A Spectrum Analyzer Based on a Low-Cost Hardware-Software Integration

Efrain Santos-Luna
Departament of Electrical Engineering
Universidad Autónoma Metropolitana
Iztapalapa, Mexico City
san.lu.ef@gmail.com

Alfonso Prieto-Guerrero
Departament of Electrical Engineering
Universidad Autónoma Metropolitana
Iztapalapa, Mexico City
apg@xanum.uam.mx

Rafael Aguilar-Gonzalez
Departament of Electrical Engineering
Universidad Autónoma Metropolitana
Iztapalapa, Mexico City
r.aguilar@xanum.uam.mx

Victor Ramos
Departament of Electrical Engineering
Universidad Autónoma Metropolitana
Iztapalapa, Mexico City
vicman@xanum.uam.mx

Miguel Lopez-Benitez
Department of Electrical
Engineering and Electronics
University of Liverpool. United Kingdom
M.Lopez-Benitez@liverpool.ac.uk

Marco Cardenas-Juarez
Facultad de Ciencias
Universidad Autónoma de San Luis Potosí
San Luis Potosí, México
mcardenas@fc.uaslp.mx

***Abstract*—Instruments are a very important tool for the good comprehension of any science. In the engineering electrical field, most of the practice laboratories are equipped with a large set of devices. One of the major challenges to get a complete laboratory is the economic cost. Professional electronic devices as multimeters, oscilloscopes or spectrum analyzers are quite expensive on the market. Fortunately, with the aid of software-defined radio (SDR), low-cost equipment can be built. A fundamental device in these projects is the RTL-SDR dongle. This electronic accessory captures wireless signals and puts them on a USB port. Therefore, with digital signal processing it is possible to get access to low-cost instruments like spectrum analyzers. In this work, an integration of hardware and software is presented. An application is developed with Python and implemented on a Raspberry Pi3 B+. This proposal captures, demodulates, and displays AM and FM signals. The signals obtained can be observed on frequency or time or can be recorded as well.**

***Keywords*—RTL-SDR Dongle, Raspberry, Python, FM, AM.**

I. Introduction

Mathematical concepts are fundamental for basic science and engineering. In the case of Electrical Engineering, a lot of physical processes are presented in models requiring from simple to very complex mathematical operations. Through digital signal processing, mathematics are implemented on electronic devices and presented as instruments on laboratories. For example, in the case of electronic communications, radio-electric waves cannot be simply observed by the human eye; however, they are present in almost every place. For the right understanding of these phenomena, spectrum analyzers are very important instruments [1]. With these instruments, concepts, as Fourier transforms, may be applied to radio signals or other applications.

Spectrum analyzers can be found in several presentations and at different prices. They are considered by the private sector or academic institutions to perform multiple studies. However, most of them represent an expensive investment, that not all entities are willing to make. Thanks to the improvements in electronic devices, nowadays it is possible to develop a spectrum analyzer considering several low-cost devices. One can find in the literature several proposals related to this topic. Even if these projects do not account with the same capability as a professional spectrum analyzer, they are able to exhibit similar results. This represents a very attractive and economic solution to acquire equipment for educational labs, small companies or just as a hobby.

Several low-cost spectrum analyzers are based on software-defined radio (SDR). The target of this technology is to reduce physical components and perform most of the changes by software [2]. Among several applications, SDR is an overriding part of cognitive radio (CR) [3]. Recently, it has been considered in educational institutions [4]–[6]. In the beginning, due to its high price, SDR was just available for researchers and industry. On the market, there exist several SDR platforms [7], [8]. For example, the RTL-SDR dongles, which offer an SDR receiver for approximately only $20 dollars [9]. These devices may be used on computers, cellphones, or Raspberry boards.

Most of low-cost spectrum analyzers consider GNU radio software as the programming interface. This software is one of the main platforms among SDR users. GNU radio was developed mainly to be used with the Universal Software Radio Peripheral (USRP) developed by Ettus [10]. However, the use of function blocks on GNU radio sometimes limits the understanding of modulations, demodulations or similar processes. On the other hand, the trend on developing software based on Python has gained terrain. Python is a programming language that is popular since it is easy to learn [11]. In this regard, thanks to Python and the support for RTL-SDR libraries, it is currently possible to develop applications according to each specific use.

