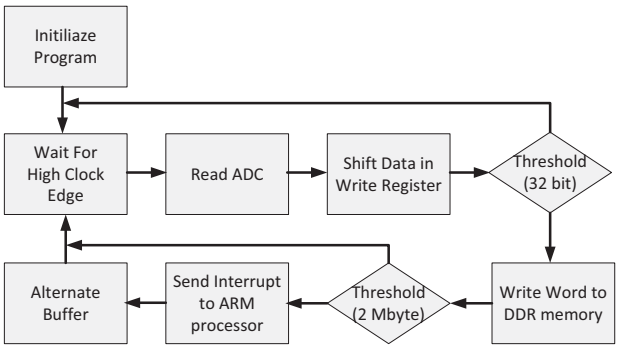

Fig. 5. Flow-chart of continuous data-sampling in the PRU cores

being emptied. The PRUs has been configured to generate an interrupt to the ARM processor, when one of the receive buffers is full. In this way we offload the ARM processor, as it only has to write the stored data onto a file on a SD card whenever an interrupt has been is triggered.

2) Software Description: The first step in the configuration of the BeagleBone Black as a data acquisition device, has been to make a Device Tree Overlay to configure the pin-muxes for the GPIO to enable PRU inputs. This overlay is then loaded at boot-time to instruct the operating system to configure the GPIO accordingly. Subsequently a driver package for the PRU-ICSS has been installed [9]. This package was initially created by Texas Instruments, but is now open-source community driven. The package provides a linux user-space driver which allows external memory allocation, program control functions and interrupt handling from the ARM system. The installed driver-package allows to write program for the PRU's in assembly. Texas Instruments, have also released a C compiler for the PRU cores, but due to the time-critical operations in the system, we have preferred to program directly in assembly.

The software for the ARM processor has been written in C++ and compiled using the GNU Compiler Collection (gcc). The main task of the host code is to handle interrupts from the PRUs and transfer data from reception buffers to an sd-card.

The code executed on each PRU has been written in assembly. A flow-chart of the continuous operation is shown in Figure 5. The main task of this code is to read ADC data when a high clock edge is present and store samples continuously in the receive buffers. The PRU connected to the GPS front end stores 4 samples in each byte (2 bit I) and the PRU connected to the GLONASS front end stores 2 samples per byte (2 bit I + 2 bit Q). A write to DDR memory is initiated every time 32 bits (word) of data has been collected. Each time a write has been done, the program checks if 2 Mb of data has been captured, this amount correspond to half of the total buffer-space for each PRU. If this is the case an Interrupt is raised and the start-address of the buffer is alternated between buffer 1 or 2.

The software written for the ARM processor creates two Interrupt handling threads which read out the reception buffers and writes blocks of data to files on the SD card.

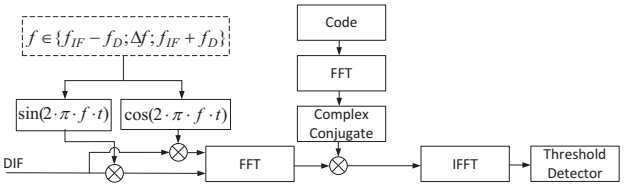

Fig. 6. Parallel Code-Phase Search

IV. Baseband Processing

This section provides a brief summary of basic GPS and GLONASS acquistion and tracking algorithms. The focus in this paper, has been on the hardware implementation of the GNSS sampler and not on baseband processing algorithms. For sake of completeness and to inform the reader, how data has been in subsequent sections has been processed, we present the basic methods used in this work. The code we have used for Baseband processing is based on a modified version of the GPS MATLAB SDR toolbox developed by [10]. The modifications have been made to support GLONASS acqusition and tracking.

A. Satellite Acquisition

The Satellite acquisition is based on the well known Parallel Code Phase Search Algorithm [11], as shown in Figure 6. This algorithm effectively performs correlation in the frequency domain by using the cross-correlation theorem.

$$(f \star g)(x) \leftrightarrow \mathcal{F}^{-1}\left(F^*(u)G(u)\right) \tag{1}$$

where F^* is the complex conjugate of the fourier-transform of f and G is the fourier tranform of g.

