Table of contents

Array signal processing optimization in GNU Radio for tracking and receiving applications

E. Bieber[1], C. Campo[1,2], L. Bernard[1], H. Boeglen[2], S. Hengy[1], J.-M. Paillot[2]
[1] French-German research institute of Saint-Louis (ISL), Saint-Louis, France
[2] Université de Poitiers, XLIM, UMR 7252, Poitiers, France

Abstract

Among other missions the French German research Institute of Saint-Louis (ISL) works on array signal processing for secured communications between high speed projectiles and allied base stations. Within that framework, a projectile tracking receiving station based on commercial Software-Defined Radios (SDR) was developed using four channels to steer an antenna array and recombine the received signals, hence improving the gain of the receiving station. A transmitter embedded in the projectile sent data to the developed receiving station at a 2 Mbits/s. In order to decode and process in real time the data received by the four channel antenna array, a high sampling rate was required. As this highly resource consuming application resulted in sample overflows that is, in periodic losses of data between the SDR and the computer, an optimization of our algorithms computed on GNU Radio and the communication between our blocks proved to be necessary.

This paper intends to provide feedback on our optimization work. Some of the main problems we encountered and the solutions we propose to solve them are briefly exposed and will be further detailed in our oral presentation.

1 Introduction

Among other missions, the ISL works on developing secure communications between fired projectiles and ground stations for future smart ammunitions. Antenna arrays then offer many advantages such as directional radiation patterns that can be dynamically reconfigured to follow a moving transmitter, fight against hostile jammers or listeners, etc. In this context a SDR-based receiving station was developed using GNU Radio and the commercial Universal Software Radio Peripherals (USRPs) sold by National Instruments, and proved to be able to electronically follow a transmitter by steering a four element Uniform Linear Array (ULA), increasing the gain on the received signal. However in order to simultaneously decode the transmitted signal at a 2 Mbits/s baud rate, the sampling rate for all channels needed to be raised to 8 MSamples/s. To compose with the SDR requirements it was necessary for our laptop to receive data at a total rate of 33.33 MS/s (for all four channels), process the received data with our implemented algorithms such as beamforming and direction finding (DOA) that were introduced in [1], and record the whole in real time, resulting in a highly data consuming application. Our first attempt to run this application with a laptop equipped with an Intel i7 processor, 32 GB of RAM and a Samsung 850 evo SSD resulted in data overflows, i.e. in periodic data losses due to the lack of computation power, hence forcing us to think carefully about computation efficiency when implementing our application in GNU Radio.

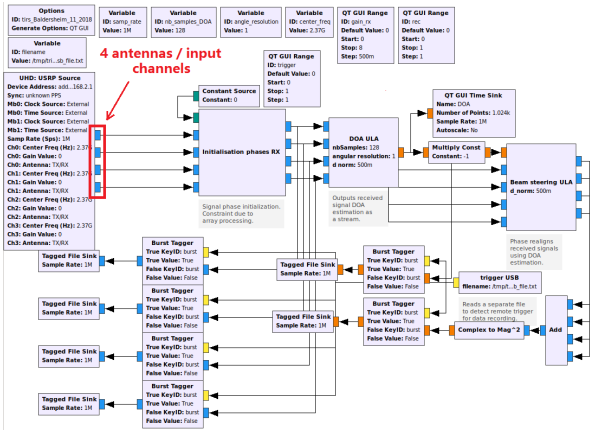

Figure 1: Flowgraph runnable at 1MS/s but creating data overflows at 8.33MS/s.

Fig. 1 exhibits a flowgraph that managed to perform projectile following at 1MS/s but created data overflows when higher sampling rates were required. The important number of streams and use of loops instead of vector oriented library kernels (Volk) were partly responsible for these overflows.

This paper does not focus on our application and results, but intends to present our work on code optimization, especially to extend the computation efficiency of our algorithms and flowgraphs developed in C++ in GNU Radio [2]. The remaining of this abstract briefly covers suggested improvements we have explored to avoid overflow issues.

2 Suggested improvements for optimization

The first and most obvious question that arises is the network throughput between the SDR and the laptop, as well as the laptop capability to record all the needed data fast enough. In the case of our application, the total 33.33MS/s sampling rate forced us to install a Thunderbolt 2 SANLink adapter. It can also be useful to create a RAMDisk if the drive is not capable to record fast enough. Reducing the amount of written data will have a positive effect on the bitrate: users should write binary files rather than ASCII ones and use data types with smaller memory size.

Once it is sure network throughput and data recording speed are not the bottlenecks that create data overflows, one can investigate his source code to enhance his application efficiency. Due to GNU Radio's way of processing streams as buffers of data, the *work()* method of a block is usually a two-level loop that parses each sample of each input stream. One should avoid multiple computations of invariant values inside loops, but also try to use optimized functions such as *memcpy()* or the Volk library [3] kernels instead of these loops whenever possible.

However even if correctly managing streams between blocks allows to spare resources, it remains important to limit those streams when they are expendable. Although it might be tempting as a fast implementation to simply add a stream to a block as a trigger or a way to share a variable between blocks, it is computationally expensive and can be responsible for data overflows when high sampling rates are required. In order to efficiently communicate information between blocks, GNU Radio natively offers the possibility to tag existing streams with metadata. Since a tag is associated to a sample of a data buffer, we can consider tags as a synchronous communication vector. For asynchronous communication GNU Radio allows blocks to send messages to other blocks. A message is a 1 to N communcation carried out by the sender: the receiver message handler is called for each pending message. As no native option is given for users to develop blocks that can asynchronously use a shared variable, we developed a new communication vector based on static variables that allow variables to be read and written by several blocks in a thread-safe way assured by a mutex. Further details on our proposed communication vector between blocks will be given in our presentation.

After information communication between blocks has been verified enabling real time scheduling option will allow GNU Radio threads to have priority over concurrent threads. Finally blocks using computationally heavy algorithms like DOA estimation, etc, can be bound to a dedicated processor core while less demanding threads are bound to a pool of remaining cores. It can be noted that sometimes splitting such blocks into several ones will take advantage of the multi-core environment.

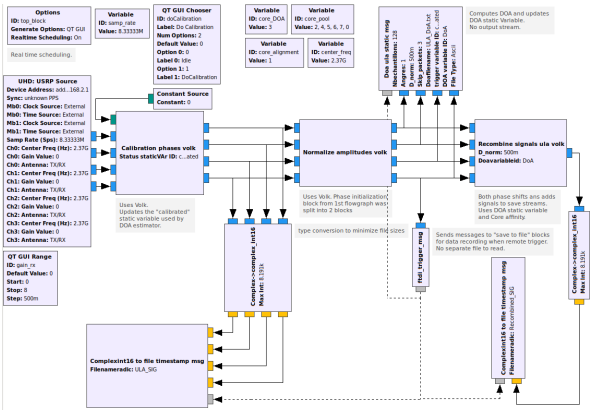

Figure 2: Flowgraph runnable at 8.33MS/s.

Fig. 2 shows an optimized version of the previous flowgraph that can be run on the same laptop at 8.33MS/s with a graphical view of the four received signals.

3 Conclusion

This paper presents the main modifications we brought to our developed blocks, making our application runnable for four channels at a 8.33MS/s sampling rate. An alternative communication vector between blocks that fits some of our particular needs has been mentioned, and all the suggested improvements presented above will be further detailed in our oral presentation.

References

[1] C. Campo, L. Bernard, H. Boeglen, S. Hengy, J.-M. Paillot, *Software-Defined Radio system for tracking application*, EuCAP London 2018.

[2] *GNU Radio 3.6.4.2 C++ API documentation* at `https://www.gnuradio.org/doc/doxygen-3.6.4/index.html`

[3] *Vector Optimized Library of Kernels (Volk) doxygen documentation* at `libvolk.org/doxygen/`

SatNOGS: Towards a Modern, Crowd Sourced and Open Network of Ground Stations

Manolis Surligas[1,2], Matthaios Papamatthaiou[1], Ilias Daradimos[1], Vasilis Tsiligiannis[1], Pierros Papadeas[1], Agisilaos Zisimatos[1], Fredy Damkalis[1], Kostis Triantafillakis[1], George Vardakis[1,2], Nestoras Sdoukos[1,2], Nikos Karamolegkos[1,2]
[1] Libre Space Foundation, Athens Greece, info@libre.space
[2] Computer Science Department, University of Crete

Abstract

Over the last years the launching cost of a payload in space has been significantly reduced and this trend is expected to continue, as the interest for space applications is increasing. The reduced launch cost and the advancements in technology, gave the opportunity to small satellites to revolutionize access to space.

The majority of the small satellites missions are targeting the Low Earth Orbit (LEO). Due to the nature of this particular orbit, communication with a satellite is possible only for a few minutes per day for a given location. This raises the need for multiple ground stations in several geographic locations. Although such an infrastructure is possible, most of the times it is both complicated and expensive for research or educational entities to obtain. Given the fact that each ground station exhibits a small per day utilization for a specific satellite, the idle time can be used for reception of other missions.

SatNOGS is an open source software and open hardware project that addresses this problem by interconnecting all participating ground stations, offering their idle time to other users of the SatNOGS network.

1 Introduction

SatNOGS is a global network of satellite ground stations, designed as an open source participatory project that receives data from Low Earth Orbit (LEO) satellites. This particular orbit provides the ground station with a reception window that is limited to a few minutes. Therefore, a ground station remains underutilized and idle for most of the time. On the other hand, using a single ground station means that there are only a few communication windows to send or receive data from a LEO satellite each day.

The concept of SatNOGS, is to use the deployed ground stations around the world in an efficient manner so that both the underutilization of them as well as the coverage problems can be resolved with a single solution. Aground station operator joining the SatNOGS network, provides the idle time of their ground station to other users, while at the same time they can take advantage of ground stations at various locations of the earth to schedule observations.

2 Architecture

The SatNOGS ecosystem consists of several components operating interchangeably. The SatNOGS Network infrastructure orchestrates the scheduling of each ground station, based on the trajectory of the targeted satellites and the online available ground stations, while allowing the owner of a ground station to have complete control over her hardware. The SatNOGS Database (DB)[1] is responsible for keeping the orbital elements and communication related information of each satellite. It also holds decoded frames from the deployed ground station network. The Data Warehouse, visualizes graphically the gathered data from each satellite using a web interface and the Grafana visualization framework. Last but not least, the SatNOGS Ground Station is the necessary software and hardware, that allows satellite tracking, the control of the RF front-end, signal reception and demodulation of possible data frames.

2.1 The network

The SatNOGS Network[2] infrastructure is the backbone of the SatNOGS ecosystem and instruments essential functionalities for the flawless, efficient and effective operation of the entire project. The network infrastructure services can be divided into two main categories, the back-end and the front-end. At the back-end, the SatNOGS Network keeps track of the available online ground stations and the list of the observation jobs that each one has been assigned. Moreover, it receives observation data from the deployed ground stations. The received data are either stored in the network infrastructure to be used through the web interface or are submitted to the SatNOGS DB[1] for fur-

ther analysis. The front-end of the SatNOGS Network provides a set of different web interfaces. For the ground station operators, a web-based control panel is available in order to control and configure remotely their stations. It also allows operators to schedule

2.2 Rotator hardware

SatNOGS Rotator is the mechanism that allows tracking of satellites in both azimuth and elevation axis. By design, the goal of the rotator is to keep the cost low, by using widely available materials of common sizes and 3D printed parts. For the users that cannot afford a rotator, SatNOGS can still operate without one, using a less directional antenna.

2.3 Client software

The SatNOGS Client[4] is a Python program that runs on the ground station computer. The client is responsible to retrieve observation jobs from the network and execute them. When a new job is received from the network, it is placed in an execution queue, sorted in chronological order based on the start time of the satellite pass. The client constantly monitors the local time of the ground station and the starting time of the first observation at the queue. When the timing is proper, the client removes the observation job from the queue and prepares to execute it. To do so, it initializes the SDR front-end with the RF parameters described by the job and then executes the appropriate GNU Radio script provided by the gr-satnogs OOT module[3]. Meanwhile, the client controls the rotator, so the antennas can track the trajectory of the targeted satellite. When the observation job is finished, the client instructs the rotator to place the antennas in a park position. Then, all the resulting files generated by the grsatnogs OOT[3] are uploaded back to the network. The client continues to operate, waiting for the execution of the next job in the queue.

2.4 Gnuradio OOT module

Each ground station is equipped with an SDR device for the signal reception. The architecture is modular enough, so it can support a wide range of different SDR hardware, depending on the target cost of the station. For example, there are stations using an RTL SDR dongle costing about 15USD, whereas others utilize a high-end device like the USRP B210 with a cost of 2000USD. For the signal analysis and demodulation, the GNU Radio Out-of-Tree (OOT) module called gr-satnogs [3] is used. The software on its core is written in C++, using also some Python bindings. The first task of this module is to receive the signal from the SDR front-end, apply coarse filtering and re-sample it. The later is crucial in order to reduce the sampling rate of the signal originating from the SDR device, so the processing can be performed in CPU limited devices like the Raspberry Pi 3. Afterwards and based on the satellite trajectory, the gr-satnogs compensates the Doppler effect which introduces a constantly changing center frequency offset. Then, the Doppler corrected signal passes additional and more fine grained filtering stages. Finally, the module tries to automatically demodulate the resulting signal in real-time, based on the coding/modulation scheme of the targeted satellite. At the same time, the module produces a waterfall spectrum analysis plot and an audible representation of the spectrum. The waterfall plot is an excellent tool, for immediate and visual spotting of satellite transmissions, nearby interference, possible RF performance issues or misconfiguration at the station setup. The audible transformation of the spectrum, is a technique quite popular in the amateur community and many amateur signal analysis tools utilize it. For each observation, the decoded frames, a waterfall spectrum analysis plot and the audio file are uploaded back to the SatNOGS Network, for visualization and further analysis. Currently, gr-satnogs provides automated demodulators decoders that cover a wide range of satellite missions. From beacons to weather pictures.

3 Conclusion

The use of gnuradio enable an unmatched agility in a groundstation network. The ability to demodulate virtually any kind of transmission is fully future proof. It only requires automated software update of the client. It is also the perfect match with low cost SDR receiver like rtl-sdr.

References

[1] *SATNOGS Database :* `https://db.satnogs.org/`

[2] *SATNOGS network :* `https://network.satnogs.org/`

[3] *gr-satnogs :* `https://gitlab.com/librespacefoundation/satnogs/gr-satnogs`

[4] *SATNOGS client :* `https://gitlab.com/librespacefoundation/satnogs/satnogs-client`

VLBI with GNU Radio, RFNoC and White Rabbit

P. Boven
JIVE, Netherlands

Abstract

1 Introduction

Frequency locking a laser on a spectral hole pattern with a multi-channel heterodyne method using SDR and GnuRadio

N. Galland[1,2], N. Lucic[2], H. Alvarez-Martinez[2], S. Zhang[2] , B. Fang[2], R. Le Targat[2], A. Ferrier[3,4], P. Goldner[3], S. Seidelin[1,5], Y. Lecoq[2]

[1] Université Grenoble Alpes and CNRS, Institut NEEL, F-38042 Grenoble, France

[2] LNE-SYRTE, Observatoire de Paris, Université PSL, CNRS, Sorbonne Université, 61 avenue de l'Observatoire 75014 Paris

[3] Université PSL, Chimie ParisTech, CNRS, Institut de Recherche de Chimie Paris, 75005, Paris, France

[4] Sorbonne Université, 75005, Paris, France

[5] Institut Universitaire de France, 103 Boulevard Saint-Michel, F-75005 Paris, France

Abstract

High precision spectroscopic probing of a narrow spectral hole pattern imprinted in the inhomogeneously broadened absorption spectrum of $Eu^{3+} : Y_2SiO_5$ crystal can be used to stabilize a laser frequency. A multi-hole pattern can be burn and all the holes can be probed simultaneously using a multiple frequency signal. The dispersion induced dephasing acquired by the light through the crystal is measured to derive an error signal suitable to lock the laser frequency to the spectral hole pattern. An Ettus USRP X310 driven by a python program based on GNU Radio is used for both generating the multiple frequency signal and to compute the correction applied to the laser frequency.