Thanks to Programa para el Desarrollo Profesional Docente para el Tipo Superior-Secretaría de Educación Pública (PRODEP-SEP, UAM-PTC-665).

In this work, we present a low-cost spectrum analyzer considering a Python code to process the radio-electrical spectrum, which is mounted on a Raspberry Pi3 B+ board. With our design, it is possible to get mainly three important aspects of wireless communications such as a general signal analyzer, as well as FM and AM signal analyzers. The corresponding application is able to display spectrograms, discrete-time signals (DTS), power spectrum density (PSD) signals, radio broadcast data service (RDS), and record the received signals. In the next, in Section II we present a summary of related work. The contributions of this work are highlighted in Section III. In Section IV, the hardware considered is described. Section V explains how signals are processed by the application developed. Finally, conclusions are presented in Section VI.

II. Related work

An early low-cost spectrum analyzer designed to monitor TV white spaces is presented in [12]. Such a design considers a Raspberry Pi to read the information coming from the sensor RF explore and a battery for portability [13]. The frequency reception range goes from 240 MHz to 960 MHz; the system is calibrated based on a professional spectrum analyzer. This integration is a good beginning; however, such an option has to be seen more as a sensor than as a spectrum analyzer. In [14], another option for a spectrum analyzer is presented along with cellphones or tablets. In this design, some devices as a frequency translator, a router with a wireless card among other accessories are included. On one hand, the nice features of this design are its frequency range and sensitivity. On the other hand, its cost, spectrum sampling, and portability of the integration among other characteristics are features that must be improved. A proposal similar to the one presented here appears in [15]. In this paper, authors use a Raspberry Pi2 and an RTL-SDR dongle to develop a spectrum analyzer. In this work, authors also describe the details related to the setup of the devices. The software programming for this project is GNU radio, some FM bands can be tuned with the integration. However, the portability of the project is limited because there is no battery or screen. These projects and the one presented in [16] compare the results obtained with RTL-SDR dongles with commercial spectrum analyzers. The outcomes of both choices are very similar, so the RTL-SDR dongle is a good option to get an operational spectrum analyzer. Recent work appears in [17], where an RTL-SDR dongle, a Raspberry Pi3 B with GNU radio are used for spectrum data acquisition. Such a design is compared with a commercial spectrum analyzer and it is found that the former outperforms the spectrum analyzer. However, the proposal's design is not portable. In all of these developments, only the analysis of the spectrum is considered. Some of these analysis are captured and listened by the FM signal but none of them displays discrete-time signals for AM, FM, or PSD. The work presented here adds the feature of observing all these signals, save them and use them in other applications.

III. Contributions of our design

In the previous section, we described several contributions on this research direction. Even if they are interesting contributions, most of them lack of one key feature, which is having a low-cost. Besides of being nicely operational, our work adds tasks to the low-cost spectrum analyzer to do more attractive this kind of developments. Thus, our integration includes functionalities such as general analysis of DTS, DTS for AM and FM signals. Also, our design allows the display of RDS signals in the frequency and time domains. This design includes options such as recording quadrature signals in a *.wav* file, FM stereo signal demodulation, spectrograms, stop signals, a zoom tool and changing the gain values. The software development we present here is mounted on a platform that can be used for other application such as the Internet of Things (IoT), robot control or as a simple computer. Fig. 1 shows the full integration we present in this work. Here, we can see all the hardware elements and the application running a general spectrum analysis. The project's cost is about $300.00 USD. In the next, the hardware and software characteristics are described.