The difference between GPS and GLONASS acqusition is, that for GPS we search through all 32 PRN codes, and in frequency only make a coarse sweep in carrier (IF) frequency to cover all possible doppler frequencies. For GLONASS, as the satellites are distributed in different frequency channels and all satellites use the same PRN sequence, the acquisition algorithm was modified accordingly. Subsequent to the Parallel-Code Phase Search, we perform a fine-frequency estimation using a coherent sequence of 10 ms data multiplied with the detected C/A code phase, in order to wipe off the code signal.

B. Satellite Tracking

The tracking system for GPS and GLONASS C/A code signals are very closely related. As already explained, there is only one PRN sequence shared for all GLONASS satellites, where each GPS satellite have a unique sequence. In terms of implementation, the primary difference has been our use of real sampling for GPS and quadrature sampling for GLONASS. In addition to the diagram in Figure 7, in the quadrature case, a phase rotation is applied prior to the Early, Prompt and Late code-correlators.

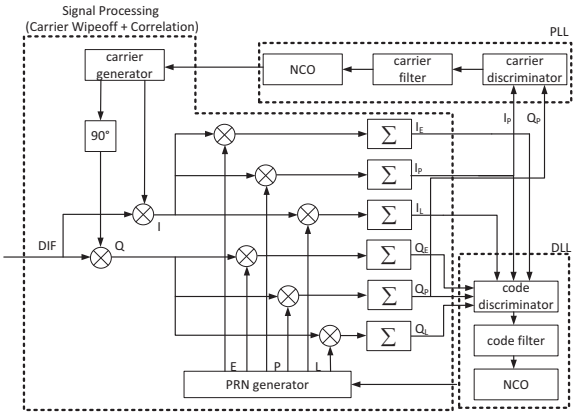

Fig. 7. Satellite Tracking loop

V. Results

A picture of the lab prototype is shown in Figure 8. The prototype is mounted on a payload aluminium plate for use and testing on a UAV.

A. Maximum Bandwidth and Storage Requirements

Both connected front ends have a sample clock-rate of 16.368 MSamples/s. An emperical test has shown, that the maximum clock frequency that could be used is around 20 MHz. Higher rates is possible for short intervals, but the operations to switch between buffers and generate interrupt in the PRU cores limits the continuous performance. In terms of storage, the GPS data file packs 4 (2-bits) samples into one byte, whereas for GLONASS a byte can only store 2 samples (2 bit I + 2 bit Q). The data throughput rate is approximately 4 MB/s for GPS and 8 MB/s for GLONASS. For a 32GB SD card this would be approximately 45 minutes of IF data.

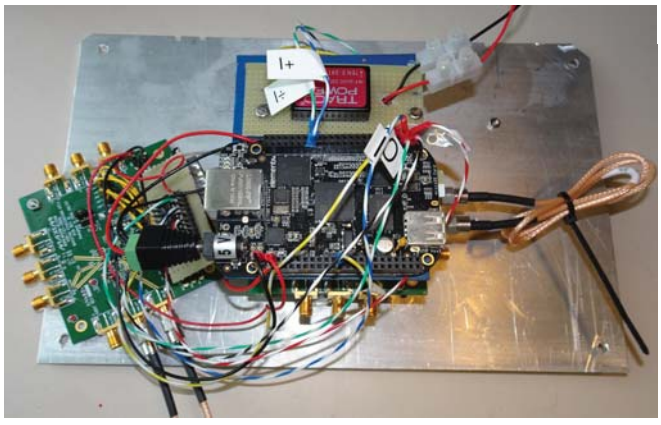

Fig. 8. Lab prototype of the GNSS sampler mounted on aluminium plate. In addition to the aforementioned, components, a 5V DCDC converter is part of the setup to power the system from a UAV battery

B. Static Test using roof-mounted Antenna

In order to test the system, we performed a static test using a high-end GNSS antenna mounted on the roof of DTU Space Institute building, located in Kgs. Lynbgy, Denmark. The data collection took place the 3rd of July, 2016 at 16:24 (UTC+1). The primary aim from this test, has been to verify that satellites could be acquired and tracked from both the GPS and GLONASS constellations using captured data from the developed system.

