Spectrum analyzer with USRP, GNU Radio and MATLAB

António José Costa, João Lima, Lúcia Antunes, Nuno Borges de Carvalho
{antoniocosta, jflima, a30423, nbcarvalho}@ua.pt

January 23, 2009

Abstract

In this paper, a spectrum analyzer in the 2.3 - 2.7GHz band is proposed using the USRP, GNU Radio and the MATLAB software. With this solution, a cheap and reliable analyzer can be made, also serving as an introductory work for a cognitive radio. A RF signal generator was used to perform the tests, and the results achieved are in accordance with the ones obtained with a standard spectrum analyzer. We observed that the system response was considerably affected by the LNA gain, thus further regulation on its gain will be crucial for correct measurement.

1 Introduction

The concept of Software-Defined Radio (SDR) has been acquiring a great deal of attention in the past several years. Created in 1991 by Joseph Mitola [1], it is "a radio whose channel modulation waveforms are defined in software". This will allow users to operate the radio in different environments and applications. Since the software is easily reconfigurable, the radio itself would become adaptable to a wide range of situations, depending on the user's needs.

This concept brought up a new idea, the Cognitive Radio (CR). Also devised by Mitola[2, 3], it is essentially a SDR that senses its environment, tracks changes, and reacts according to its findings. CR is widely regarded as being the next step in the evolution of radio system, since it can adapt itself to every situation, replacing the traditional single function radios.

2 Software-Defined Radio

Software-Defined Radio is a system where the main part of the radio is implemented in software. Typical components such as amplifiers, filters, mixers, modulators / demodulators and detectors (among others) that were usually implemented in hardware, can now be implemented in software. This system is characterized by its flexibility and reconfigurability, due to the fact that changing its behavior only requires modifying or replacing the software.

Since SDR is a radio where its components are implemented in software, there is a need for a standard set of tools to process the signal received on the antenna. The GNU Radio is a toolbox to implement such a SDR. Essentially it consists of a signal-processing package which users can use to construct their own radio. Implementing a radio is done by connecting blocks according to the desired structure.

The SDR architecture, depicted in Figure 1, consists of a RF front end, an Intermediate Frequency (IF) part, and software to perform all the baseband computations. The front end is comprised by the antenna, LNA and filter, and will downconvert the radio frequency signal to a frequency the ADC can process. To receive the RF signals, a RF Front End is all that is required; this was previously accomplished using a PC sound board. Currently it is still not possible to implement an ideal SDR. It is envisaged that future deployments won't require an IF, but that is dependent on the ADC conversion rate. The present SDR implementations occur as shown in Figure 1.

The Universal Software Radio Peripheral (USRP) is the hardware which receives the RF signal and downconverts it to the baseband for digital processing. It consists of a RF front end and

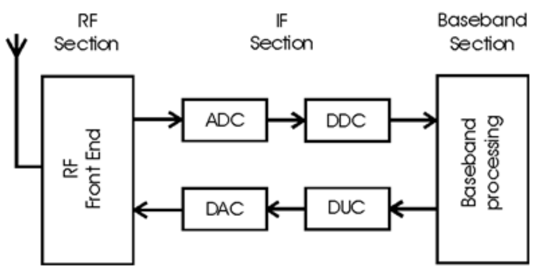


Figure 1: SDR architecture

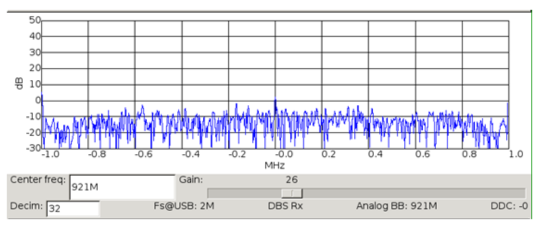
IF blocks. Since its design is open and schematics and drivers are freely available, at an affordable cost, it is a worthwhile choice for those who work with Software-Defined Radio.

With the access to the SDR being far easier with the solutions presented, all the requirements are met to move forward towards Cognitive Radio. As seen in the introduction above, one of the important parts of the CR is spectrum sensing. This task is usually carried out with professional equipment, however the costs associated with it usually restrict its employment. Our proposal is a spectrum analyzer using GNU Radio tools, MATLAB, and the USRP hardware.

3 Practical implementation

3.1 GNU Radio

The GNU Radio package is a set of tools for the deployment of software-defined radio systems. One of the most relevant tools for spectrum sensing is the GNU Radio spectrum analyzer (usrp_fft.py), which is shown in Figure 2.



Center freq: 921M
Gain: 26
Decim: 32
Fs@USB: 2M
DBS Rx
Analog BB: 921M
DDC: -0

Figure 2: GNU Radio spectrum analyzer

This analyzer can effectively detect frequency peaks, perform arithmetic averages, and view multiple frequency scales. It can also locate and focus on the central frequencies of the spectrum, select decimation value, and adjust the Low Noise Amplifier (LNA) gain.