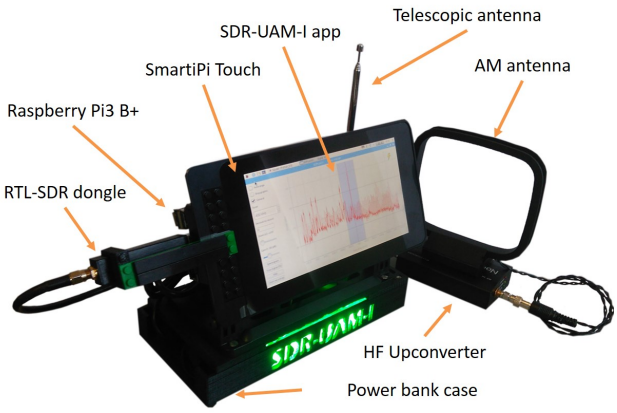


Fig. 1. SDR-UAM-I: A low-cost spectrum analyzer.

IV. Hardware description

An important part regarding this integration is the hardware. The electronic components considered in this project are easy to get and have a low-cost. Also, in order to enhance the view of the project and to protect some devices, a couple of parts were designed and manufactured on a 3D printer.

A. Raspberry Pi3 B+

This board is one of the main platforms for the study of several topics such as computer science, education, and IoT. The version of this board includes a system of a chip (SoC) solution for a Broadcom BCM2837B0 with a Cortex-A53 (ARMv8) 64-bit at 1.4GHz, 1 GB SDRAM, WiFi, Bluetooth, 4 USB ports among other characteristics. Basically, Raspberry is a computer only by $35.00 dollars. At the time of submission of this work the newest Raspberry Pi 4 appeared, but it was still not available on the market [18]. To account with a