1) GPS reception: The acquisition result for the GPS constellation, is shown in Figure 9.

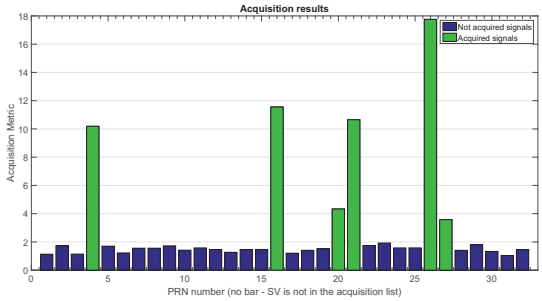

Fig. 9. Results from GPS Acquisition. The green bars indicate acquired satellites.

From the figure, it can be seen that we could acquire 6 GPS satellites from the recorded IF samples.

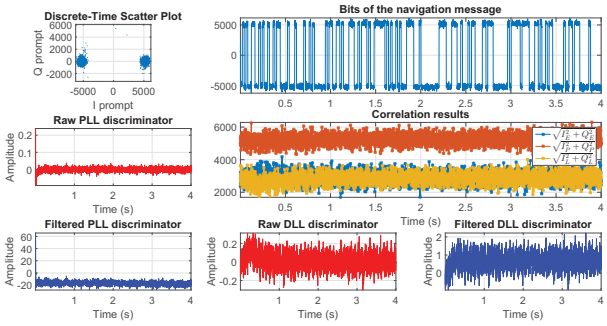

Fig. 10. Tracking of GPS SV26

A tracking plot for PRN26 based on 4000 ms of data is shown in Figure 10. This figure, show a scatter plot of the In-phase and Quadrature prompt outputs, the raw and filtered discriminator outputs from the code- and carrier tracking. It furthermore shows a time-series of the In-phase prompt output as well as the combined energy in the early, prompt and late correlations. From this metric, we can see that the tracking maintains a solid lock on the code-phase as the energy in the prompt correlators is approximately double of the early and late correlators.

2) GLONASS reception: In Figure 11, the result of the GLONASS acquisition is shown. At the time of the test, it can be seen that 4 satellites was acquired.

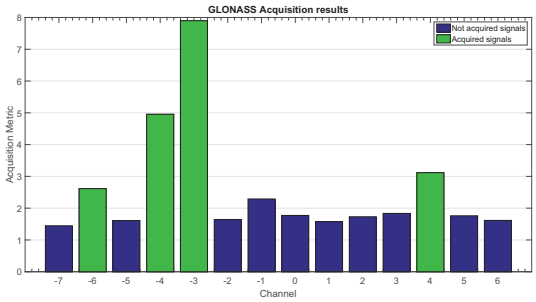

Fig. 11. Results from Glonass Acquisition. The green bars indicate acquired satellites.

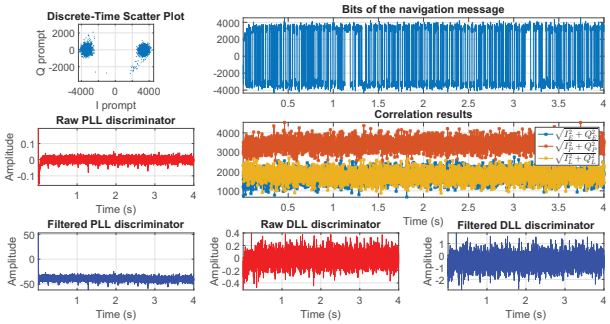

Fig. 12. Tracking of GLONASS satellite (frequency slot -3)

A tracking plot for the GLONASS satellite in frequency slot -3, based on 4000 ms of data is shown in Figure 12. This figure, show the same metrics as for the GPS tracking. Notice, the more frequent bit changes in the time-series of the in-phase promt correlator. This is due to the 100 Hz, meander sequence used for GLONASS modulation. Similar to the tracking of GPS SV26, a solid lock is obtained on the code-phase as the energy in the prompt correlators is approximately double of the early and late correlators.