1 Introduction

Frequency stabilization of ultra-stable lasers is typically realized using high finesse optical Fabry-Perot cavities (FPC) that can provide a stability of a few 10^{-16} at about 1 s integration time. However, recent development concerning optical lattice clocks appear to require lower frequency noise at this time scale to obtain the best performances of the clock.

A new direction of development for laser stabilization relies on spectral hole burning. The spectral pattern imprinted in a rare earth doped crystal at cryogenic temperature is expected to show higher stability than the brownian motion limited FPC [1], [2].

The spectral hole burning technique relies on two physical processes occuring in $Eu^{3+} : Y_2SiO_5$ crystals. First, the initially narrow $^7F_0 \rightarrow {}^5D_0$ transition of Eu^{3+} is, due to inhomogeneities in the host crystalline matrix, broadened from 120 Hz for an isolated ion to approximately 2 GHz. Secondly, at cryogenic temperatures, a narrow linewidth laser can excite some ions to metastable levels with a lifetime that can reach tens of days at 4 K. By doing so, the absorption is saturated around the laser frequency and a hole is created in the absorption spectrum that can be use as a frequency discriminator for laser stabilization.

2 Experimental setup

To ensure the possibility of spectral hole burning, the europium doped yttrium orthosilicate ($Eu^{3+} : Y_2SiO_5$) crystal is maintained at a cryogenic temperature, typically between 3.2K and 4K, in a commercial close-cycle cryostat with vibration isolation.

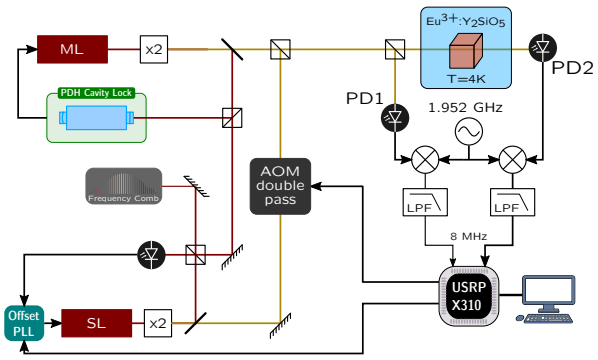

Figure 1: Schematics of the experimental setup. The GNU Radio flowgraph is driving the USRP to generate multi frequency signal for AOM and computing the correction applied to the offset of the PLL from the two RX signals

A two-laser optical system is used to burn and

then probe the spectral hole. The master laser at 1160 nm is pre-stabilized to a high finesse FPC. An offset phase-lock loop is used to servo the slave laser at 1160 nm to the master laser with a controlled offset frequency (typically 980 MHz). Both lasers are independently frequency doubled to reach 580 nm, the center of the broadened transition $^7F_0 \rightarrow {}^5D_0$. The slave laser is then sent to a double-pass AOM to generate an arbitrary spectral pattern in the light (in the 1 MHz range around the center frequency). Both yellow lasers are then spatially overlapped and the resulting beam is splitted in two channels, one going through the crystal and to a 2 GHz bandwith photodiode, the other directly to an identical photodiode for reference.

The beatnote obtained on each photodiode is demodulated to 8 MHz and amplified before acquisition by the two RX channels of an Ettus USRP X310. One TX channel is used to drive the frequency offset between the two lasers. The other TX channel is used to drive the double-pass AOM.

The GNU Radio flowgraph is on the one hand deriving the error signal from the dephasing between RX channels induced by the spectral hole dispersion. A spectral mask is applied to band pass filter the RX signal arround different frequencies and remove any unwanted spectral component. The a proportionnal and integrator filter applied on the error signal provides a correction modulating the TX channel driving the offset frequency. On the other hand, another flowgraph is used to generate an arbitrary spectral pattern applied to the AOM.

3 Results

An optical frequency comb stabilized on a state-of-the-art ultrastable laser allows us to evaluate the stability of our laser over a timescale from 1 s to 1000 s. We can therefore measure the stability of the laser pre-stabilized to our FPC to approximately 10^{-14} fractional frequency instability at 1 s.

By using a double hole pattern with one reference mode (in a wide hole and so experiencing a small dispersion) and one signal mode (in a narrow hole, experiencing a big dispersion as a function ot the detuning), we obtain a good frequency lock of the laser on the narrow hole. The laser then exhibits a stability in the low 10^{-15} for 1 s to 10 s time scale. This result is almost one order of magnitude better than the previous results obtained on this experiment while using a single hole to derive the correction signal [3].

4 Conclusion

The double-hole based detection has improved a lot the detection noise of the detuning induces dispersion. Indeed, the previous detection scheme using only one hole was exhibiting a noise level compatible with a stability in the low 10^{-15} at 1 s. The new scheme shows a detection noise compatible with a stability in the mid 10^{-16} at 1 s.

We then infer that the residual instability of the laser when locked to the spectral hole is not due to the detection noise of the setup, but to other technical issues. Next imporvement will be to reduce the thermal fluctuation of the sample holder in the cryostat and the residual vibration due to the working cycle of the pulsed tube. An investigation to find the optimal spectral pattern will also be conducted to reach the low 10^{-16} at 1 s or below.

This project has received fundings from Ville de Paris Emergence Program, LABEX Cluster of Excellence FIRST-TF (ANR-10-LABX-48-01), within the Program "Investissements d'Avenir" operated by the French National Research Agency (ANR), Région Ile de France; European Union's Horizon 2020 research and innovation program under grant agreement No 712721 (NanOQTech); ANR under grant number 14-CE26-0037-01 DISCRYS; EMPIR 15SIB03 OC18 and from the EMPIR program co-financed by the Participating States.

References

[1] M. J. Thorpe, L. Rippe, T. M. Fortier, M. S. Kirchner and T. Rosenband, "Frequency stabilization to 6×10^{-16} via spectral-hole burning", Nat. Photon. **5**, 688-693 (2011)

[2] D. R. Leibrandt, M. J. Thorpe, C.W. Chou, T. M. Fortier, S. A. Diddams,and T. Rosenband, "Absolute and Relative Stability of an Optical-Frequency Reference Based on Spectral Hole Burning in Eu3+:Y2SiO5", Phys. Rev. Lett. **111**, 237402 (2013)

[3] O. Gobron, K. Jung, N. Galland, K. Predehl, R. Le Targat, A. Ferrier, P.Goldner, S. Seidelin, and Y. Le Coq, "Dispersive heterodyne probingmethod for laser frequency stabilization based on spectral hole burning inrare-earth doped crystals", Optics Express **25** (13), 15539-15548 (2017)

Embedded and Connected Receiver System for nano-satellite in Low Earth Orbit (LEO)

F. Spies[1,2], S. Givron[2]
[1] FEMTO-ST Complex Computer System, Montbéliard, France
[2] Université de Franche-Comté, Montbéliard, France

Abstract

The objective of this work is to design a set of satellite signal reception, embedded, connected and low power consumption. This set must be simple to implement with the ambition of being widely deployed on a global scale to provide complete and continuous coverage so that each satellite transmission can be received at any time without loss. The altitude of the satellite orbit must allow the planet to be covered with less than a hundred reception stations on earth. The stations will be located mainly in universities with an *eduroam* connection to facilitate the transmission of information on a server.

1 Introduction

Communication systems with nano-satellites remain complex to implement. They must have a suitable receiving chain in order to be able to process small amplitude signals. The implementation altitude is around 500 km. The nano-satellite transmission chain must be as light as possible, including the antenna, power amplifier and communication board. The objective of nano-satellites is to be as light as possible with a reduced altitude to limit the cost of placing them in orbit as much as possible. Therefore, the ground station must be able to compensate for the transmission and reception constraints of nano-satellites. It is necessary to be able to amplify weak signals received with motorized directional antennas combined with power amplifiers. The objective of this work is to provide a ground-based reception solution consisting of a simple omnidirectional antenna and a Software Defined Radio (SDR) interface. The advantage of radio software is that it can receive several different nanosatellite signals and can be reprogrammed at any time. In addition, this solution can be updated over time to adapt to changes in modulation in the nano-satellites of next generations. Finally, a study of the dimensioning of the ground reception module will make it possible to reduce the unit cost so that a large-scale deployment can be envisaged.

2 Demodulation Chain

The frequencies mostly used by these nano-satellites are VHF/UHF amateur radio frequencies (145 and 435/438 MHz) or the S band (2.2 to 2.3 GHz). Modulations are generally of the FSK, BPSK or GMSK type. Baud rates are generally between 1200 and 9600 baud. The use of two symbols per period significantly reduces the reception threshold and thus increases the reception distance. One of the advantages of using software radio is that it allows you to reconfigure the demodulation chain for each passage of a nano-satellite. There are software programs such as *gpredict* that integrates the paths of all active in-orbit nano-satellites to predict the path times.

3 Picsat Demodulation

Picsat, a specific nano-satellite put into orbit in 2018, demodulated the signal transmitted in BPSK 1200 baud. The demodulation chain performed by [1] integrated all the demodulation functions necessary to interpret the information transmitted by the satellite and send it back to the server centralizing all the reception on the planet. Indeed, at this altitude, the flight times of an area are generally between 5 and 15 minutes. We understand the importance of being able to multiply the number of reception sites around the world in order to be able to increase the exchange times with the nano-satellite. If ground systems are connected to the Internet, it is possible to gradually move towards a permanent and continuous connection that would limit the redundancy of data transmitted from the satellite and thus increase its ability to transmit information to the ground. All you need is between 80 and 100 ground receiving systems evenly distributed to be in permanent communication with the nano-satellite. A maximum distance between 2 reception systems of about 3500 km (30 degrees) would allow this continuity of communication.

4 Systems Implementation

Before attempting to embed the receiving chain on low-cost electronic boards, it is preferable to vali-

date oversized solutions to facilitate development. Thus, we used a laptop computer, a Raspberry Pi 3, a Raspberry Pi zero and a microcontroller ESP8266 [2] including a Wi-Fi interface to perform the signal demodulation. The signal reception was performed with a USRP B200 mini, an ADALM-Pluto and an RTL-SDR.

On laptops and Raspberry PIs that use a UNIX system, the use of gnuradio is possible. On Raspberry PI boards, computing power is limited, the USB interface works in version 2 and the use of command line gnuradio is preferable. On the microcontroller, the program is downloaded from the Arduino development interface in C/C++/LUA language.

5 Signal Generator

When designing a demodulation function, it is necessary to decode a good quality signal, but this is not enough. It is also necessary to be able to measure the qualities and characteristics of the algorithm designed to identify the potential of the reception sensitivity threshold obtained. If the reception sensitivity threshold is too high, decoding will only be possible when the nano-satellite is near the zenith of our position. If the sensitivity threshold is low enough, decoding can be carried out up to positions close to the horizon where the link budget is most unfavourable.

We have produced several types of files containing signals including increasingly large attenuations and small constant, linear and polynomial frequency shifts to gradually reproduce the Doppler effect of LEO orbits.

6 Decoding Tests

First, we validated that reception could be achieved with an omnidirectional antenna placed horizontally in the perpendicular of the satellite path connected to a USRP B200 mini interface. The whole thing was connected to a laptop PC with gnuradio/gr-picsat [1] to ensure sufficient computing power and memory capacity. The reception time is approximately 8 minutes out of a potential 10 to 12 minutes. This first result was considered sufficient to qualify the reception.

In a second step, we validated the computing and memory capacity of the Rapsberry Pi zero Wi-Fi (ARM 1 core 1GHz 512GB RAM) to demodulate the I/Q signals received in gnuradio/gr-picsat in command line mode.

In a third step, the demodulation was coded directly into the micro-code of the ESP8266 to allow the restitution of the transmitted data. Software radios have a USB port and can only be connected to the ESP8266 at a low speed equivalent to version 1.0 or 1.1 of about 1Mbps. We have chosen to use the radio software in SDR-IP mode (Fig. 1), i.e. through a Wi-Fi/IP connection to validate the processing capacity of the microcontroller. Data reception centered on the corrected frequency could be completely integrated, in purely integer functions for more efficiency. In order to limit the energy consumption by the entire station, it is only executed when a nano-satellite passes through and remains in deep sleep between two passes. The times of the visits are obtained on servers located on the Internet.

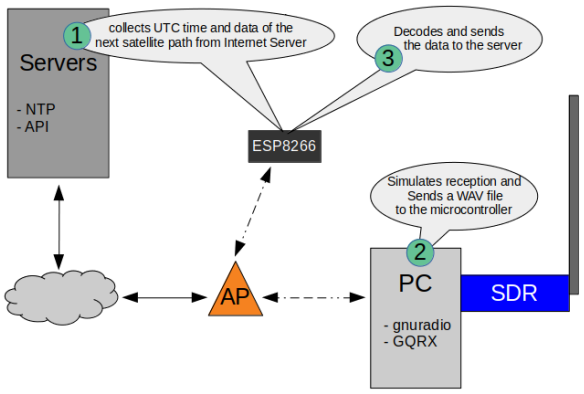

Figure 1: Global Architecture with the Wi-Fi microcontroller ESP8266.

7 Conclusion

Porting demodulation functions into a microcontroller has allowed us to improve the overall efficiency of the receiving chain. The Doppler effect correction remains to be integrated into the microcontroller to complete the demodulation chain. We finally chose to reintegrate the entire demodulation functions into gnuradio to reduce processing time and use a low-power Raspberry Pi zero that easily connects an SDR interface.

References

[1] *PicSat telemetry parser added to gr-satellites* at `https://destevez.net/2018/01/picsat-telemetry-parser-added-to-gr-satellites/`

[2] *ESP8266: Wi-Fi microchip with full TCP/IP stack and microcontroller* at `https://en.wikipedia.org/wiki/ESP8266`

Using GNU Radio to do signal acquisition and analysis with Scopy

A. Suciu
Analog Devices Inc.

Abstract

GNU Radio is the go-to library when it comes to open source software-defined radio. However GNU Radio can go beyond that. In this paper we will discuss the use of GNU Radio as a base library for an end product application that requires general signal processing as well as other decoding libraries. Specifically, we will address how GNU Radio can be leveraged for applications beyond software defined radio and other communication systems.

1 Introduction

In order to create Scopy – an open source mixed signal analysis and generation toolkit, we chose to use GNU Radio, along with a variety of open source libraries and frameworks such as libiio, libsigrok, Qt5 or Qwt. Scopy currently interfaces with the ADALM-2000 hardware which provides two analog input channels, two analog output channels, as well as 16 digital I/O pins – capable of high speed synchronized buffered operations. Future plans include extending Scopy to interface with other hardware.

2 Experimental setup

Scopy uses `libiio` to interface with ADALM-2000. This allows Scopy to connect to the hardware via USB, as well as ethernet. By using an off the shelf (COTS) Wi-Fi dongle, the hardware can connect to a wireless network and Scopy can acquire data remotely from the ADALM-2000.