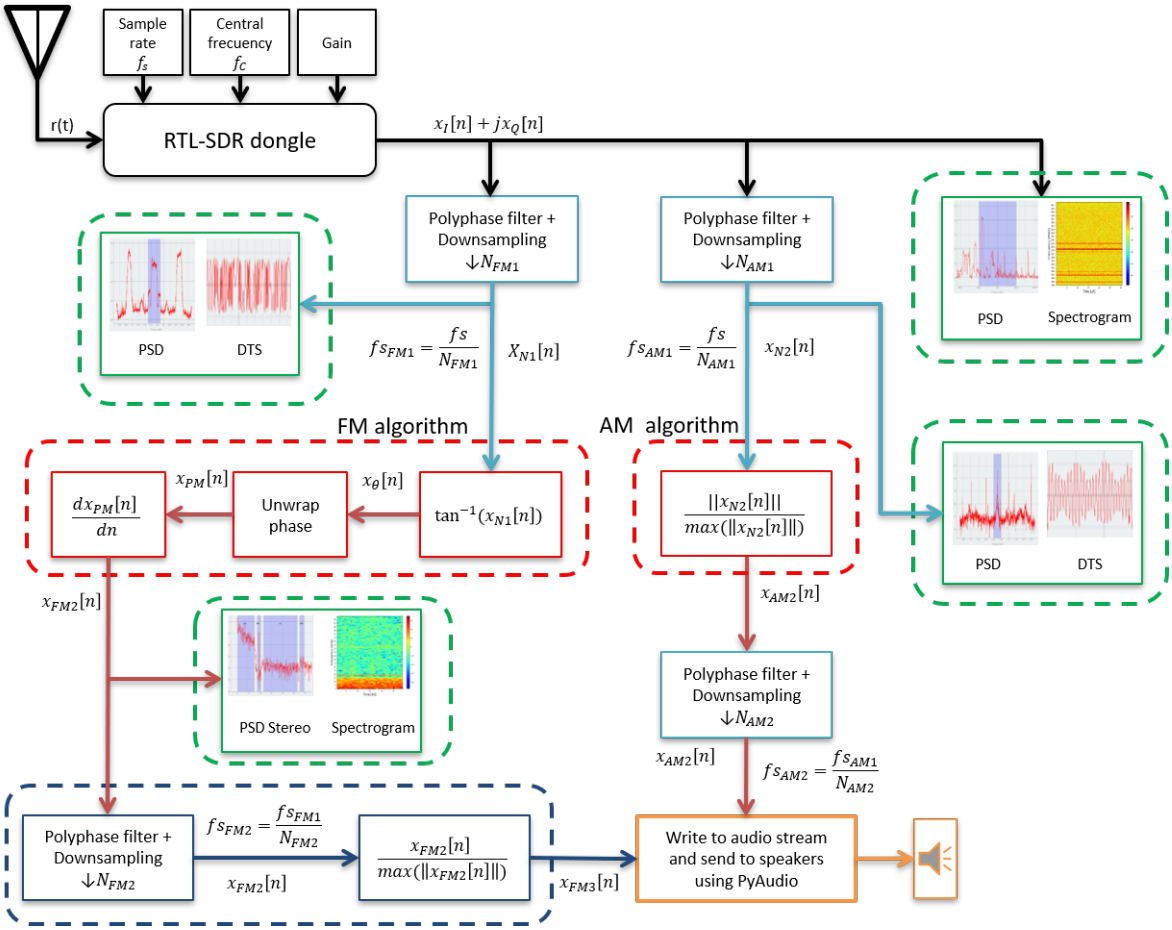


Fig. 2. Diagram blocks of SDR-UAM-I software development.

complete equipment, a SmartPi touch case is included along with the official Raspberry Pi 7" touchscreen display. Also, the Raspbian Operating System is installed on a 32 GB micro SD SanDisk extreme memory.

B. *Nooelec NESDR SMArt RTL-SDR dongle*

Among a long list of dongles, this device is probably the most complete product just for $29.95 dollars. The manufacturer points out several product features such as 25 MHz – 1750 MHz frequency range, a telescopic antenna, an ISM antenna, a UHF antenna, a temperature compensated crystal oscillator (TCXO) at 0.5 PPM for frequency correction and a metallic enclosure. This dongle provides an I/Q signal with an 8-bit resolution and a bandwidth of 2.4 MHz. To capture the AM signal, the HF Up-converter and an AM antenna are necessary. This accessory is also offered by Nooelec at $64.95 dollars. The frequency range of this device goes from 300 Hz to 65 MHz. Then, the operation frequency range of this integration is from 300 Hz to 1750 MHz [19]. It is important to mention that on the market there are more RTL-SDR dongles, which also can be used for this integration.

C. *Power bank*

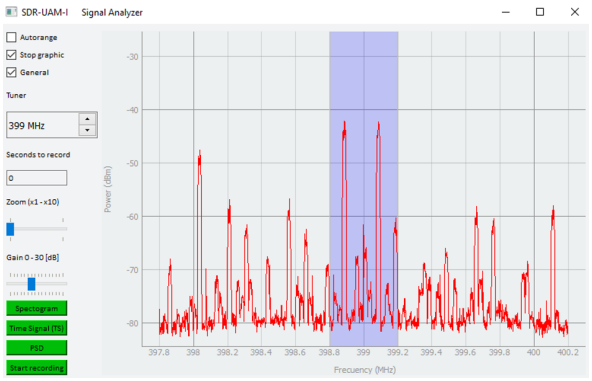
A very important characteristic of this development is the mobility. A spectrum analyzer just for the laboratory is not something desired here. However, if the reader finds high the price of the power bank, it could be omitted. The ADATA P20000D is considered to provide enough energy to the Raspberry. The capacity of this device is 20000 mAh or 72 Wh. The total integration may work on several places in order to make spectrum measurements, just powered with the battery for around 9 hours [20].

D. *Power bank case and accessories*

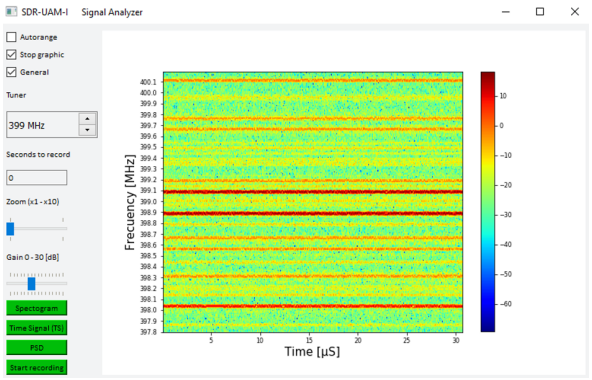
To protect the power bank, a case is designed on a 3D printer. Power cables can be saved in this accessory so as to provide stability to the integration. To improve the design, some characters are printed and lightened with LEDs on a printed circuit board. Additionally, a small part is designed fixed to a LEGO piece in order to avoid movements of the RTL-SDR dongle when it is connected to the Raspberry. Also, at the back of the Raspberry case, a metallic plate is included to place the antenna mast.