The main challenge we faced for obtaining good reception for GLONASS satellites has been to select a optimum gain for the Programmable Gain Amplifiers after the mixing-stage, as the AGC has been disabled for the GLONASS front end. Due to the wideband setting of the IF filter, the front-end seems also more sensitive to out-of-band interference. We have not used external RF filters for either front ends but we believe that optimizing the reception-chain in this manner, would lead to better reception quality and more visible satellites.

VI. Future Improvements

The system presented in this paper is a minimal working prototype and proof-of-concept that the BeagleBone can be used as an interface and storage device for two GNSS RF front-ends. It is also possible to configure, the MAX2769 RF front end for Galilleo reception. This can be done, by using a wide-band, low-IF setting of the GPS front-end. We have not experimented with this possibility at the time of writing, but it is something we will look into in the future.

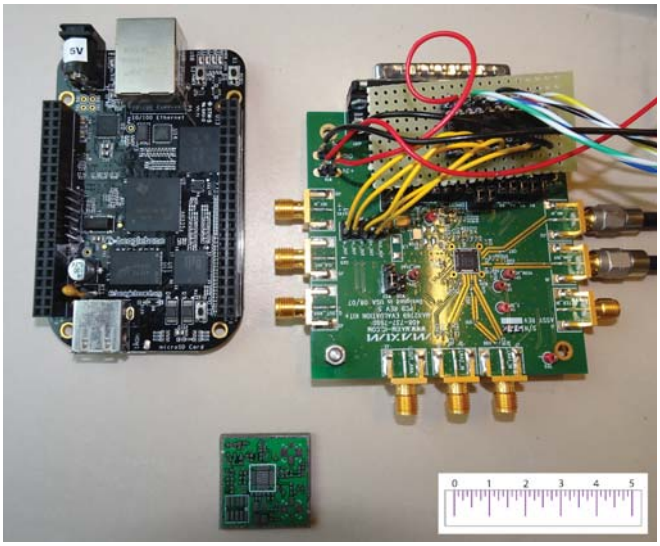

Fig. 13. Size comparison of MAX2769-EVKIT (right) and MAXIM reference design (bottom). The BeagleBone Black SBC is shown to the left. A ruler has been overlayed the picture for reference. The units are in cm's.

A nice-to-have addition to the current design would be to establish an SPI interface to both MAX2769 front ends. This task is certainly possible, as the BeagleBone embeds two SPI controllers. This interface would allow us to reprogram the front-ends directly from BeagleBone and thus make it possible to implement run-time reconfigurations. From the MAX2769, it would furthermore have been possible to obtain the IF samples over this interface as the SPI controllers are internally connected to a Direct Memory Access (DMA) controller, which can be used for automatic addressing of incoming data.

Another thing, which we will work on for a future revision, is to use a common reference oscillator for both front ends, to ensure receiver clock-offset and drift rates would be same for both GPS and GLONASS.

As discussed in the introduction, portability and compactness is a major concern for experimenting with SDR on UAV's. The MAX2769 EV-KITS used as the basis for this paper are very flexible and great for evaluation, but in terms of size and weight they are suboptimal. The authors are currently working on a second revision of the system, in which the EV-Kits will be replaced with a more compact PCB, based on a reference design from Maxim Integrated [12]. The revised PCB measures only $25mm \times 25mm$. For a size comparison, see Figure 13.

The second revision of the GNSS sampler, is still under developement, but a photo of the current prototype is shown in Figure 14. The new design is made as a cape, which directly can be plugged into the BeagleBone's pin headers.