GNU Radio is used in this context to multiplex the data streams received from the hardware via `gr-iio` to the oscilloscope/ spectrum analyzer/ network analyzer. GNU Radio's efficient vector-optimized operations are used to implement instrument functionalities such as the oscilloscope reference waveform, digital AC coupling as well as math channels. The network analyzer uses the GNU Radio flow to create a full network analyzer signal chain. Combined with the M2K hardware it is able to characterize circuits up to 30MHz and represent the results on Bode, Nichols and Nyquist plots. The spectrum analyzer allows marker operations as well as different types of windowing up to 50MHz. The signal generator uses GNU Radio to output various types of user configurable signals such as sine, square waves or the results of time-dependant mathematical equations.

3 Results

We used GNU Radio for an end application that does not comply with the traditional scope of the framework. Although GNU Radio is well modularized, starting and stopping instruments has been one of the challenges we faced. Since there is no trivial way to reconfigure a GNU Radio flow, we had to develop a method that recursively deletes blocks starting with a parent. This and the use of the copy block eased up flow reconfiguration. Gnuradio is a good fit for this application as it abstracts the complexities of signal acquisition and analysis into an efficient data flow giving us more headroom to develop a more user friendly, touchscreen compatible

4 Background

The ADALM-2000 also known as M2k [1] is a Software Defined Measurement Platform that is a cross platform (Windows, Mac and Linux) USB oscilloscope and multi-function instrument that allows users to measure, visualize, generate, record, and control mixed-signal circuits of all kinds. It features two-channel digital oscilloscope and arbitrary function generator, with Time, Network and Spectrum Analyzer views. A 16- channel digital logic analyzer and pattern generator, with a countless number of Bus Analyzers for all kind of protocols such as I2C, SPI, UART, CAN, JTAG, SPDIF, etc. just to mention a few. Besides this it also has true RMS voltmeters and programmable power supplies all in a pocket sized instrument.

References

[1] *ADALM2000 for Developers* at `https://wiki.analog.com/university/tools/m2k/developers`

Fully Digital Electronics for Fiber-Link Frequency Transfer Implemented on Red Pitaya

A. C. Cárdenas-Olaya, C. E. Calosso
INRiM, Quantum Metrology & Nanotechnology, Turin, Italy

Abstract

This paper presents a digital instrumentation for frequency transfer on optical fiber links. The proposed system detects the phase and amplitude of the beatnotes at the two ends of the fiber for (actively or passively) compensating by the phase noise and the polarization rotation. The implementation is performed on Red Pitaya, an open source platform that integrates fast Analog-to-Digital and Digital-to-Analog converters with a Zynq System-on-Chip. The system features a detection bandwidth of 10 MHz, compatible with thousand kilometers links, that can be finely tuned for reaching an adequate Signal-to-Noise Ratio minimizing the generation of cycle slips.

1 Introduction

Fiber links have been demonstrated to be a suitable medium for optical frequency transfer featuring an stability of the order of $10^{-19} \div 10^{-21}$ [1]. However, mechanical stress, environmental factors and temperature fluctuations generate fiber length variations that degrade the frequency stability of the signal being transmitted. In order to compensate for such fiber noise, two techniques are widely used: the classical Doppler [2] and the more recent Two-Way [3]. The implementation of these techniques relies on the detection of the phase and the amplitude information contained in the beatnote between the received and the transmitted signals which is obtained through photo-detection at the local and the remote ends. There are three critical implementation requirements that may limit the adequate noise compensation and consequently the frequency transfer performance: First, the detection bandwidth that must be wide enough for extracting the phase information before it gets corrupted by the noise. In current applications, it is of the order of hundreds of kilo Hertz which limits implementations on long links because the phase can not be properly tracked. Second, additive noise is generated during the beatnote photo-detection, which is revealed as excess of noise in the measurement degrading the Signal-to-Noise Ratio (SNR) and leading to the generation of cycle slips. This effect is in general removed by reducing the detection bandwidth. And finally, the polarization rotation that should be maintained in order to avoid amplitude fluctuations on the detected beanote that could degrade the SNR.

In this work we present a fully digital instrument for frequency transfer over fiber link. The system provides the capability of finely tune the detection bandwidth from 10 MHz down to 1 kHz, allowing to reach the best tradeoff between the SNR and the detection bandwidth for minimizing the cycle slips rate. In addition, it implements the simplex algorithm, yielding a polarization control bandwidth up to 100 kHz.

A fiber emulator is implemented on the FPGA with the aim of studying the critical requirements (detection bandwidth, SNR, polarization) under different scenarios (noise typologies and levels), while taking advantage of the System-on-Chip (SoC) bandwidth for long term analysis. In particular for the evaluation of cycle slips whose rate in current implementations is one cycle slip per one or two minutes.

Preliminary results of the instrument are intended to be presented at the workshop.

2 Method

The proposed system, depicted in Fig. 1, was implemented on Red Pitaya [4]. The platform integrates a 14-bit, 125 MSps, dual channel Analog-to-Digital Converter (ADC), a 14-bit, 125 MSps, dual channel Digital-to-Analog Converter (DAC) and a SoC containing a FPGA and a Dual-core ARM processor.

The instrument subsamples directly the beatenote from the photo-detection and retrieves the phase (φ) and amplitude (α) information through an I/Q demodulator in a 10 MHz bandwidth which can be finely tuned down to 1 kHz. The phase, if actively compensated, is tracked by a servo that drives the Acousto-Optic Modulator (AOM) through a Numerically Controller Oscillator (NCO). The amplitude feeds a simplex algorithm that optimizes the polarization between the received and the transmitted signal for obtaining the maximum power on the beatnote.

A two-channel system is implemented on a single Red Pitaya which allows compensation and monitoring at each side. The useful data are stored in

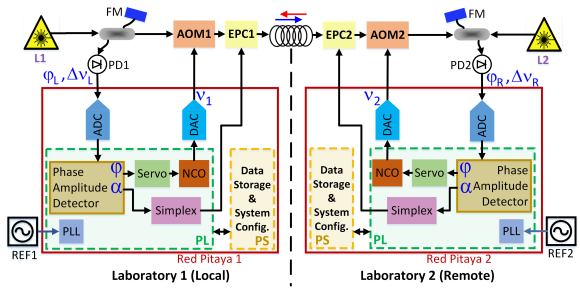

Figure 1: General scheme. PD: Photo-detector, FM: Faraday Mirror, EPC: External Polarization Controller, PL: Programmable Logic (FPGA), PS: Processing System (ARM processor).

memory and transferred to an external computer for post-processing.

Fig. 2 sketches the block diagram of one channel including the fiber emulator. The emulator generates the main noise sources involved in a setup for frequency transfer such as: fiber noise, photo-detection noise and noise induced by the polarization. In addition, it is provided with the emulator of a polarization controller based on a four plates configuration. When the emulator is enabled, it acts on the NCO signal (DAC input) that instead of driving the AOM will be the instrument input, emulating the beatnote.

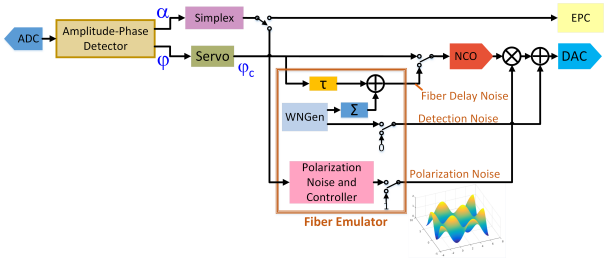

Figure 2: One channel system. τ: Fiber delay, WNGen: White Noise Generator, EPC: External Polarization Controller.

3 Conclusion

The proposed digital instrument is a compact solution that integrates different functionalities such as phase-meter, stability control and polarization control in a single platform.

The instrument is provided with the capability of applying different configurations of the two compensation techniques by properly setting the different blocks parameters, such as: the compensation action (open or close loop), compensation bandwidth (if any), the frequency of the beatenote detected that differs a factor of two from Doppler to Two-Way, the detection bandwidth, the data channels to be stored, etc.

References

[1] K. Predehl et al., *A 920-kilometer Optical Fiber Link for Frequency Metrology at the 19th Decimal Place*, Science, April 2012, **336(6080)**

[2] L.-S. Ma, P. Jungner, J. Ye and J. L. Hall, *Delivering the same optical frequency at two places: accurate cancellation of phase noise introduced by an optical fiber or other time-varying path*, Optics Letters, Nov. 1994, **19(21)**

[3] C. E. Calosso, E. Bertacco, D. Calonico, C. Clivati, G. A. Costanzo, M. Frittelli, F. Levi, A. Mura and A. Godone, *Frequency transfer via a two-way optical phase comparison on a multiplexed fiber network*, Optics Letters, March 2014, **39(5)**

[4] Red Pitaya, *Documentation and technical information*, `http://redpitaya.readthedocs.io/en/latest/`

KiwiSDR as a new GNURadio source

C. Mayer

Abstract

By now, >300 world-wide distributed KiwiSDRs are online. After introducing the KiwiSDR, an implementation of a GNURadio websocket client for the KiwiSDR. As an example on how to use the GNURadio KiwiSDR client, the coherent combination of KiwiSDR IQ streams, and decoders implemented in GNURadio for digital HF modes using the KiwiSDR client are shown.

1 Introduction

The KiwiSDR [1] was developed by John Seamons ZL/KF6VO and by now >300 of these SDRs are available on the internet [2]. In this contribution, a GNURadio websocket client for the KiwiSDR [3] is presented, along with some applications [4].

2 Coherent combination of KiwiSDR IQ streams

The maximum sampling rate for IQ data streams available from KiwiSDRs is either 12 kHz or 20.25 kHz. In order to aqnalyze signals with wider bandwidth, a method was developed which allows to coherently combine up to 6 IQ data streams from a single KiwiSDR [3]. This method uses precise GNSS-derived timestamps to align samples from the same KiwiSDR. Remaining phase offsets between the IQ data streams are estimated and corrected for, see Figure 1.

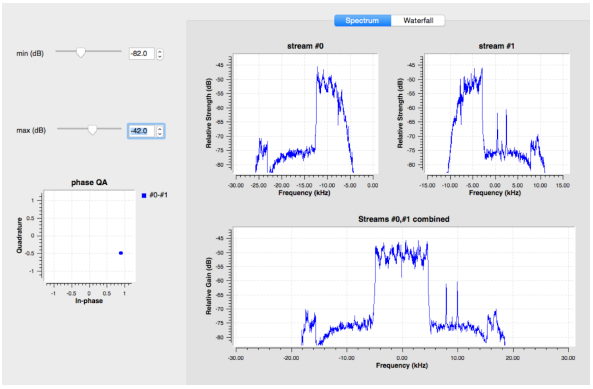

Figure 1: Coherent combination of two KiwiSDR IQ data streams for the case where the boundary between the streams is centered on a DRM signal.

3 Decoding digital modes on HF using GNURadio

A GNURadio OOT module is being developed which allows to concisely specify the physical layer characteristics of a given phase-modulated digital mode on HF in python [5], where preamble cross-correlation, doppler estimation and adaptive filtering are taken care of the framework. As an example the demodulation of a STANAG 4285 signal is shown in Figure 2.

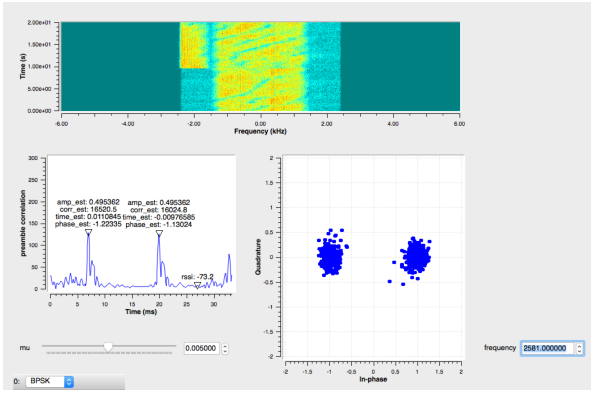

Figure 2: STANAG 4285 signal demodulation using a KiwiSDR IQ data stream. The constellation diagram shows symbols after adaptive filtering and after descrambling.

4 TDoA with KiwiSDR

The IQ data streams from KiwiSDRs contain precise GNSS-derived time stamps. Using this absolute time reference, time difference of arrival (TDoA) methods can be explored. Currently the code for the KiwiSDR TDoA analyis is written in Octave, see [6], which is used by the KiwiSDR TDoA extension written by John Seamons.

Figure 3 shows cross correlations between a LORAN signal received by two KiwiSDRs, and Fig. 4 shows the TDoA map obtained by using the cross correlations from three KiwiSDRs.

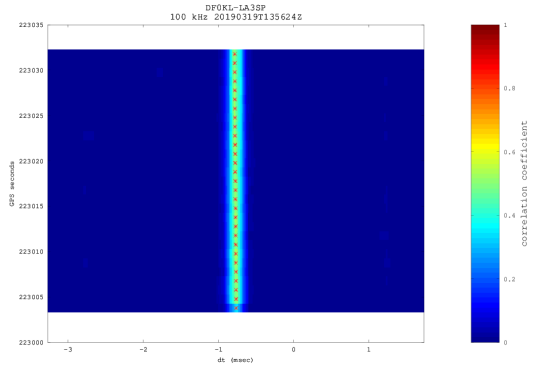

Figure 3: Cross correlations between two KiwiSDRs (LORAN).

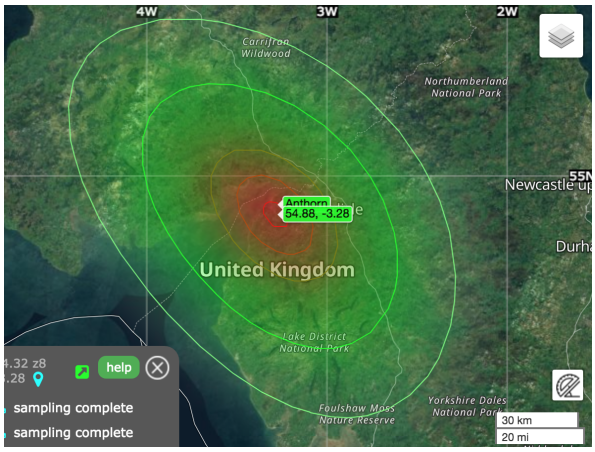

Figure 4: TDoA map for the LORAN signal from Anthorn, obtained using data from three KiwiSDRs.

References

[1] *KiwiSDR* at `http://kiwisdr.com`

[2] *Map of available KiwiSDRs* at `http://rx.linkfanel.net/`

[3] *gr-kiwisdr* at `https://github.com/hcab14/gr-kiwisdr`

[4] *Singal Analysis and Monitoring blog* at `https://hcab14.blogspot.com/`

[5] *gr-digitalhf* at `https://github.com/hcab14/gr-digitalhf`

[6] *TDoA* at `https://github.com/hcab14/TDoA`

Phase noise & digital noise: what is it and why it is important for groundbreaking RF-applications.

P.-Y. Bourgeois
FEMTO-ST, Time&Frequency dpt, UMR 6174 CNRS, Besançon, France

Abstract

For more than a decade, digital electronics has a massive impact on about every research field offering flexibility, robustness and reconfigurability.

1 Introduction

Numerous developments of complex scientific instrumentation now employ routinely partial or fully digital systems, inherited from telecommunications and Software Defined Radio (SDR).

Within the framework of Time & Frequency metrology, it has become challenging in the development of modern sensitive instrumentation where quantization noise and signal noise paths take a significant place for qualifying ultrastable clocks, oscillators, frequency transfer and timing systems.