V. Software Description

The application is developed with Pyhton 3.7 and it is called SDR-UAM-I. Several updated Pyhton libraries are used in our design; for example, pyrtlsdr, pyqt, numpy, soundevice among others. Firstly, an analog signal is captured by an antenna, where the sample rate (f_s), the central frequency (f_c), and the gain are configured. Next, the RTL-SDR dongle performs an I/Q sampling and puts the information on a USB port [21], [22]. Thus, the discrete signal coming from the dongle is $x_I[n] + jx_Q[n]$, which can be seen in Fig. 2. From this point, according to the user settings, the signal is processed for three different options: (1) general, (2) FM, or (3) AM signal analyzer. Next, we describe such options.



(a) PSD of the frequency band of 399 MHz



(b) Spectrogram of the frequency band of 399 MHz

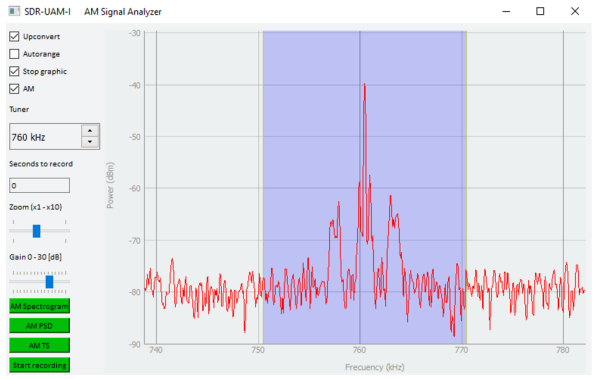
Fig. 3. Two windows of the general signal analyzer.

A. General signal analyzer

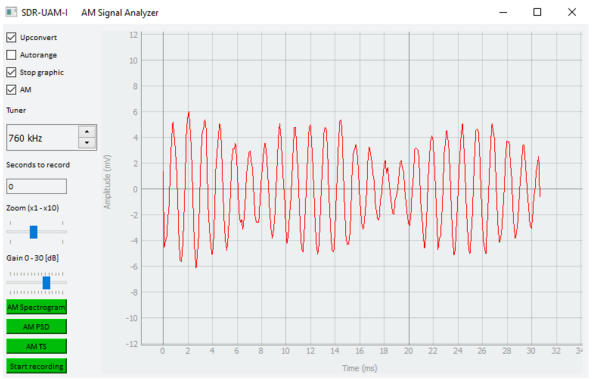
In this option, I/Q signals coming from the RTL-SDR dongle are processed directly. After checking the box *General*, four buttons appear, which are *Spectrogram*, *Time Signal (TS)*, *Power Spectrum Density* (*PSD*) and *Start recording*, as can be seen in Fig. 3. By pushing the *PSD* button, a Discrete Fourier Transform (DFT) of 2048 points is computed and the result is converted to `dBm`. The signal displayed follows the RTL-SDR dongle limits, the frequency range goes from 25 MHz to 1.7 GHz with a bandwidth of 2.4 MHz. The value f_c can be moved from the *Tuner* box. The plotted signal has 1.2 MHz up and down from the f_c value. For the case of the recorded signal, the number of seconds should be captured on the *Second to record* box and then click the button *Start recording*. When the application finishes the process, a *.wav* file with the I/Q information is stored automatically in the same folder where the app is located. For the frequency band of 399 MHz, Fig. 3(a) shows the PSD and Fig. 3(a) presents the spectrogram of the same signal analyzed.