However, this standard analyzer also has some limitations. It does not allow the detection of unused frequencies, making it impossible to perform the sensing part of the CR. Moreover, it is only possible to observe a narrow frequency band, and thus one cannot visualize the whole gap. As mentioned above, the CR should be able to sense the environment to detect the free spaces, and with this short range of operation that is not possible.

3.2 Frequency sampling

In order to overcome such a limitation, we evaluated another GNU Radio function, the usrp_rx_cfile.py. This function allows us to collect samples taken from the USRP and store them in a data file (.dat). We can also tune our input parameters in order to change the sampled central frequency, the LNA gain, the number of samples, decimation value, among others. An implementation example is described below: *./usrp_rx_cfile.py -d 32 -f 2.4G -g 5 -N 20 test.dat*

Our idea to sense the proposed spectrum - from 2.3 to 2.7 GHz - is to pick up separate parts of the spectrum, and then aggregate the different data in one big picture. To ease the work, a Bash script was created in order to automatically take the samples with the proper input parameters.

With all the samples taken, the next task is the signal processing. Since MATLAB is a powerful tool which includes FFT toolboxes, we have opted to work on all signal processing parts with it.

For a user-friendly interaction, the graphical interface depicted in Figure 3 was built in MATLAB. This interface allows us to collect all the samples, and display them in an intuitive graphic with a simple click. Afterwards, it is possible to choose the central frequency and range, in order to have the domain according to our needs.

Moreover, and because the main purpose of this spectrum analyzer is the CR, we are required to separate the used frequencies from the non-used. To overcome this adversity, we created an auxiliary vector containing the FFT position of the channels. This vector, with the length of the FFT, is initialized to “1”. After searching for used channels, the positions on this vector representing occupied fre-

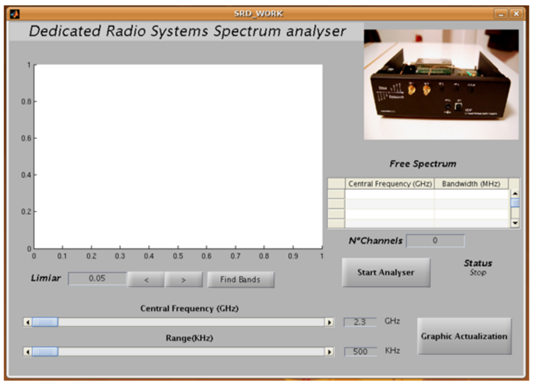


Figure 3: Spectrum analyzer graphic interface

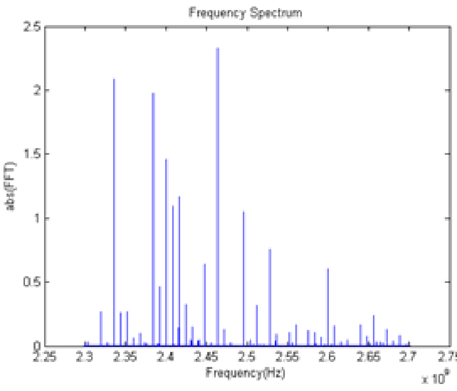


Figure 4: Frequencies spectrum with a direct cable with an LNA gain of 70dB

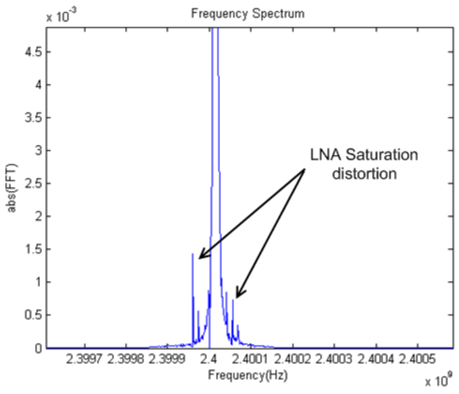


Figure 5: Frequency spectrum around 2.4GHz

quencies are set to "0". It should be noted that not only the central frequency but the whole 40 MHz centered in it is set to "0". A typical protocol operating in the given range is WiFi (centered around 2.4 GHz). Since it occupies a 22MHz bandwidth, we have chosen 40MHz as a reasonable value to avoid interference with other channels. The resulting vector works as a mask, allowing us to efficiently know which frequencies are occupied, requiring only a single sweep across the whole vector, checking which values are set to "1".

4 Experimental Results

To assess the validity of the proposed solution, several experimental tests were performed. The first was to directly connect the frequency generator to the USRP with a direct cable. A sine wave with a -30dBm gain centered at 2.4GHz was then created, and the sensing operation was carried out with an LNA gain of 70dB. The results are displayed in Figure 4. As shown in the figure, the spectrum does not correspond to what we expected. The spectrum shows several frequencies, whereas our frequency generator was only transmitting around 2.4GHz.