From Amateur Radio to VLBI (Very-long-baseline interferometry), DSN (Deep Space Networks), LIGO(Laser Interferometer Gravitational-Wave Observatory)[1], Evolved LISA(Laser Interferometer Space Antenna)[2], ACES (Atomic Clock Ensemble in Space), ..., it is worth to note that some knowledge on instrumentation limitations and their proper characterization is of importance.

Unfortunately there is a lack on handy design tools and techniques for complex designs leading to high computational efforts in programming and prototyping that are ultimately greatly error prone.

Although a few of valuable libraries and software do exist (GNURadio[3], Spiral[4], liquidsdr[5]...), they are currently not adapted to oscillator metrology modeling (parametric noise simulations, variances, normalized spectra ...).

Along the journey, I propose to present different techniques and pitfalls in the measurement and qualification of the digital signal processing chain kernel (Signal→ADC→DDC) commonly used in every SDR environment[6, 7].

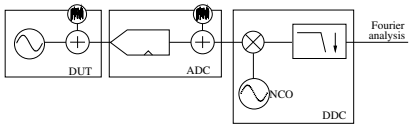

Figure 1: Elementary digital processing chain for oscillator metrology

I will also show how it is currently difficult to perform such characterizations within the GNU Radio context.

References

[1] B P Abbott *& al.* Characterization of transient noise in advanced LIGO relevant to gravitational wave signal GW150914. *Classical and Quantum Gravity*, 33(13):134001, jun 2016.

[2] He-Shan Liu, Yu-Hui Dong, Yu-Qiong Li, Zi-Ren Luo, and Gang Jin. The evaluation of phasemeter prototype performance for the space gravitational waves detection. *Review of Scientific Instruments*, 85(2):024503, 2014.

[3] Gnu radio. gnuradio.org.

[4] Franz Franchetti, Tze Meng Low, Doru Thom Popovici, Richard M. Veras, Daniele G. Spampinato, Jeremy R. Johnson, Markus Püschel, James C. Hoe, and José M. F. Moura. Spiral: Extreme performance portability. *IEEE special issue on From High Level Specification to High Performance Code*, 206(11), 2018.

[5] Liquidsdr. liquidsdr.org.

[6] P-Y Bourgeois, G Goavec-Merou, J-M Friedt, and E Rubiola. A fully-digital realtime soc fpga based phase noise analyzer with cross-correlation. In *Joint EFTF/IFCS*, pages 578–582. IEEE, 2017.

[7] B Marechal, A Hugeat, G Goavec-Mérou, G Cabodevila, J Millo, C Lacroûte, and PY Bourgeois. Digital implementation of various locking schemes of ultrastable photonics systems. In *2018 IEEE International Frequency Control Symposium (IFCS)*, pages 1–4. IEEE, 2018.

Using GNU Radio Companion to improve student understanding of signal processing theory through VHF Omni-Directional Range (VOR) signal demodulation

A. Blais, C. Morlaas
ENAC, Université de Toulouse, France

Abstract

The École Nationale de l'Aviation Civile (ENAC), the French Civil Aviation University, proposes a graduate engineer program with, among three others, a *major*, in Aeronautical and Space Telecommunications (SAT). This *major* is characterized by advanced theoretical courses in signal theory, signal processing, digital communications and navigation in particular. Some practicals are given to help the students understanding specific points in these courses. However, this scattered approach does not provide any hindsight on the interest and the usefulness of these courses, either independently nor together. This is the reason why these practicals are completed by a long project which proposes, as a global case of application, to demodulate a VHF Omni-Directional Range (VOR) signal. This paper details this long project, from its pedagogical approach to some interesting points of its implementation.

1 Introduction

The sometimes complex notions being taught in signal theory and signal processing courses need to find some practical realizations to be fully understood by the students. It is the objective of the long project described in this paper. Indeed, it proposes the demodulation of a VOR signal using the GNU Radio Companion (GRC) software. The simultaneous learning of this powerful software is an interesting aside of this project, even if it is not its primary goal. The signal in question is either received by mean of a DVB-T USB dongle (one is distributed per pair of students) or a synthetic one, generated in the course of the project by the students themselves. In the first part of this paper, a model of the VOR signal is presented. Then, the pedagogical approach selected by the teaching team is described. Two interesting technical points are adressed in the third part, to point out the signal processing challenges the students have to face. At last, a brief conclusion is drawn.

2 A VOR signal model

The VOR system is a navigation aid designed to give the angle, in relation to the magnetic north, to the aircraft from a ground radio beacon of known position. This is done with the help of an hybrid signal carrying several components. A noise-free time model of this signal [1], as seen by the receiver, is presented to the students:

$$e(t) = (1 + M(t)) \cos(2\pi f_0 t + \theta_0) \quad (1)$$

$$\begin{aligned} M(t) = {} & 0.3 \cos(2\pi 30t - \mathrm{QDR} + \theta_{30}) \\ & + 0,05\ \mathrm{ident}(t) \cos(2\pi 1020t + \theta_{1200}) \\ & + 0.3 \cos(2\pi 9960t + \theta(t) + \theta_{9960}) \end{aligned} \quad (2)$$

$$\begin{aligned} \theta(t) &= 16 \cdot 30 \cdot 2\pi \int_0^t \cos(2\pi 30u + \theta_{30}) du \\ &= 16 \cdot \sin(2\pi 30t + \theta_{30}) - 16 \cdot \sin(\theta_{30}) \end{aligned} \quad (3)$$

with:

- $f_0 \in [108, 118]$ MHz, the carrier frequency,
- QDR the magnetic bearing from the station to the aircraft, in units of radian,
- ident(t) the identification signal of the VOR beacon (Morse code),
- θ_{30}, θ_{1200} and θ_{9960} random initial phases.

A graphical representation of the spectrum corresponding to the amplitude modulation $(1 + M(t))$, figure 1, is also shown to train the students to think in the frequency domain as well as in the time domain.

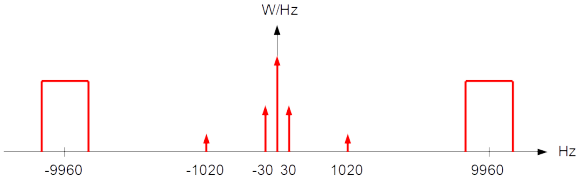

Figure 1: Schematic representation of the baseband spectrum of a VOR signal.

The models of the I & Q signals delivered by the DVB-T dongle are then derived from equations (1), (2) and (3)), after a short explanation of how this dongle works. This is an effective use of the complex envelope concept introduced in the digital communications course. These models are the starting point of the practical implementation.

3 The pedagogical approach

Strong guidelines are set to the students to support them toward the solution. Indeed, the project is split in four distinct and successive steps, each one improving gradually the skill and knowledge of the student:

1. The first step is an interactive implementation of a simple FM radio-broadcast receiver. This first SDR receiver serves as a presentation of the GRC GUI capabilities as well as a proof of the SDR concept for the students: it works!

2. The target of the second step is a more comprehensive handling of GRC and of the USB dongle. The students have to produce flow diagrams which read, display and record to a file the I & Q samples.

3. Then, to deepen the students understanding of the VOR signal structure, as well as to provide the future receiver with synthetic test signals, a I & Q VOR signal generator has to be implemented, component after component. The generated test signals are validated with the SDR receiver of the teaching team.

4. Finally, the VOR receiver itself has to be developed. Again, a component by component method is suggested to the students, so they can safely move towards a proper final design.

The project ends with a double evaluation: a one hour written examination and a test of reception of a real VOR signal with a QDR to decode. The latter test is realized by radiating a low power VOR signal in the room with the help of a Marconi 2030 signal generator.

4 Two specific technical points

The philosophy of this long project being to improve the understanding of the students in signal theory and signal processing, the use of high level blocks, like the `FM Demod` block, is forbidden. The students have then to understand the processings to be able to build them atomically. In this aspect, the two following ones are particularly interesting.

4.1 The frequency demodulation

Figure 2 illustrates the demodulation principle the students not only have to implement, but also to understand! The calculation of the value of the delay is always an instructive brainstorm.

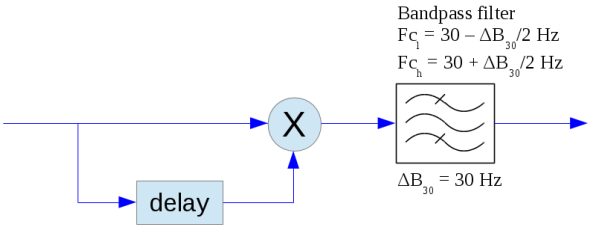

Figure 2: Synoptic view of the FM demodulator.

4.2 The phase comparator

The phase comparator, needed to extract the QDR, the value of interest, is an opportunity to introduce the students with the principle of the `Hier` block. They are asked to embedded this phase comparator in a dedicated `Hier` block, to be tested and validated independently from the main flow diagram. Figure 3 displays a possible implementation for this phase comparator `Hier` block.

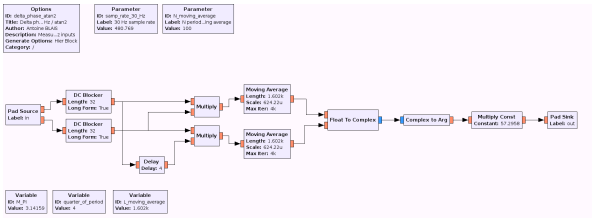

Figure 3: Flow diagram of the phase comparator

5 Conclusion

It is the third year this long project is given. The teaching team has observed each time the benefits of this materialization of the theoretical notions taught during the related courses. The knowledge and skill of the students are clearly improved as well as their hindsight on the covered topics.

References

[1] Subdivision Radionavigation conventionnelle et par satellite. VOR Généralités. Technical report, École Nationale de l'Aviation Civile, 2010.

A LoRaWAN Security Assessment Test Bench

T. Claverie, J. Lopes Esteves
Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)
Paris, France

Abstract

LoRaWAN is a recent protocol and despite having been already studied from a security perspective, several attacks have not been reproduced in practice mostly due to a lack of details regarding the test benches used. After presenting previous work on the LoRaWAN protocol and the various platforms described, we present an environment based on hardware, software and SDR to study the radio layer of the protocol. The efficiency of this architecture is demonstrated by reproducing theoretical attacks on LoRaWAN 1.0.

1 Introduction

LoRaWAN is a protocol dedicated to Low Power IoT devices; its current version (1.1) was released in October, 2017 [5]. It is built upon the LoRa (Long Range) modulation patented by Semtech.

In [4] a *replay and decrypt* attack is detailed on LoRaWAN 1.0 and the authors provide a very accurate and complete overview of the state of the art up to late 2017. A formal verification of the security of the LoRaWAN handshake has been proposed in [8]. In [19], the biasing of a random number generator in presence of electromagnetic interferences is demonstrated. Session cryptographic material desynchronisation between a device and the network has been discussed in various articles [4, 19, 14, 11, 12]. The possibility of spoofing LoRaWAN gateways in order to provide fake time and position references appeared in [12, 20]. Message modification and forgery in order to attack the network has been suggested in [15, 10].

Several LoRaWAN security testing environments have already been mentioned in previous work. However, none have been described precisely enough to allow reproductibility of the results and some do not involve instrumentation of the radio layers.

In [17, 18], the author describes using a LoRaWAN gateway, two test end devices and an open-source LoRaWAN server to emulate a complete infrastructure. However, these tests concerned only the communication in the core network, beyond the gateway. A testing platform is mentioned in [12]; it is able to capture and analyze messages, however no information on its exact capabilities and building blocks are provided. In [11], hardware attacks on devices and radio attacks are mentioned. In particular, eavesdropping of LoRaWAN communications, replay attacks, and various denial of service by flooding an object with messages are described. Again, the test infrastructure description does not allow reproducing the setup.

Even when specific radio testing has been investigated, the information regarding the test setup were left aside: no details on the jamming methods [3, 6], nor on how a LoRaWAN device is perturbated by electromagnetic interference [7] were given.

2 Experimental setup

In this section, the different components used for our test bench are introduced.

LoRaWAN Evaluation kit In order to rapidly deploy a LoRaWAN 1.0 network without having to develop the core network services, a Microchip LoRaWAN development kit [13] has been set up. This development kit includes two LoRa Mote (based on the SX1276 chipset) and a LoRa gateway (SX1301) which can be connected via ethernet to core network services packaged in a ready-to-use docker container. The administration interface allows to manage gateways, applications, devices, encryption keys and some radio parameters (frequency, modulation). It constitutes a ready-to-use LoRaWAN infrastructure but is intended only for high level interaction with the protocol and thus lacks low layer flexibility.

LoRa programmable dongle A PyCom FiPy board has been used to act as a malicious LoRa node. It is a MicroPython development board supporting LoRa modulation and including a LoRaWAN stack. With this tool one can send and receive LoRa MAC frames with a better timing accuracy and lower latency than the LoRa Mote and act as both a gateway and a device.

GNU Radio In order to also have physical layer instrumentation, the GNU Radio framework was used along with a software defined radio (SDR) USB dongle based on a Realtek RTL2832U and a R820T2 tuner. Several SDR-based LoRa receivers are available, with varying completion rates [16, 2, 1].

The gr-lora block from [9] provided very good results along with detailed explanations. However significant latency is induced by the decoding layer and it only supports demodulating uplink or downlink transmissions at once. A minor change has been made to enable decoding both uplink and downlink messages within a single LoRa decoder block.

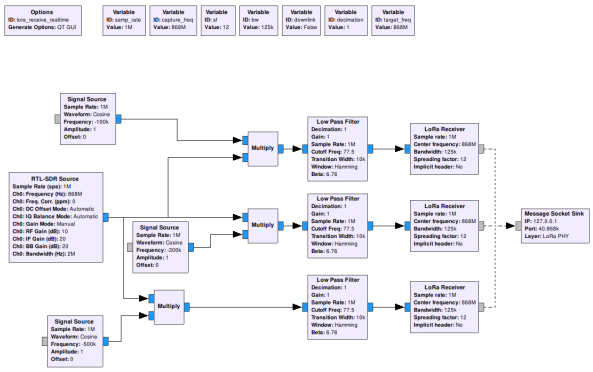

Figure 1: GNU Radio flowgraph for capturing LoRaWAN communications on 3 channels

Using this block, we implemented a GNU Radio LoRa decoder which listens on several channels and forwards the decoded frames to a UDP port (Fig. 1). This strategy allowed receiving all 21 LoRa channels in parallel. Along with the traditional waterfall visualization of the spectrum, the use of GNU Radio allows debugging the physical layer of aformentioned radio transmitters.

3 Reproducing the replay or decrypt attack

The LoRaWAN infrastucture from the development kit was used as a test network. Being easily programmable, the FiPy was used to attack. The RTL-SDR dongle with LoRa decoder block was used to monitor communications, understand the behavior of the devices and investigate in case of problems.

Attack implementation This setup reimplements various radio tests on a LoRaWAN network. In particular, it was possible to reimplement the *replay or decrypt* attack on the LoRaWAN protocol 1.0 described in [4].

This attack leverages a nonce reuse (*DevNonce*) in the handshake protocol between a device and the network: The device sends a *JoinRequest* message and the network replies with a *JoinAccept* message. After that, they both have shared cryptographic material and can start sending data frames to each other. The attack scenario is the following: the attacker listens and waits until he captures a full join handshake from the targeted device. He captures messages in this first session.