B. AM signal analyzer

Before selecting this option, it is necessary to connect the HF Upconverter and AM antenna to the RTL-SDR dongle that appear in Fig. 1. In the AM signal analyzer option, four buttons are displayed: *AM spectrogram*, *AM PSD*, *AM TS* and *Start recording*. Fig. 2 shows how in order to get AM signals, a polyphase filter and a downsampling process are necessary. The latter is a decimation operation, where the decimation factor is N_{AM1} with $N_{AM1} = 12$; then we use this signal to compute a DFT. Fig. 4(a) shows the result of this operation applied to the 760 kHz frequency band. In this figure, the PSD of a real signal from an AM broadcasting station is plotted. Here, sometimes in order to observe in a better form the AM broadcasting, the gain value should be increased.



(a) AM PSD of the frequency band of 760 kHz



(b) AM TS of the frequency band of 760 kHz

Fig. 4. Two windows of the AM signal analyzer.

Afterwards, to see correctly the *AM TS*, the decimation of $N_{AM2} = 12$ is applied. Up to this point, it is important to remember that the typical bandwidth of an AM signal is about 10 kHz. Previous to the AM algorithm shown in Fig. 2, (1)

is computed. The parameters of the dongle are set to $f_s = 2.4 \times 10^6$ sps. Thus the new sample rate is $f_{s_{AM1}} = 200kHz$ (see (1)). The result corresponds to the AM time signal that can be plotted with the *AM TS* button as can be seen in Fig. 4(b)

$$f_{s_{AM1}} = \frac{f_s}{N_{AM1}} = \frac{2.4 \times 10^6}{12} = 200 \times 10^3 = 200\ kHz. \quad (1)$$

Next, the AM algorithm is executed with the $x_{N_2}[n]$ signal and then the $x_{AM}[n]$ is obtained, as expressed by (2).

$$x_{AM}[n] = \frac{\parallel x_{N_2}[n] \parallel}{\max \parallel x_{N_2}[n] \parallel}. \quad (2)$$

To save and listen correctly the baseband signal, another polyphase filter and a downsampling process are applied to $x_{AM}[n]$. The new decimation operation N_{AM2} is applied. This value must be set according to the frequency parameter of the audio card where this application is executed. This part is considered as future work in our development. By now, it is not possible to find a right down sampling value that corresponds to the frequency sound card of the Raspberry Pi3.

C. FM signal analyzer

In order to complete the basic signal analysis, in this section, the FM algorithm is presented. In the FM signal analyzer option, several signals such as *RDS TS*, *FM stereo TS*, *RDS PSD*, *FM stereo PSD*, *FM spectrogram*, and *FM PSD* can be obtained. However, in this description, just the general FM algorithm is explained. Fig. 2 shows how a process similar to the AM analysis is done for the case of FM. Firstly, the decimation operation $N_{FM1} = 12$ is executed and the DFT operation is applied. Again, the result is converted to `dBm` and plotted as can be seen in Fig. 5(a), where the PSD of the frequency band of 97.7 MHz is presented.

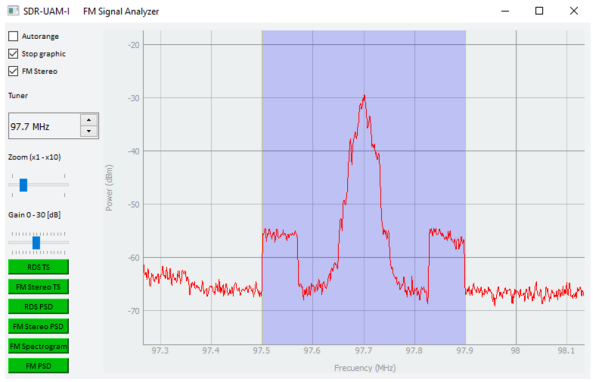
To limit the frequency band to 200 kHz, a decimation factor of $N_1 = 12$ is applied to an FM signal of typical bandwidth. The result of the new sampling frequency $f_{s_{FM1}} = 200\ kHz$ is computed with (3). Fig. 5(b) shows the time signal obtained when pushing the button *FM TS*.

$$f_{s_{FM1}} = \frac{f_s}{N_{FM1}} = \frac{2.4 \times 10^6}{12} = 200 \times 10^3 = 200\ kHz. \quad (3)$$