Zooming in a bit more, in Figure 5 we can see that around the 2.4GHz frequency there are spurious frequencies caused by LNA saturation distortion. To attest these results, and to verify that the obtained results were not as we expected, we used a professional frequency sweep device to obtain a base for comparison. With it, we were able to confirm that the distortion was caused by the USRP.

After some experimental tests, we discovered that the cause for the distortion issue is the LNA gain, which cannot be above a given value or it will distort the signal.

After decreasing the LNA gain to a suitable value - 10dB - and taking samples again, we could see that the spurious frequencies have disappeared as we expected. The frequency window is as in Figure 6. Zooming in further, we could see in the same Figure that the frequency bandwidth around 2.4 GHz is as expected, and the spurious frequencies cease to appear.

In the second simulation, an antenna was connected to the frequency generator in order to sweep the spectrum. One should notice that the received power is considerably inferior to the one received with direct connection to the USRP. Due to this

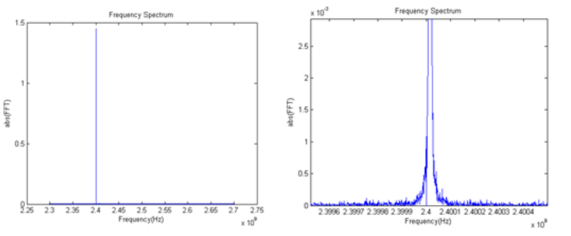


Figure 6: Frequency windows after decreasing the LNA gain to 10dB

fact, the LNA gain was experimentally adjusted to a higher value. This simulation consisted of several tests with the generated frequencies ranging from 2.3 to 2.7GHz and varying the output power in order to accurately simulate real-life conditions. The experimental values for the LNA gain, in order to obtain acceptable values, ranged from 50 to 70dB. To avoid the referred distortion, we used a value of 50dB for the following tests.

After calibrating the LNA gain, we moved to a real-life situation. Without an external frequency generator, we sensed the spectrum in search of used frequencies. The result is depicted in Figure 7.

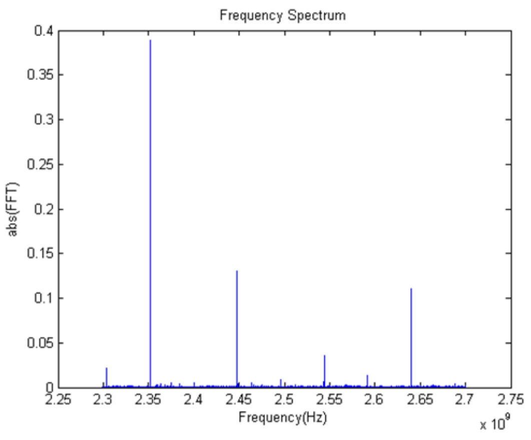


Figure 7: Frequency spectrum in a real-life experiment

We can see that the frequencies shown in the figure are centered around 2.4GHz, and correspond to well known protocols.

In Figure 8 we can also observe the differences in the spectrum around 2.4GHz when the external frequency generator is off. The image on the left of Figure 8 shows the spectrum without the external generator, while the one on the right shows the central frequency band broadened.

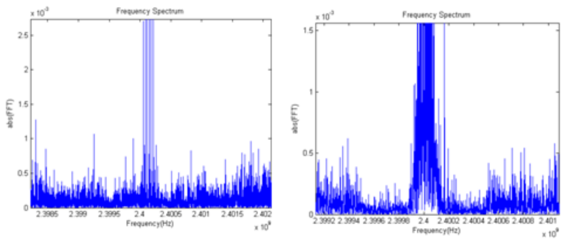


Figure 8: Effects of the frequency generator

5 Conclusion

In this paper we proposed an alternative to the expensive frequency spectrum analyzers using an USRP, GNU Radio and MATLAB software. In the first part of the work, a more theoretical study was made in order to assess the capabilities of the GNU Radio toolbox and the characteristics of the USRP. Afterwards, and due to the insufficiency of the GNU Radio spectrum analyzer, a new application was proposed. It was developed in order to overcome the narrow bandwidth spectrum given by the original tool.

In the experimental and testing phase, it was seen that the results obtained were in accordance with the ones achieved with a professional spectrum analyzer.

It is the authors' opinion that this tool consists in a powerful and cheap solution to analyze the spectrum, and with further development it can be a step forward in order to achieve the devised Cognitive Radio.

References

[1] Joseph Mitola III, Software Radio Architecture, John Wiley & Sons, 2000

[2] J. Mitola III and G. Q. Maguire, "Cognitive radio: making software radios more personal," IEEE Pers. Commun., vol. 6, no. 4, pp. 13–18, 1999.

[3] J. Mitola III, "Cognitive radio: an integrated agent architecture for software defined radio," Ph.D. dissertation, Computer Communication

System Laboratory, Department of Teleinformatics, Royal Institute of Technology (KTH), Stockholm, Sweden, May 2000.