When the device initiates a new join procedure, the attacker spoofs the gateway and forces a reinitiation until the *DevNonce* sent by the device is the same as in the captured session; the attacker then responds with the captured *JoinAccept*.

Once the join procedure succeeds, the device starts sending data frames. Due to the nonce reuse, they will be encrypted with the same keystream as the captured session, which allows an attacker to partially break the confidentiality of the messages.

Results Our implementation of the *replay or decrypt* attack took about three days to complete, with around 2^{15} *DevNonce* tested and a ten-second delay between two trials. With this setup, it is also possible to implement several desynchronisation attacks described in [4, 19, 14, 11, 12].

More generally, this combination of ready-to-use infrastructure and SDR proved invaluable for security testing the LoRaWAN protocol. This setup can also be used to monitor the LoRaWAN communications and develop detection heuristics for these attacks. An unusual spectral occupation infringes the specifications and may indicate a problem or an attack.

4 Conclusion

In this paper we described a low cost security assessment platform for the LoRaWAN protocol. We provided a precise description of the building blocks in order to guarantee accurate reproducibility of the test conditions.

Furthermore, we validated our strategy by very quickly implementing a theoretical attack on LoRaWAN 1.0 (which has been fixed in LoRaWAN 1.1). The theoretical results have been reproduced and the estimations of the attack cost and complexity were confirmed.

The hybrid radio approach involving a ready-to-use development kit, a progammable radio dongle and a software defined radio has shown interesting benefits in this case, combining the advantages of each platform while compensating for their drawbacks.

In particular, it provides an interesting framework to analyze the impacts of attacks on all layers, such as jamming or intentional electromagnetic interference, and to test protocol stacks.

References

[1] Lora-sdr, 2016. https://github.com/myriadrf/LoRa-SDR.

[2] rtl-sdrangelove, 2016. https://github.com/hexameron/rtl-sdrangelove.

[3] Emekcan Aras, Nicolas Small, Gowri Sankar Ramachandran, Stéphane Delbruel, Wouter Joosen, and Danny Hughes. Selective Jamming of LoRaWAN Using Commodity Hardware. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous 2017, pages 363–372, New York, NY, USA, 2017. ACM. event-place: Melbourne, VIC, Australia.

[4] Gildas Avoine and Loïc Ferreira. Rescuing LoRaWAN 1.0. Technical Report 651, IACR, 2017.

[5] LoRa Alliance Technical Committee. LoRaWAN 1.1 Specification, 2017.

[6] S. M. Danish, A. Nasir, H. K. Qureshi, A. B. Ashfaq, S. Mumtaz, and J. Rodriguez. Network Intrusion Detection System for Jamming Attack in LoRaWAN Join Procedure. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2018.

[7] S. M. Danish, H. K. Qureshi, and S. Jangsher. Jamming Attack Analysis of Wireless Power Transfer on LoRaWAN Join Procedure. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, December 2018.

[8] Mohamed Eldefrawy, Ismail Butun, Nuno Pereira, and Mikael Gidlund. Formal security analysis of lorawan. *Computer Networks*, 148:328 – 339, 2019.

[9] Matt Knight. gr-lora, 2016. https://github.com/BastilleResearch/gr-lora.

[10] JungWoon Lee, DongYeop Hwang, JiHong Park, and Ki-Hyung Kim. Risk analysis and countermeasure for bit-flipping attack in LoRaWAN. In *2017 International Conference on Information Networking (ICOIN)*, pages 549–551, Da Nang, January 2017.

[11] Franck L'Hereec and Nicolas Joulain. Sécurité LoRaWAN. In *Computer & Electronics Security applications Rendez-vous (C&ESAR) 2016*, pages 92–108, Rennes, France, 2016.

[12] Renaud Lifchitz. Security review of LoRaWAN networks. In *Hardwear.io*, The Hague, Netherlands, 2016.

[13] Microchip. Lora technology evaluation kit, 2018. https://www.microchip.com/DevelopmentTools/ProductDetails/DV164140-1.

[14] Robert Miller. LoRa Security - Building a Secure LoRa Solution. Whitepaper, MWR Labs, 2016.

[15] Robert Miller. LoRa the explorer: Attacking and Defending LoRa Systems. In *Syscan 360 Singapore*, Singapore, 2016.

[16] Pieter Robyns. gr-lora, 2018. https://github.com/rpp0/gr-lora.

[17] Sébastien Roy. Lorawan: Déploiement d'une infrastructure de test - partie 1/2. Guide technique, MISC Mag, 2018. https://www.miscmag.com/lorawan-deploiement-dune-infrastructure-de-test-partie-1-2/.

[18] Sébastien Roy. Lorawan: Déploiement d'une infrastructure de test - partie 2/2. Guide technique, MISC Mag, 2018. https://www.miscmag.com/lorawan-deploiement-dune-infrastructure-de-test-partie-2-2/.

[19] S. Tomasin, S. Zulian, and L. Vangelista. Security Analysis of LoRaWAN Join Procedure for Internet of Things Networks. In *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6, March 2017.

[20] X. Yang, E. Karampatzakis, C. Doerr, and F. Kuipers. Security Vulnerabilities in LoRaWAN. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 129–140, April 2018.

A 60GHz digital link with GNU Radio and USRP radios

H. Boeglen[1]

[1] XLIM Laboratory, UMR CNRS 7252, Limoges, France
University of Poitiers

In this communication we present a 60 GHz radio link for Ettus USRP radios. The link is built around dedicated integrated circuits from Analog Devices. The transmission chain is validated with a 100Mbps OFDM system designed with GNU Radio.

Keywords: Millimeter wave communications, RF front-ends

1 Introduction

Most Ettus USRP SDR equipments can be fitted with RF daughter boards featuring different radio ranges and bandwidths. The maximum RF frequency available for Ettus boards is currently 6 GHz. Due to the overcrowded RF spectrum and to enable very high data rates (i.e. several Gbps) millimeter wave bands are being envisaged for 5G telephony. The forecasted bands are 26, 40 and 60GHz. Millimeter wave bands have specific characteristics which have to be evaluated in real transmission conditions. It is therefore necessary to have measurement equipments able to work in these bands. The goal of this paper is twofold. Firstly, to present the constraints associated with millimeter bands and to look for existing solutions adaptable to USRP radios. Secondly, to design an affordable 60GHz transceiver solution based on available off-the-shelf integrated circuits (IC). The rest of this paper is organized as follows. Section 2 presents the constraints associated with the design of printed circuit boards (PCB) for millimeter wave radio communications. In section 3, a USRP based 60 GHz radio link is detailed and validated with the help of an OFDM transmission chain built with GNU Radio. Finally, section 4 concludes the paper by giving evolution possibilities of the current demonstrator.

2 Electronics design constraints for millimeter wave bands

In recent years, the design and the realization of transceiver solutions up to about 6 GHz have been simplified by the availability of relatively low cost dedicated chips and measurement equipment. Moreover, the manufacturing of PCB even for more than two signal layers is relatively easy and low cost. Nowadays, it is no longer necessary to be a member of a specialized firm or lab to take advantage of RF PCB manufacturing. Several Chinese firms provide this service for a relatively low cost. For example, a lot of 10 pieces of 50 mm x 50 mm FR4 4 layer boards would cost you about 50$ [1].

Up to a frequency of 10 GHz the PCB design process is relatively easy. But what about millimeter wave bands applications? As you may know, pushed by the 5G mobile telephony potential market, things are changing rapidly so that dedicated ICs and suitable measurement equipment will be easily available in the next few years. As far as measurement equipment is concerned, the cost is definitely going to be a problem for amateurs. Again, this problem can be overcome by the Software Defined Radio (SDR) technology. The only problem that remains is the availability of millimeter wave front-ends for off-the-shelf SDR hardware. It is possible to build one using frequency mixers and gain blocks available from manufacturers like Mini-Circuits but it is going to be bulky, costly and currently the maximum frequency achievable is 40 GHz [2]. After some research, I found an interesting IEEE paper about an open source SDR frontend for the 60 Ghz band by Zetterberg et al [3]. Unfortunately, the Hittite ICs they used have been phased out. In the meantime, Hittite was acquired by Analog Devices (ADI). Frequently in the past, I used to have several good ideas with no facilities to implement them, but was finally lucky to find their realization by ADI. Again, I dreamt about a 60 GHz transceiver solution and ADI did it!

3 A HMC6300/6301 60 GHz link with USRP radios

The HMC6300/6301 Systems on Chips (SoC) from ADI are an evolution of the HMC6000/6001 from Hittite. Their main characteristics are summarized in Table 1. Figure 1 presents a block diagram of the HMC6300 SoC. ADI provides an evaluation kit for the HMC6300/6301 for about 3500$. This kit allows the user to set up a half-duplex link but does not include the antennas. In order to obtain a link

budget of more than 100m we decided to fit the boards with 23dBi horn antennas from SAGE [4]. The total price of this setup is around 5700$.

HMC6300/6301
Frequency range: 57 GHz to 63 GHz
Frequency steps: 250MHz, 500MHz or 540MHz
RF signal bandwidth: 1.8GHz
Universal analog IQ baseband interface
3 wire SPI interface for configuration
HMC6300 P1dB: 15dBm
HMC6301 NF: 8dB

Table 1: HMC6300/6301 specifications.

The last operation to perform is to connect the kit to the USRP radios. This is easily accomplished thanks to LFTX/LFRX or BasicTX/BasicRX daughter boards from Ettus. These boards can transmit/receive baseband data to/from the evaluation board as long as the user provides a differential/non differential adapter (balun transformer or Fully Differential Amplifier) to the IQ interface of the HMC6300/6301. The testbench has been successfully tested over a 100 m link with a 100 MSymb/s OFDM chain designed with GNU Radio. The transmit power was set to 10dBm. Figure 2 shows a picture of the transmitter section.

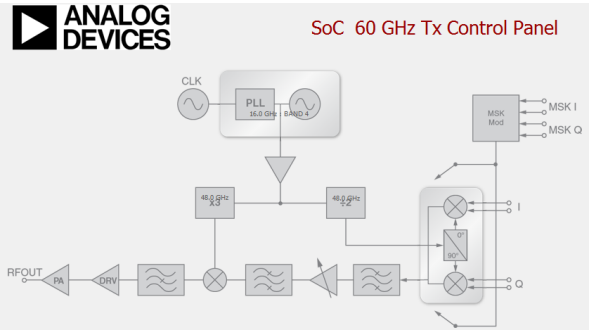

Figure 1: Block diagram of the HMC6300 SoC.

4 Conclusion

The daughterboard interface of N210 and X310 radios is quite generic and allows to easily connect user designed boards like the 60 GHz evaluation kit presented in this paper. We are currently working on a cheaper and a better integration of ADI 60 GHz SoCs with USRP radios. This requires the design of a PCB integrating an HMC6300/6301 eval board and a 71.42857 MHz ECL oscillator circuit needed by the HMC6300/6301 synthesizers. The setup of the SoC transmission parameters (e.g. output power, carrier frequency) is made via an SPI interface which can be implemented with the available GPIO pins on the LFTX/RX or BasicTX/RX daughter boards. The working testbench and its evolutions will be presented at the European GNU Radio Days in Besançon and the design files will be shared with the community.

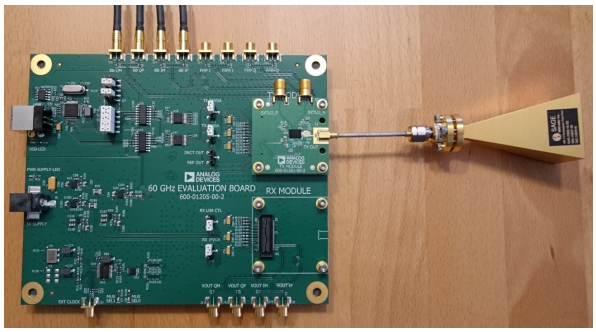

Figure 2: HMC6300 TX front-end fitted with SAGE horn antenna.

References

[1] `https://www.pcbway.com/`

[2] `https://ww2.minicircuits.com/homepage/homepage.html`

[3] P. Zetterberg, R. Fardi, *Open Source SDR Frontend and Measurements for 60-GHz Wireless Experimentation*, IEEE Access, Vol. 3, pp. 445-456, 2015.

[4] `https://www.sagemillimeter.com/23-dbi-gain-wr-15-v-band-rectangular-horn-antenna/`

A Software Defined Radio 802.11 Infrared Transmission System

S. Joumessi-Demeffo,H. Boeglen,S. Sahuguede,P. Combeau,D. Sauveron,L. Aveneau, A. Julien-Vergonjanne
CNRS, XLIM UMR 7252, F-87000 Limoges & F-86000 Poitiers, France

Abstract

We present a 802.11 infrared packet transmission system built around GNU Radio and homemade infrared front-ends. The communication system is validated by a demonstration of audio transmission.

1 Introduction

Optical wireless transmission is of great interest in many critical environments due to the confinement of optical beams (higher level of security) and the lack of interference of existing radio frequency connections. This is particularly true for aeronautical contexts. As part of the Aircraft Light Communication (ALC) project (European project CleanSky2 H2020), our goal is to design a multi-user optical communication system in an aircraft cockpit. Since the brightness inside the cockpit must be greatly reduced during the critical phases of flight, the visible band is not appropriate. Therefore, the optical wavelengths in the infrared (IR) domain are used.One of the most successful IR standard (IrDA) is not applicable due to small operating distance and the need for directed Line-Of-Sight (LOS) links with strong alignment. Because of growing interest in visible light communications, there is today a lot of activities on optical wireless communication standards. This includes different groups such as IEEE802.11bb, IEEE802.15.7r1, IEEE802.15.13, and the ITU G.vlc. We decided to follow the IEEE 802.11 standard and adapt it to our study using IR technology. To enable interoperability with the 802.11 radio-frequency standards that dominate the market and thus facilitate light communication penetration, the future standard should be as close as possible to existing 802.11 specifications.Particularly in the ALC project, the interoperability of the system with the 802.11 standard is important because it guarantees a system evolution and the ease of adaptation of this system to the future 802.11bb standard. The standardization effort consists of the data link control layer (subdivided into two sublayer namely: the Logical Link Control (LLC) and the medium access control (MAC)) and physical (PHY) layer.We choose a multi-step work: firstly we implement the On-Off-Keying (OOK) modulation which is one of the most used modulations in optical wireless transmission systems, and second we will study other modulations used in 802.11 versions. In order to cope with the design challenges involved, we found it practical and efficient to develop our solution using the SDR technology and in particular with USRP radios and GNU Radio. Indeed, the GNU Radio platform already provides a large library of digital communication blocks and allows implementing rapidly design modifications in software. One can find several research works dealing with the implementation of the 802.11 standard for SDR applications. These implementations are focused on RF communications. As far as optical communications are concerned one can find several Visible Light Communications (VLC) implementations focused on the 802.15.7 standard. The rest of this paper is organized as follows. Section 2 discusses the main characteristics of the developed communication chain. Section 3 presents an infrared packet transmission system using homemade IR front-ends before concluding in Section 4.

2 Communication chain

The developed chain uses the classical MAC/PHY layer model. We focus in particular on the transmission and the reception of packets for the LLC sub-layer and the frames of the IR PHY physical layer.The LLC sub-layer is, among others, in charge of building 802.11 compliant packets. One can find an existing GNU Radio transmission chain which includes this compliant packet forming scheme [1]. The 802.11 compatibility can be verified by using the network analyzer tool Wireshark. The LLC blocks are: WIFI MAC in charge of the packetizing operation at the transmission side and WIFI PARSE MAC in charge of the packet analyze at the reception side. The packetizing operation has the structure presented in Fig.1.