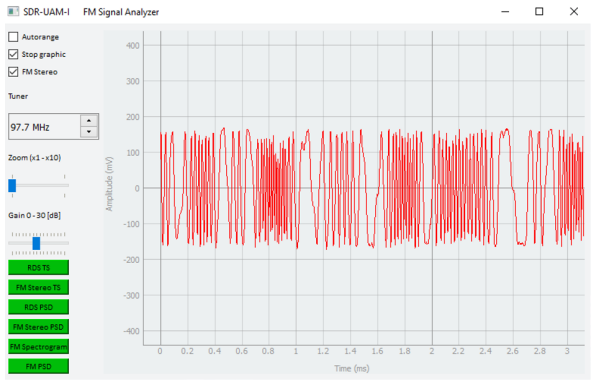
Additional operations are necessary to capture FM stereo signals. The FM algorithm considers the $x_{N_1}[n]$ vector to obtain the phase changes with $x_\phi[n]$, as expressed by (4).

$$x_\phi[n] = \tan^{-1}(x_{N_1}[n]). \quad (4)$$

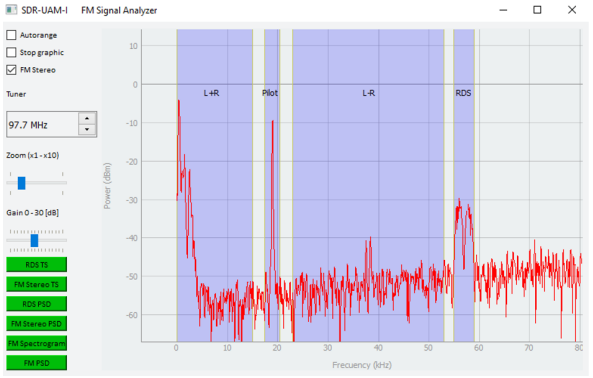
In this part, it is necessary to unwrap the signal phase, where the result is the $x_{PM}[n]$ vector. The last signal is derived to recover efficiently the FM stereo data, as expressed by (5).



(a) FM PSD of the 97.7 MHz frequency band.



(b) FM stereo TS of the 97.7 MHz frequency band.



(c) FM stereo PSD of the 97.7 MHz frequency band,

Fig. 5. Three windows of the FM signal analyzer.

$$x_{FM2}[n] = \frac{dx_{PM}[n]}{dn}. \quad (5)$$

The function $x_{FM2}[n]$ represents the FM stereo signal. These data are separated in $L + R$, Pilot, $L - R$ and RDS as shown in Fig. 5(c).

In order to capture the FM stereo signal, a new decimation is applied with $N_{FM2} = 12$. Now, a new sampling frequency is used only to obtain the $L + R$ information. The approximated bandwidth of the $L + R$ signal is 16 kHz, which is the reason to compute (6).

$$f_{s_{FM2}} = \frac{f_{s_{FM1}}}{N_{FM2}} = \frac{100 \times 10^3}{12} 16.6 \times 10^3 = 16.6\ kHz. \quad (6)$$

To listen the FM stereo signal, (7) is computed. Finally, the $x_{FM3}[n]$ vector is obtained; then, this signal is sent to the speakers. As with the AM option, this step is part of the future work.

$$x_{FM3}[n] = \frac{x_{FM2}[n]}{\max \parallel x_{FM2}[n] \parallel}. \quad (7)$$

VI. Conclusions

In this work, the design of a low-cost spectrum analyzer was presented. Following recent trends, this project is based on SDR fundamentals. An important device considered here is the RTL-SDR dongle. Thanks to this component, it is possible to receive frequencies from 25 MHz to 1.7 GHz. Among several solutions, our proposal comes with an application designed with Pyhton, which is able to display PSD, TS, spectrograms, RDS or saving spectrum information. Also, this work considers a common widely-used platform like Raspberry Pi as a data processor. We consider that these improvements may increase the interest of the private sector or educative institutions on acquiring low-cost devices. As future work, the application will be implemented in different platforms like Windows or Linux operating systems. Also, the SDR-UAM-I application will be available on-line for the users interested. In order to approximate the project to a complete CR, a low-cost transmitter is being designed

Acknowledgment

This work was supported by the Programa para el Desarrollo Profesional Docente para el Tipo Superior-Secretaría de Educación Pública (PRODEP-SEP, UAM-PTC-665).