VII. Conclusion

In this paper a portable low cost GNSS sampler with integrated storage built from off-the-shelf components was presented. The design does not require expertise in hardware engineering and thus could be a good starting point, for

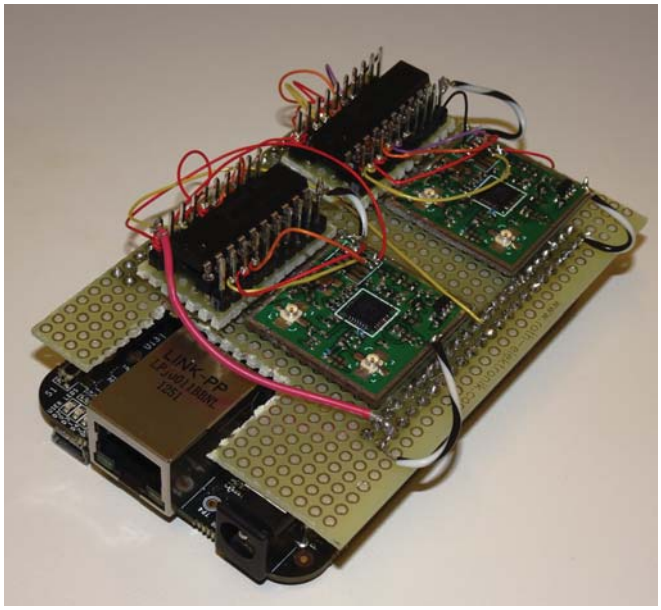

Fig. 14. Second-revision prototype. Two RF front-ends based on the MAXIM reference design are mounted on a perfboard, which has direct connections to the pin-headers on the BeagleBone Black SBC.

researchers and students interested in getting started with Software-Defined GNSS receivers. To the authors best knowledge, commercial GNSS Front-ends for SDRs are built as streaming device to support real-time processing. In many applications, post-mission processing is adequate and the purpose of this work was to design a simple, small-size, self-contained GNSS sampler where the IF samples can be stored on a removable media. This was accomplished by utilizing the two coprocessors featured in the processor on the BeagleBone Black SBC.

We have shown the feasibility of this design and verified the integrity by performing a static test from which, we could acquire 4 GLONASS satellites and 6 GPS satellites by standard baseband processing algorithms. The design of the sampler was initiated in order to perform IF data-collection from small-UAVs.

ACKNOWLEDGMENT

The authors would like to thank the Innovation Fund Denmark for partial funding of the first author's PhD study. We would also like to thank Michael H. Avngaard, assistant engineer at DTU Space for his help and assistance in the laboratory during the development of the GNSS sampler.

REFERENCES

[1] J. T. Curran, M. Bavaro, J. Fortuny, and A. Morrison, "Developing an ionospheric scintillation monitoring receiver," *InsideGNSS*, vol. 9, no. 5, 2014.

[2] C. Fernndez-Prades, J. Arribas, and P. Closas, "Turning a television into a gnss receiver," in *Proceedings of the 26th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2013)*, Nashville, TN, September 2013, pp. 1492–1507.

[3] D. Olesen, J. Jakobsen, and P. Knudsen, "Software-defined gps receiver on the parallella-16 board," in *Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2015)*, Tampa, FL, September 2015, pp. 3171–3177.

[4] *Navstar GPS Interface Specification (IS-GPS-200H)*, 2013.

[5] *GLONASS Interface Control Document, ver. 5.1*, 2008.

[6] *MAX2769 Datasheet*, Maxim Integrated, 2010.

[7] *Sitara AM335x ARM Cortex-8 Microprocessors*, Texas Instruments, 2013.

[8] *AM335x PRU-ICSS Reference Guide*, Texas Instruments, 2013.

[9] "Github repository for beaglebone black pru-icss api," https://github.com/beagleboard/am335xprupackage, accessed: 2016-06-24.

[10] K. Borre, D. Akos, N. Bertelsen, P. Rinder, and S. Jensen, *A Software Defined GPS and Galileo Receiver*. Birkhauser, 2007.

[11] J. Bao and Y. Tsui, *Fundamentals of Global Positioning Receivers: A Software Approach*. John Wiley and sons, 2000.

[12] *MAX2769 GPS Reference Design, APP4279*, Maxim Integrated, 2008.