The WIFI PARSE MAC block is able to distinguish and list the different fields of the received packet. In the Bloessel implementation [3], the block in charge of this operation is part of the OFDM physical layer. This is the reason why we could not use it in our IR PHY implementation and

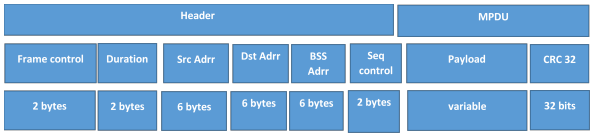

Figure 1: IR PHY packet structure.

had to develop a MAC packet data unit (MPDU) Parse block. This block is in charge of recovering the payload.The LLC sub-layer packets are then aggregated in a frame compliant with the 802.11 IR PHY standard specification [2], shown in Fig.2. This is a starting point because this specification is no longer present in the latest versions of the standard. One of our goals is to adapt it to the constraints of the ALC project.

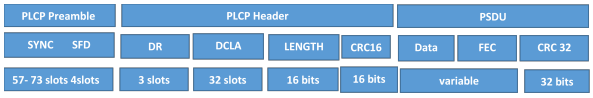

Figure 2: LLC packet structure.

We have worked with the available GNU Radio tutorials and documentation. We have developed a PHY Formatter block dealing with the insertion of the synchronization fields and the transmission header forming a frame. The Parse PHY block at the reception side analyzes the frame and returns the length field to the Demux Header/Payload block which then demultiplexes the header and the payload. For simplification reasons, as a first step, we fix the time slot value to the bit time instead of 8s as specified in the 802.11 IR standard. Us PHY layer implements an OOK modulation by adding an offset to the BPSK modulation scheme. This offset is required by the IR front-ends. OOK modulation is one of the most used in wireless optical communication systems.The main GNU Radio tools for packet communication systems design are the tags and the messages.These tools use polymorphic type (PMT) data. Several blocks of our chain use PMT. For example the convolutional code ECC block specified by the 802.11 standard, the packetizing blocks WIFI MAC, WIFI Parse, PHY Formater, PHY Parse and Header/Payload Demux.

3 Front-ends & demonstration

The 802.11 PHY IR specification recommends data rates of 1 Mbps and 2 Mbps. In a multi-user context, it is essential to have a high data rate to ensure good communications of all users. For that purpose, we designed RX and TX front-ends [3] having a 10MHz bandwidth. Due to the constraints of IR technology, the link has to work in baseband (OOK,L-PPM modulations). To comply with this requirement, Ettus LFTX and LFRX daughterboards have been selected as they allow communications between 0 and 30MHz. The designed TX IR front-end converts the LFTX board voltage ranging between 0 and 3.3V to a 0-100mA current into the IR LED thanks to a high speed video op amp. The role of the RX front-end is to convert the very low PIN photodiode current (in the order of 1uA) into a voltage value of around 1V. This is accomplished by a special type of op amp called a transimpedance amplifier (TIA).The demonstration is made of two parts: first, we send a file from the IR transmitter and analyze the frame received at the receiver side using Wireshark. Secondly, a user directly speaks in a microphone connected to the PC of the IR transmitter and it is possible to listen the sound recorded by the computer connected to the IR receiver.

4 Conclusion

As part of the H2020 ALC project we are developing a multi-user infrared system for communications in a cockpit. To ensure scalability, the system should be as close as possible to the 802.11 standard currently under development for optical wireless communications. Our approach is based on the implementation of an 802.11 packet transmission system developed with GNU Radio, tested with USRP radios including specifically manufactured optical front-ends. Currently, our system is able to transmit 802.11 packet data with an OOK-modulated PHY IR. Our future work will focus on the integration of other modulation schemes such as OFDM used in 802.11 a/p/n and the implementation of the DCF access method.

References

[1] *B. Bloessl et al., "Towards an Open Source IEEE 802.11p stack: A full SDR-based transceiver in GNU Radio," 2013 IEEE Vehicular Networking Conference, Boston, MA, 2013, pp. 143-149.*

[2] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications p1491 2012.*

[3] *H Boeglen, Steve Joumessi-Demeffo, et al.. Optical front-ends for USRP radios. French GNU Radio Days 2018, Jul 2018, Lyon, France.*

Study of the use of a SDR Passive RaDAR for the safe outdoor operation of an atmospheric LiDAR

F. Peyrin[1], J.-M. Friedt[2]
[1] OPGC, UMS 833 CNRS, Univ. Clermont Auvergne, Aubière, France
[2] FEMTO-ST, UMR 6174 CNRS, Univ. Franche-Comté, Besançon, France

March 28, 2019

Abstract

Our project aims to evaluate the potential of the Software Defined Radio technology associated to GNU Radio ecosystem in order to propose new and pragmatic solutions when applied to scientists needs, thus bridging the gap between engineering and research points of view. Here, we study the ability for a passive RaDAR to detect aircrafts and allow safe operation of a LiDAR.

1 Introduction

The OPGC [1] is an Observatory of Earth Sciences dedicated to Volcanology and Physical Meteorology. Among instruments implemented, a LiDAR helps to characterize the composition of the atmosphere in aerosol particles.

The LiDAR activity requires compliance with international air traffic regulation [2]. Due to strong Laser emission, the potential ocular hazard for the pilots must be cancelled by stopping the Laser emission while an aircraft is flying over the critical zone of the LiDAR. Usually, air traffic safety is provided by a X-band pulsed radar determining the presence (position, altitude, speed) of any aircraft entering the vicinity of the LiDAR.

On the one hand, a first alternative solution has been developed based on acquiring and processing ADS-B frames transmitted by the IFR aircrafts [3]. On the other hand, we also study and present here the potential of a passive RaDAR solution based on J.-M. Friedt previous work [4] using an existing non-cooperative source.

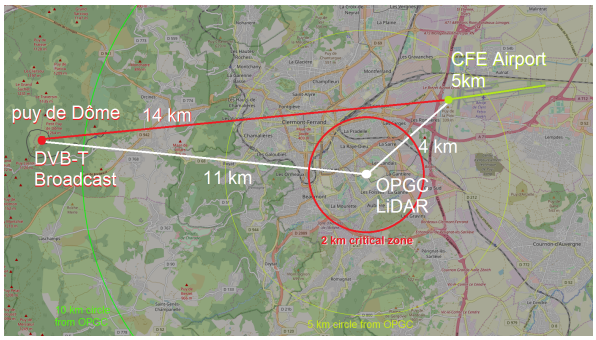

Figure 1: LiDAR location and critical zone.

We used GNU Radio to acquire and save DVB-T signal (Fig. 2) echoed by the moving distant target.

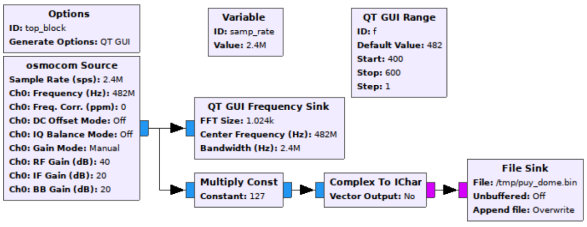

Figure 2: GNURadio code for DVB-T acquisition

2 Experimental setup

Experimental setup involves the reception of echoes of the local Terrestrial Digital Video Broadcasting (DVB-T) source reflected by cabin's aircrafts. The LiDAR and receiver are located (Fig.1) on the roof of the OPGC, 420 m above sea level, 4 km away from the airport and 11 km from the Puy de Dôme, 1465 m asl., where is located the DVB-T emitter.

The receiver hardware configuration was based on a RTL-SDR USB stick that features the Realtek RTL2832U chipset and the R820T2 tuner specifically designed for use in SDR mode. We used this broadband receiver coupled to a UHF yagi antenna to acquire DVB-T broadcast at 482 MHz.

Then, using MATLAB (MatWorks), we post-processed these data by autocorrelation to find the transmitted signal, delayed in time, so shifted in frequency by Doppler shift.

3 Results

Several tests have been processed (available at [5]). Among these, a measurement at a bistatic distance of 10 km (Fig.3), which is sufficient compared to the critical zone, suggests the ability to detect aircraft flying up to ten thousand meters above our LiDAR.

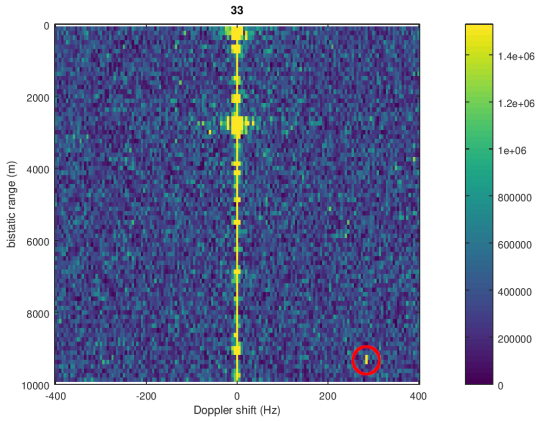

Figure 3: Remote detection (a.u.) of the target, circled with red, at a bistatic distance of 10 km, i.e. an aircraft at a distance of 5 km in a configuration where source, receiver and target are aligned.

The DVB-T emitter provides promising results that need to be confirmed with a suitable hardware, i.e. two synchronized receivers fed by directional antennas, one pointing to the transmitter (reference signal) and an other one pointing to zenith to detect eventual mobile targets (surveillance channel). Separating the reference and monitoring channels improves the signal-to-noise ratio and, more importantly, eliminates artifacts related to autocorrelations of signals due to multiple targets.

The main limitation currently observed is the processing time of the acquired signals. In the absence of any optimization (arbitrary choice of 221 processed points, for a Doppler shift analyzed by steps of 5 Hz between -400 and +400 Hz), and post-processing the acquired data with an interpreted language (GNU / Octave), two minutes are needed to process every second of recording. Without considering a processing shorter than the acquisition time, the main effort, in addition to the experimental setup, must focus on the reduction of this computation time, highly parallelizable and optimizable in terms of choice of the parameters of analysis. Towards the aim of real time processing, shifting part of the processing chain from general purpose Central Processing Units (CPUs) to FPGA parallel computing units is being developed [6].

4 Conclusion, Perspectives

This study highlights the potential of the SDR concept in regard to our applications. While detecting ADS-B with a dongle could already give an inexpensive solution for air traffic safety during LiDAR activity, the passive RaDAR implementation could be complementary for operational purpose if hardware is upgraded and calculation time is lowered.

More generally, SDR implementation may also be helpful for further Earth Science Experiments driven at the OPGC, in particular when applied to remote sensing instruments. For example, Software Defined Radio could also be dedicated to passive or active RaDAR concepts involved in characterization of atmospheric clouds or volcanic ash plumes. The main technical objectives are to improve the ergonomics and the mobility of instruments on the field, to improve the spatial and temporal resolutions (size and movement of the targets), to reach a greater modularity (emission-reception, directions, frequencies ...), and to optimize the overall cost of development and maintenance of these new systems. This ongoing study will be focused on free and open source ecosystem such as GNURadio.

References

[1] at `http://wwwobs.univ-bpclermont.fr/`

[2] *Doc 9815 AN / 447 2003 of the International Civil Aviation Organization, 2003* at `https://www.skybrary.aero/bookshelf/books/3849.pdf`

[3] *Tests of the ADSB-SDR technique to detect aircrafts - An alternative to radar solution for air traffic safety during Lidar activity, F. Peyrin et P. Freville, 2018* at `https://www.researchgate.net/publication/329874059_Tests_of_the_ADSB-SDR_technique_to_detect_aircrafts_-_An_alternative_to_radar_solution_for_air_traffic_safety_during_Lidar_activity/`

[4] *RADAR passif par intercorrélation de signaux acquis par deux récepteurs de télévision numérique terrestre, J.-M Friedt, 2018* at `http://jmfriedt.free.fr/passive_radar.pdf`

[5] *RADAR passif pour la sécurité d'opération d'un LiDAR, F. Peyrin et J.-M Friedt, Experiment Findings, 2018* at `https://www.researchgate.net/publication/331929425_RADAR_passif_pour_la_securite_d%27operation_d%27un_LiDAR/`

[6] J.-M Friedt, W. Feng, S. Chrétien, G. Goavec-Merou, M. Sato, *Passive radar for measuring passive sensors: direct signal interference suppression on FPGA using orthogonal matching pursuit*, accepted SPIE Multimodal Sensing and Artificial Intelligence: Technologies and Applications (München, 2019)

Embedded GNU Radio running on Zynq/PlutoSDR

G. Goavec-Merou[1], P.-Y. Bourgeois[1], J.-M Friedt[1]
[1] FEMTO-ST Time & Frequency, Besançon, France

Abstract

GNU Radio has been ported to the `buildroot` environment and hence can be run on any platform supported by this development framework, including the Zynq. We extend the frozen version of `buildroot` used by Analog Devices (2018.02) to the `BR2_EXTERNAL` mechanism allowing to use the latest release of `buildroot` and hence the latest added packages, including GNU Radio and associated packages to run on the Zynq as found on the PlutoSDR. We demonstrate running the demodulation scheme on the PlutoSDR itself, and streaming the resulting audio file, as well as providing custom FPGA bitstream for embedded RF frontend processing.

1 Introduction

Current embedded platforms and associated electronics frontends exhibit on the one hand increasing flexibility and on the other hand increasing embedded computational power, with a bandwidth bottleneck at the data transfer from one processing unit to another. A demonstration of this evolution is Analog Device's (ADI) PlutoSDR combining on a same board an AD9363 radiofrequency frontend streaming digital data to the Zynq 7010 System on Chip providing both FPGA (PL) and general computational (PS) functionalities on the same chip. The resulting complex I/Q data are then streamed to a personal computer through a USB connection for further processing. The architecture provided by ADI, in which the Zynq is only used to collect the data and stream them to the personal computer on the one hand restricts the available bandwidth due to the USB bus, and prevents using fully the PS capability of the Zynq. In order to run GNU Radio on the PS and take advantage of the processing power as well as the huge communication bandwidth between PL and PS, we have ported the development framework provided by ADI to the `BR2_EXTERNAL` framework providing a homogeneous, fully consistent development framework. Hence, the latest release of the `buildroot` framework shall be used on the embedded software, including the latest packages such as the necessary extensions to GNU Radio needed to collect data from the AD9363 and process such data on the embedded board. Since the processing requiring most bandwidth is run on the PS, the resulting decimated stream becomes consistent with USB bandwidth when streamed to the personal computer.

In addition to providing the Buildroot framework to add custom software, including GNU Radio blocks, to the PlutoSDR, we provide the ability to tune the bitstream configuring the Zynq PL with custom processing blocks, such as a sound output, making the PlutoSDR a fully autonomous, embedded radiofrequency transceiver.

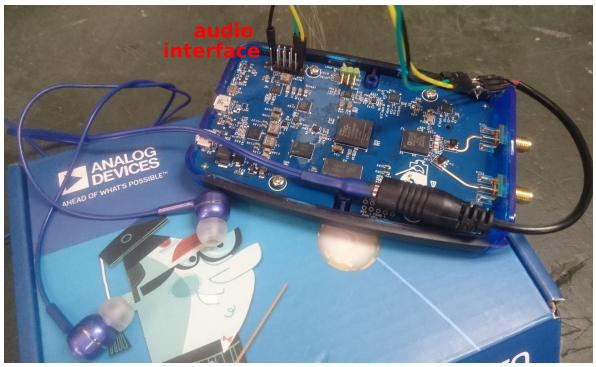

Figure 1: Adding an audio interface to the PlutoSDR through EMIO driven by a PWM added to the original bitstream. In this example, the PWM datapath is independent of the data stream coming from the AD9363 radiofrequency frontend providing the raw I/Q signals needed to demodulate a broadcast FM station and playing sound on the headphones.