References

[1] C. Rauscher, *Fundamentals of Spectrum Analysis*. Rohde & Schwarz, 2007.

[2] IEEE, "IEEE standard definitions and concepts for dynamic spectrum access: Terminology relating to emerging wireless networks, system functionality, and spectrum management," Tech. Rep., Oct 2008.

[3] J. Mitola and G. Q. Maguire, "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, Aug 1999.

[4] S. G. Bilén, A. M. Wyglinski, C. R. Anderson, T. Cooklev, C. Dietrich, B. Farhang-Boroujeny, J. V. Urbina, S. H. Edwards, and J. H. Reed, "Software-defined radio: a new paradigm for integrated curriculum delivery," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 184–193, May 2014.

[5] A. M. Wyglinski, D. P. Orofino, M. N. Ettus, and T. W. Rondeau, "Revolutionizing software defined radio: case studies in hardware, software, and education," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 68–75, January 2016.

[6] V. P. G. Jimenez, A. L. Serrano, B. G. Guzman, and A. G. Armada, "Learning mobile communications standards through flexible software defined radio base stations," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 116–123, May 2017.

[7] M. T. Masonta, M. Mzyece, and F. Mekuria, "A comparative study of cognitive radio platforms," in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, Oct 2012, pp. 145–149.

[8] R. Akeela and B. Dezfouli, "Software-defined radios: Architecture, state-of-the-art, and challenges," *CoRR*, vol. abs/1804.06564, 2018. [Online]. Available: http://arxiv.org/abs/1804.06564

[9] B. RTL-SDR. (2019) About RTL-SDR. [Online]. Available: https://www.rtl-sdr.com/about-rtl-sdr/

[10] E. Blossom. (2004) GNU radio: Tools for exploring the radio frequency spectrum. [Online]. Available: https://www.linuxjournal.com/article/7319

[11] (2019). [Online]. Available: https://www.python.org/

[12] A. Arcia-Moret, E. Pietrosemoli, and M. Zennaro, "Whisppi: White space monitoring with Raspberry Pi," in *Global Information Infrastructure Symposium - GIIS 2013*, Oct 2013, pp. 1–6.

[13] (2019). [Online]. Available: http://j3.rf-explorer.com/

[14] T. Zhang, A. Patro, N. Leng, and S. Banerjee, "A wireless spectrum analyzer in your pocket," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '15. New York, NY, USA: ACM, 2015, pp. 69–74. [Online]. Available: http://doi.acm.org/10.1145/2699343.2699353

[15] E. G. Sierra and G. A. R. Arroyave, "Low cost SDR spectrum analyzer and analog radio receiver using GNU radio, Raspberry Pi2 and SDR-RTL dongle," in *2015 7th IEEE Latin-American Conference on Communications (LATINCOM)*, Nov 2015, pp. 1–6.

[16] A. Saeed, K. A. Harras, E. Zegura, and M. Ammar, "Local and low-cost white space detection," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 503–516.

[17] D. Ball, N. Naik, and P. Jenkins, "Lightweight and cost-effective spectrum analyser based on software defined radio and Raspberry Pi," in *2017 European Modelling Symposium (EMS)*, Nov 2017, pp. 260–266.

[18] (2019). [Online]. Available: https://www.raspberrypi.org/

[19] (2019). [Online]. Available: https://www.nooelec.com/

[20] (2019). [Online]. Available: https://www.adata.com/us/feature/425

[21] (2019). [Online]. Available: http://www.ni.com/tutorial/4805/en/

[22] M. A. Wickert and M. R. Lovejoy, "Hands-on software defined radio experiments with the low-cost rtl-sdr dongle," in *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, Aug 2015, pp. 65–70.