2 Experimental setup

ADI's development framework provided at `github.com/analogdevicesinc/plutosdr-fw` uses a `Makefile` configuration to run multiple tools in order to generate all the files needed to generate the embedded firmware. The kernel is compiled out of the buildroot environment, which is itself a version frozen at the time of the release. While the generation of the image is functional, its long term evolution is dependent on porting the updates to the current kernel to the latest Linux release. Furthermore, the version of buildroot is frozen to a version not yet supporting GNU Radio.

Various efforts aimed at leveraging the processing power of the Zynq [1] include running a web server on the embedded target (`github.com/unixpunk/PlutoWeb`) or updating the PL bitstream

`github.com/timcardenuto/testPlutoSDR`. All these projects still rely on the official ADI framework whose long term stability is questionable since buildroot and the linux kernel will keep on evolving. In order to avoid freezing features of a given `buildroot` release, we have extracted the modifications brought by ADI to buildroot and included them in an external branch designed to be merged with the latest buildroot release as provided through the `BR2_EXTERNAL` mechanism.

Furthermore, thanks to the availability of the PlutoSDR HDL firmware, a bitstream can be generated to configure the PL with custom functionalitites. We here promote the compatibility with the OscimpDigital PL/PS co-design framework as documented at `https://github.com/oscimp/oscimpDigital/tree/master/doc/tutorials/plutosdr/`.

3 Results

In order to demonstrate the embedded signal processing using GNU Radio, we tune the AD9363 (whose configuration was updated [2] to match an AD9364 to allow reaching the 100 MHz commercial FM broadcast band) to a broadcast FM station, stream the data to an embedded command-line python script generated from GNU Radio Companion, and transfer the resulting audio stream to the personal computer through the ZeroMQ framework. The personal computer then sends the stream to the sound card to assess the demodulation quality.

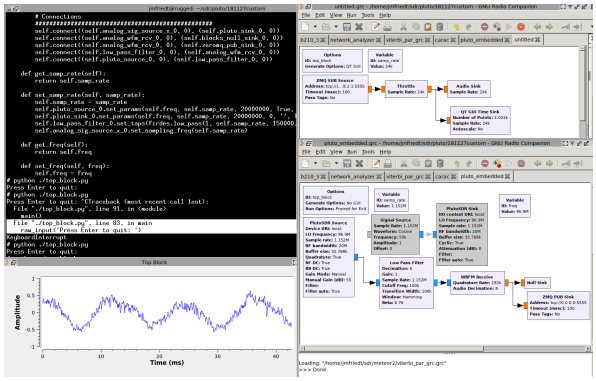

Figure 2: Flowgraphs running on the Zynq target (bottom right) and on the host PC (top-right), streaming the FM signal demodulated on the PS of the target (top-left) and using the host as a sound card.

Fig. 2 exhibits the flowcharts running on the embedded target and the host computer, as well as the resulting oscilloscope output.

Beyond allowing for processing datastreams at the PS side of the Zynq, accessing the bitstream generated to configure the PL allows for including basic preliminary processing steps at the FPGA level. Fig. 3 exhibits the initial FPGA configuration provided by ADI to fetch data from the AD9363 and stream them to the PS memory through the AXI DMA interface. While the basic design only includes FIR decimator and interpolator blocks between the AD936{1,3} block and the AXI DMA, any additional processing block complying with the interfaces might be included to preprocess the data at the FPGA level, hence removing the bandwidth limitation introduced by the PL to PS communication.

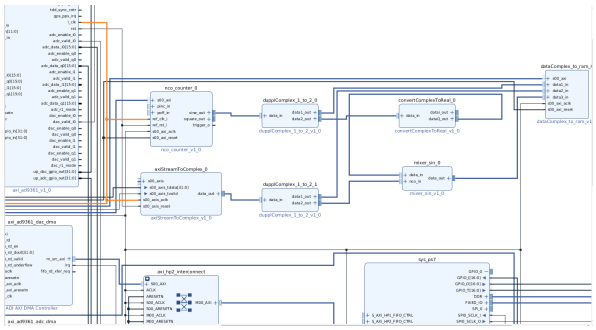

Figure 3: FPGA processing chain, from AD9363 to PS memory through the Direct Memory Access (DMA) AXI stream, and collecting the AXI stream to feed the custom processing chain here made of an NCO and a mixer to demonstrate an additional frequency transposition.

4 Conclusion

A fully functionnal extension to buildroot supporting the PlutoSDR to run embedded processing software is proposed. All development files are released at `github.com/oscimp/PlutoSDR`. The demonstration of the operational framework is achieved by streaming the sound demodulated from the incoming commercial broadcast FM radiofrequency signal onboard the Zynq processor.

References

[1] *PlutoSDR: enable 2nd CPU core for better performance* at `www.reddit.com/r/RTLSDR/comments/7h2hh2/plutosdr_enable_2nd_cpu_core_for_better/`

[2] *Updating to the AD9364* at `wiki.analog.com/university/tools/pluto/users/customizing`

GNU Radio implementation for Multiuser Multi-Armed Bandit learning algorithms in IoT networks

Julio Manco-Vasquez[1], and Christophe Moy[2], Faouzi Bader[1]

[1] IETR / CentraleSupélec Campus de Rennes, F-35510 Cesson-Sévigné, France,
{JulioCesar.MancoVasquez, Faouzi.Bader} @CentraleSupelec.fr
[2] Univ Rennes, CNRS, IETR - UMR 6164, F-35000, Rennes, France
Christophe.Moy@Univ-Rennes1.fr

Abstract

Novel access schemes based on multi-armed bandit (MAB) learning approaches has been proposed to support the increasing number of devices in IoT networks. In the present work, a GNU radio framework is implemented to recreate an IoT network where IoT devices embedding MAB algorithms are able to learn the availability of the channel for their packet transmissions to the gateway. It allows to incorporate several IoT users recognized by an identifier (ID), and provides a gateway to handle a large number of IDs as well as the packet collisions among IoT devices. The experimental results show that the introduction of learning mechanism in access schemes can improve the performance of the network.

1 Introduction

Several efforts to introduce reinforcement learning algorithms tailored for low-power wide-area (LPWA) networks have been recently carried out [1, and references therein]. However, unlike opportunistic spectrum access (OSA) schemes, where several proof of concept based on MAB algorithms have been developed [2, and references therein], the experimental evaluation for IoT networks has been overlooked.

Previous works regarding the evaluation of MAB algorithms for OSA in decentralized networks do not take into account realistic transmissions between the primary and secondary users, and utilize particular toolboxes avoiding to run reproducible experiments. In this regard, a first proof of concept to assess the potential usage of MAB algorithms for IoT scenarios is provided in [1]. It is fully implemented in GNU radio, and we consider this initial effort to introduce features concerning an LPWA network. In doing so, the emulation of an IoT network to support a large number of users is addressed. Our testbed is composed of several IoT devices and a Gateway, where each IoT device following an ALOHA wireless protocol transmits a packet containing its ID, and waits for an ACK packet transmitted by the gateway, as it is shown in Fig. 1. For that end, a data packet structure is implemented to provide the required support for a multiuser scenario. Our demonstration shows that significant gains can be obtained, when a well-known MAB approach, an Upper-Confidence Bound (UCB) algorithm [1] is embedded in IoT devices.

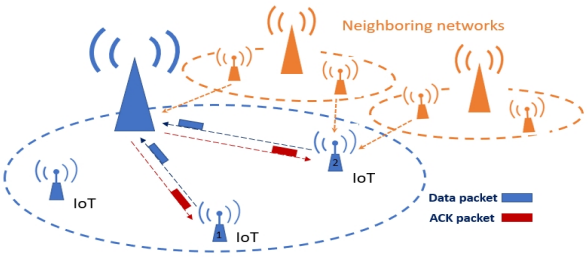

Figure 1: A LPWA scenario with IoT devices that aim to select the best channel for their transmissions to the gateway, while collisions among other IoT devices and interference from other networks may occur.

2 Experimental setup

Our testbed is implemented using N210 USRPs connected to an Octoclock for time and frequency synchronization among the devices. Each USRP running a GNU radio application implements a transceiver composed of three GNU radio blocks. For instance, in an IoT user, a first block corresponding to the physical layer detects and demodulates the packets into QPSK symbols, after which a second block detects the ID within the ACK packet. In the last block, a new packet is created and transmitted through the frequency channel pointed out by the MAB algorithm, if it is embedded in the IoT device [1].

The implemented data packet structure is shown in Fig. 2, where a preamble is utilized for the packet detection and phase correction of the received signal, while the values in the field UP/DOWN allow the receivers at the gateway and the IoT user to only receive uplink and

[1] In a similar way, it operates at the gateway side, where after demodulating the packet, an ID detection is carried out to identify the IoT user, and finally an ACK packet is created in the last block.

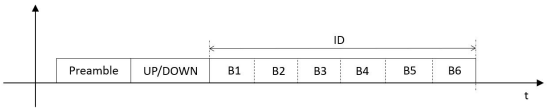

Figure 2: A data packet composed of a preamble, a field named UP/DOWN, and a user ID given by six blocks of QPSK symbols, B_k.

downlink packets, respectively. The ID user is defined by 6 blocks of QPSK symbols B_k for $k \in [1\ 6]$, which are assigned two possible constellation symbols. Then, a

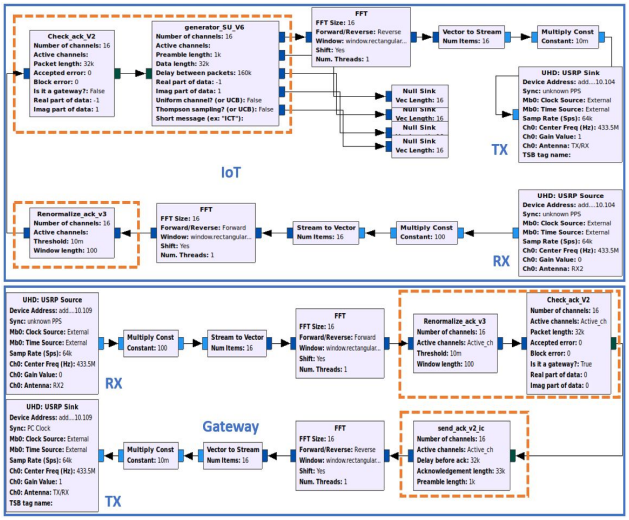

Figure 3: The GRC designs for the IoT device and the gateway are described at the top and bottom of the figure, respectively. The blocks corresponding to our framework are higlighted (in orange boxes) within the flowgraphs.

threshold is applied to the number of times that a symbol is received within each block B_k, so that a decision about the transmitted symbol is made. Hence a binary sequence of 6 bits are obtained, and consequently a total of 64 users can be supported [2].

3 Results

We evaluate our demo in a scenario with a clear line of sight (LOS) by placing two IoT users and a gateway, all of them working at a carrier frequency of 433.5 MHz. Each IoT user embedding a UCB algorithm [3] is able to select among four frequency channels and transmit packets of roughly 0.5 seconds following a LoRa standar. In Fig. 3, the implemented GNU radio companion (GRC) designs are depicted, where

[2]The length of the fields can be adjusted to support more users. In fact, our implementation allows to divide the blocks B_k into chunks of symbols so as to convey more bits. Furthermore at higher sample rates, more symbols per block B_k are available.

[3]For a more detailed illustration of the implemented UCB algorithm, the reader may refer to [1].

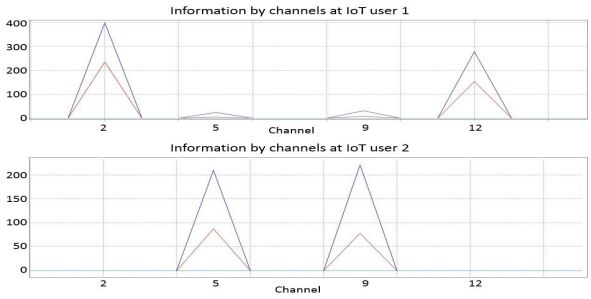

Figure 4: Number of times a channel is selected (curve in blue) and the number of successful transmissions (curve in red) for each channel and IoT users.

a sample rate of 64 kbps, and 4×10^3 symbols for each block B_k are configured. An FFT block of 16 bins is employed to a provide a channel selection mechanism with four channels indexed as 2, 5, 9, and 12. In this example, one of the IoT user is set to use the channels 9 and 12, whereas a second IoT user is able to choose among the four channels. The obtained results in Fig. 4 shows that the second IoT user learns to select the available channels 2 and 12 (curve in blue), meaning that the gateway is able to handle the incoming packets by replying with the corresponding ACK packets. On the other hand, a gap is observed between the number of trials and successful transmission due to the collisions involved in the learning process.

4 Conclusion

We have presented a GNU radio implementation that recreates an IoT network for the evaluation of access policies based on reinforcement learning approaches. Our framework introduces a packet structure to handle a large number of IoT users that may incorporate MAB algorithms. Finally, the experimental results show that a UCB approach improves the performance of the IoT user. Furthermore, the modular design of the proposed framework allows the evaluation of any novel access policy, as well as the incorporation of other physical layers.

References

[1] L. Besson, R. Bonnefoi, and C. Moy, "GNU Radio Implementation of MALIN: "Multi-Armed bandits Learning for Internet-of-things Networks"," in *To appear in 2019 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2019.

[2] R. Kumar, S. Darak, A. Sharma, and R. Tripathi, "Two-stage decision making policy for opportunistic spectrum access and validation on USRP testbed," *Springer: Wireless Networks*, vol. 24, pp. 1509–1523, 2018.

Framework for PHY-MAC layers Prototyping in Dense IoT Networks using FIT/CorteXlab Testbed

Othmane Oubejja, Diane Duchemin, Matthieu Imbert, Leonardo S. Cardoso, Jean-Marie Gorce
Univ Lyon, INSA Lyon
Inria, CITI
F-69621 Villeurbanne, France
othmane.oubejja@inria.fr

***Abstract*—In this paper we present an Internet-of-Things (IoT) network implementation developed as part of the project "Enhanced Physical Layer for Cellular IoT" (EPHYL) , using FIT/CorteXlab radio testbed. The aim of our work is to provide a customizable and open source design for IoT networks prototyping in a massive multi-user, synchronized and reproducible environment thanks to the hardware and software capabilities of the testbed. The massive access feature is managed by emulating several sensors per radio nodes. Two categories of network components are used in our design: a base station unit and a multi-sensor emulator unit. The components are separately hosted in dedicated and remotely accessible radio nodes. Their design features can be illustrated through a live demo, which is also reproducible as it is available for any interested reader.**

***Index Terms*—IoT, FIT/CorteXlab, NB-IoT, Reproducibility, LPWAN, Channel Emulation, Software Defined Radio**

I. Introduction

With the growing needs of rigorous and transparent scientific experimentation across many research fields, the importance of reliable testbeds is only getting more crucial in order to validate scientific models and simulations. Among these fields is wireless communication, and more particularly Internet-of-Things (IoT) which is a rapidly growing and challenging domain. However, one of the challenges is robust joint hardware/software prototyping that is still lacking due to various reasons (e.g. synchronization, immutable implementation solutions, unrealistic testing environment) especially for large scale experiments as required for IoT.

FIT/CorteXlab [1] radio testbed has the key properties to develop an IoT network framework, remotely and freely available for the research community. It allows to fasten the execution of large scale emulated radio experiments, as illustrated here with the evaluation of custom PHY-MAC layers derived from 3GPP Narrowband Internet of Things (NB-IoT) standard [2]. The proposed framework offers to researchers the opportunity to plug their algorithms or designs in Software Defined Radio (SDR) nodes, thus making it possible to evaluate diverse communication scenarios and perform necessary physical measurements.

II. FIT/CorteXlab Radio Testbed

FIT/CorteXlab is a large scale radio testbed composed of an extensive set of SDR nodes. These latter provide substantial RF flexibility, enabling to tune the operating frequency, the channel bandwidth, the emitted power and the waveform used to communicate. The testbed is located in a large shielded room which is partially covered with electromagnetic absorbing foams. Conclusively, the design of the room enables to ensure a high level of reproducibility of experimentations.

Last but not least, all SDR nodes are remotely accessible through a service interface allowing to deploy proprietary software [1].

III. Design Architecture

A. Overview

We started by investigating how we can deploy an IoT-like network considering the testbed capabilities for SDR applications, therefore the objective was to choose then implement key features from NB-IoT systems [2]. Our approach was to design two classes of nodes. The first class represents a Base Station (BS) and the second one a Multi-Sensor emulator (MS). In this work, a Sensor is the actual resource user. To ensure time and frequency synchronization, only USRP nodes are used, since they are connected to 8-Channel Clock Distribution modules. The wireless channels are the downlink Broadcast Channel (BCH) and the uplink, which is based on a custom and simplified version of NB-IoT PHY layer [2] [3]. Fig.1 depicts the overall network structure.

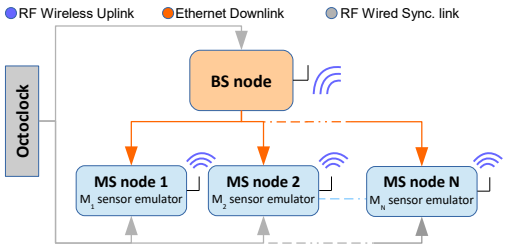

Fig. 1. EPHYL Network diagram.

B. Network Resources Scheduling

The BS follows a defined resource grid policy and a frame structure known by the sensors. At the present time, our design supports single carrier mode only. The BS starts by emitting a beacon signal (BCN) within the Broadcast Channel (BCH) which allows active Sensors to detect the BS and obtain

primary time synchronization. During the Synchronization (Sync) time slot, the Sensors compute the timing of Physical Shared Slots (S_i) and decide which slots to use according to their respective access policies (regular access, Poisson distribution, etc). More details about nodes functioning are given in III-C. A guard time is used at the end of each Shared Slot to store decoded data and avoid overlapping. The final slot within the frame is a post-processing slot and is also used to generate a wired downlink message to MS nodes. This message contains information about potential slots usage and collisions as well as a decoding success score. This feedback can be used to improve the network usage by adapting the access policy and/or PHY layer parameters.

Fig.2 illustrates a functional example of the network flow during one frame, including one BS node and 3 emulated Sensors running with different parameters.

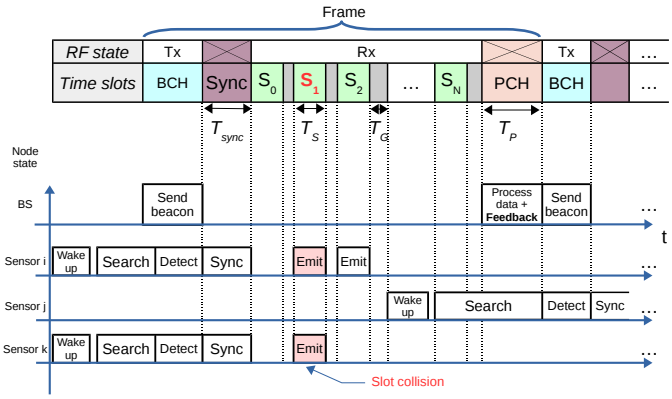

Fig. 2. Example Scenario with slot collision

C. Nodes Description

*1) **BS Node**:* The BS node is disciplined by its *Scheduler* which controls slots timing. On the one hand, the *Scheduler* triggers the *Beacon Generator* during the BCH, on the other hand, it feeds the demodulation chain which starts with a *Header-Payload Demultiplexer* and consists of two other chains. The first one is actually a loop and concerns the header. The *Channel Estimation* block removes channel impairments by correlating the expected pilot sequence with the received header. The second chain purpose is to extract the payload, decode it, identify the emitters, count the decoded messages for each sensor, then broadcast the results in the wired downlink feedback channel.

*2) **MS Node**:* The MS node is composed of emulated *Sensor* blocks which are connected to a complex adder. After the "wake-up" state, each sensor generates its own packets and waits for the *Sync* block to trigger the local *Scheduler* as soon as a beacon is detected. The *Access Policy* block dictates to the *Scheduler* which shared slots to use based one the BS feedback and modifies PHY parameters if needed. Then, the generated waveform passes through a *Channel Emulator* which applies arbitrary attenuation as well as timing and frequency offsets. (See Fig.4)

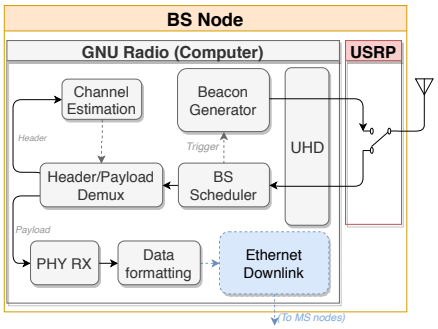

Fig. 3. EPHYL BS Node Block Diagram

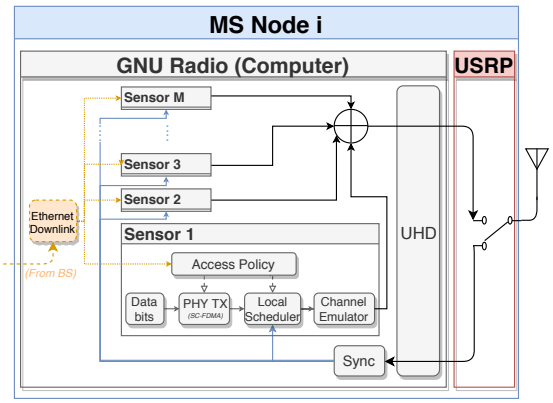

Fig. 4. EPHYL MS Node Block Diagram

IV. Demonstration

Conducting experiments in FIT/CorteXlab requires a basic expertise. All required procedures are explained in the testbed Wiki [4] through several tutorials. Furthermore, these latter and the present framework can be carried out by any person possessing a FIT/CorteXlab account, an Internet connection and a Unix Shell. All framework components of this work are customizable and can be adjusted to users needs. The proposed demo acts as an example to show the flexibility and the modularity of the design.

V. Acknowledgment

This work was supported by the French National Research Agency (ANR), under grant agreement ANR-16-CE25-0002 (EPHYL project). Experiments presented in this paper were carried out using the FIT/CorteXlab testbed.

References

[1] L. Cardoso, A. Massouri, B. Guillon, F. Hutu, G. Villemaud, T. Risset, J.M. Gorce, "CorteXlab: A Cognitive Radio Testbed for Reproducible Experiments,". in Proc. Wireless @ Virginia Tech Symposium, May 2014, Blacksburg, United States. 2014.

[2] Y. -. E. Wang et al., "A Primer on 3GPP Narrowband Internet of Things," in IEEE Communications Magazine, vol. 55, no. 3, pp. 117-123, March 2017.

[3] X. Lin, A. Adhikary and Y. -. Eric Wang, "Random Access Preamble Design and Detection for 3GPP Narrowband IoT Systems," in IEEE Wireless Communications Letters, vol. 5, no. 6, pp. 640-643, Dec. 2016.

[4] "FIT/CorteXlab Wiki". [Online]. Available: https://wiki.cortexlab.fr/. [Accessed: 17- Dec- 2018].

Hacking the DSMx Drone RC protocol

Cyrille Morin[1], Leonardo S. Cardoso[1]
[1] Univ Lyon, Inria, INSA Lyon, CITI, France

Abstract

We present a decoder for a proprietary drone radio-control protocol, DSMx, implemented in GNURadio. This decoder is able to detect a transmission, decode transmitted data, find and follow the corresponding frequency jump sequence and identify the emitter by its manufacturing ID.

1 Introduction

Unmanned Aerial Vehicles (UAVs) or drones have been around fo decades for military and scientific purposes. But they have become increasingly popular with the general public in the last 10 years.

Their widespread usage causes airspace sharing issues. Existing regulations are tailored to big aircrafts, not drones, and their use is either not properly regulated or not covered at all and the public is often not aware of the legal issues with piloting what is sold to be a toy.

This creates privacy and safety issues: It's extremely simple to fly over private properties and film unwilling third parties, or even fly by restricted areas such as airports, military fields or nuclear plants. Manned aircrafts are also at risk of collision, and some drug cartels are starting to use some to deliver drugs to dealers.

Regulations enforcement is not simple due to the drones' size and stealthiness and, even if one is caught, it does not help with locating the pilot that can be more than a kilometre away.

Systems have been developed to physically capture drones with nets, to jam the radio transmission, or to simply destroy them. All these systems require knowledge of the drone's position and to be close to it.

A study and implementation of the radio control protocols in use could allow for detection of unwanted drones without line-of-sight and either takeover or targeted jamming without blocking the radio spectrum.

DSMx is a proprietary protocol from Spektrum used by other drone manufacturers such as Horizon Hobby. A couple of hobbyist groups such as PaparazziUav[1] and Deviation, whose objective is to create an universal transmitter implementing protocols from the various drone manufacturers, have it in their support list. The Deviation GitHub repository has a very good description of it[2]. But their approach uses the proprietary hardware with a source code modification to implement other protocols and add different transmission chips for increased compatibility.

Jonathan Andersson presented a system able to detect and decode a transmission and emit signals to hijack a drone in October 2016 at PacSec in Tokyo [3]. GNURadio was used to study the protocol but the implementation was done on hardware with the original radios. So, to the best of our knowledge, there is no previous DSMx decoder implemented with software defined radio

2 Protocol description

The DSMx protocol works on a CYRF6936 radio chip that implements a frame based data transmission with Frequency Hopping Spread Spectrum (FHSS) and Direct-Sequence Spread Spectrum (DSSS) based on a GFSK modulation scheme at 1Mbit/s in the 2.4GHz ISM band. After a preamble, a start-of-packet (SOP) code is sent in a specific sequence to tell the hardware the DSSS parameters of the communication. In normal operation, the DSSS step is done by sending variations over two 64bits long pseudo-noise (PN) codes: one bit corresponds to the PN code used, one tells if a NOT operation was applied and six more describe the number of bit-wise shifts of the code, so one 64bits chip codes for one byte.

Over this, the frame consists of the last two of the four identification bytes defined in each radio card, followed by 14 bytes coding for the values of seven RC channels. If needed, frames can be sent in pairs to handle up to 12 RC channels.

The PN codes used for data and SOP are selected from a 5 by 9 matrix depending on the current radio channel and the ID bytes.

A CRC is added at the end of each frame for error detection. It uses the first two ID bytes as seed. Unfortunately, the exact algorithm and/or polynomial was not found.

The FHSS element is done by changing the radio channel after each frame in a jump sequence of 23 different channels over 74 possibilities. We won't go into the details of the algorithm, but it is a constrained pseudo-random generator with a seed being the ID bytes.

There is also a pairing frame to configure the receiver to listen to one specific transmitter. It's sent

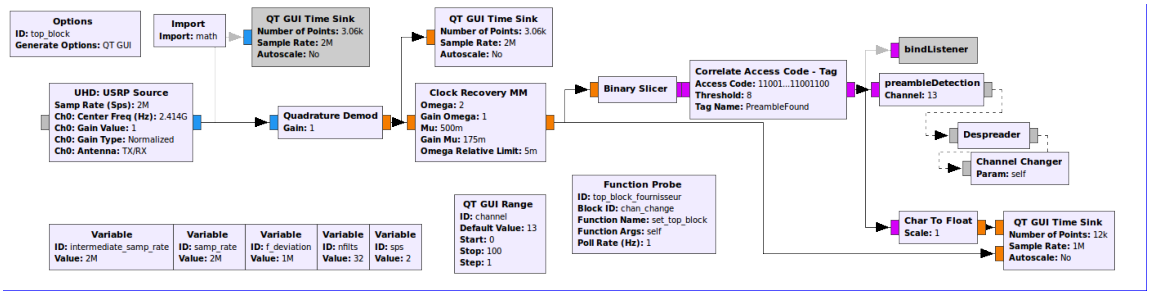

Figure 1: GNURadio flowgraph for DSMx detection, decoding, and FHSS channel sequence following

on a random radio channel and contains data describing the transmittter capabilities (RC channel number and protocol type) and, more importantly, the four ID bytes used in the jump sequence and the PN code selection.

3 Demo setup

This demonstration presents an implementation of a DSMx decoder in GNURadio.

The USRP listens to one channel at a time at 2Msps. In the three following blocks, the quadrature demod with the binary slicer handle the GFSK demodulation and the Clock recovery does the synchronisation. Then, we look for the preamble for time synchronisation and use next three custom blocks for: packet isolation and PN code discovery, data decoding, and frequency jump following and GUI output.

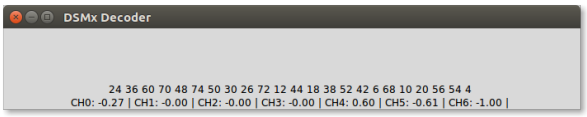

Figure 2: Gui output of the system with current RC channel values and computed jump sequence

The codes used in each frame are based on the value of the ID bytes, that we don't know, but, with the selection algorithm and the radio channel number known, it becomes simple to search for the SOP code in only seven possibilities, and with that the PN codes can be extracted. So there is no need to know anything about the transmitter to be able to decode data.

If the algorithm was known, it would be possible to search over the space of 2 bytes to extract the two ID bytes that are not sent with each frame. Those two bytes would allow for exact transmitter identification and computation of the radio channel jump sequence.

To get these, the pairing frame can be listened to. This allows to demonstrate the capacity of the USRP to follow the jump sequence over a known transmitter.

4 Conclusion

In this demonstration, we present a GNURadio implementation of a system able to decode DSMx data from an unknown transmitter with a partial identification of the transmitting radio chip. If the transmitter is known, we can show that the SDR hardware used is able to follow the protocol's FHSS jump sequence quickly enough and not lose frames. Further progress on the system depends on the discovery of the exact CRC algorithm: It would allow full transmitter identification and jump sequence computation for full data decoding, selective jamming of the concerned channels or emission of custom frames.

The implementation code and flowgraph are available on GitHub at `https://github.com/lscardoso/gr-dsmx-rc`.

References

[1] Paparazzi. (2017) Paparazzi uav wiki. [Online]. Available: http://wiki.paparazziuav.org/wiki/Main_Page

[2] Deviation. (2017) Custom firmware for walkera devo radios. [Online]. Available: https://github.com/DeviationTX/deviation

[3] J. Andersson. (2016) Attacking dsmx spread spectrum frequency hopping drone remote control with sdr. [Online]. Available: https://pacsec.jp/psj16/PSJ2016_Andersson_Hacking_DSMx_with_SDR_PacSec_2016_English.pdf

Author Index