Indian J.Sci.Res. 17(2): 247-250, 2018 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

RECEIVING FM/AM SIGNALS USING SOFTWARE DEFINED RADIO AND USRP

[1]M. Asha Jyothi, [2]Priyanka Saxena , [3]A.Ravindar
[1,2,3] Electronics and Communication Engineering, Keshav Memorial Institute of Technology, Narayanaguda, Hyderabad

Abstract - The field of electronics has seen amazing developments with new technologies emerging everyday that provide solutions to problems of all magnitudes, from socio- economic to the advanced scientific computing. One such development seen was the Open BTS project using the GNU Radio and USRP. This is an abstract of these technologies. A software-defined radio (SDR) system is a radio communication system which can tune to any frequency band and receive different modulations across a large frequency spectrum by means of a programmable hardware which is controlled by software. This paper demonstrates the flexibility and one of the many utilities of the SDR, combined with the USRP and GNU Radio. However, software defined radios or SDRs, do have characteristics that make them unique from other types of radios. As the name implies, a SDR is a radio that has the ability to be transformed through the use of software or re-definable logic. The fundamental characteristic of software radio is that"software defines the transmitted waveforms and software demodulates the received waveforms." This is in contrast to most radios in which the processing is done with either analog circuitry combined with digital chips. GNU radio is a free software toolkit for building software radios.

Keywords - USRP, SDR, GNU RADIO,FPGA.

I. Introduction

An SDR performs significant amounts of signal processing in a general purpose computer, or a reconfigurable piece of digital electronics. The idea behind software-defined radio is to do all that modulation and demodulation with software instead of using dedicated circuitry. The most obvious benefit is that instead of having to build extra circuitry to handle different types of radio signals, you can just load an appropriate program. One instant your computer could be an AM radio, the next a wireless data transceiver and then perhaps a HDTV set. This flexibility of software could be leveraged to do things that are difficult, if not impossible, with traditional radio setups.

Traditionally radio's were a hardware matter. They are often very cheap, but also very rigid. A radio created for specific transmit and receive frequencies and modulation schemes will never divert from these, unless its hardware is modified. The main idea behind Software Defined Radio (SDR) is to create versatile transceivers traditionally, hardware functions into the software domain. However a radio can never be purely software, because you need a way to capture and create the radio waves. Analog radio waves can be converted to digital samples using a Analog to Digital Converter (ADC) and vice versa using a Digital to Analog Converter (DAC). The ideal SDR scheme involves an antenna connected to a computer via an ADC for receiving and via a DAC for transmitting. All the processing on the signals, like (de)modulation, are then done in software, but the actual transceiving is done in the hardware subsystem.

SDR is the technique of getting the code as close to the antenna as possible. It turns Hardware radio problems into software problems.

II. Introduction

A universal SDR structure with the specific software (GNU Radio) and hardware (USRP/2) is given in Figure 1.

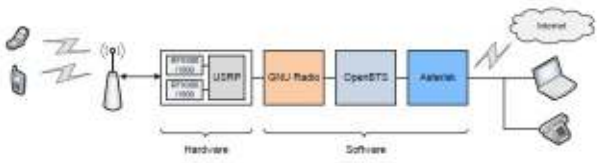

Fig1: Block diagram of SDR

Asterisk PBX, also is an open source project to act as a PBX and route the mobile calls through the VoIP and SIP.

All these softwares/peripherals when merged together, their functionality gives an amazing insight into solution for reduction of usage of the BTS towers and the relatively expensive charges paid for them.

III. GNU Radio And GRC

GNU radio is the free open source software toolkit for building software radios, in which software defines the transmitted waveforms and demodulates the received waveforms. GNU Radio provides functions to support & implement functions such as a spectrum analyzer, an oscilloscope, concurrent multichannel receiver and an ever growing collection of modulators and demodulators.

GNU Radio Companion (GRC) is a graphical user interface that allows you to build GNU Radio flow graphs.

Using GNU Radio, a radio can be built by creating a graph where the vertices are signal processing blocks and the edges represent the data flow between them. The GNU Radio components are connected using GRC tool. The signal processing blocks are implemented in C++ and the graphs are constructed and run in Python.

[1]Corresponding Author

Conceptually, a signal processing block processes an infinite stream of data flowing from its input ports to its output ports.

GNU Radio offers with its application GNU Radio Companion (GRC) the possibility to form a flow chart with graphical block elements. This application provides numerous predefined blocks, organized in different groups like signal sources, signal sinks as well as modulation and demodulation functions. As signal source for instance, USRP/2, audio card, wav files, signal generators or UDP/TCP ports may be used. Being installed with GNU Radio, GRC can be run from Linux by simply typing "grc" in an xterm shell.

IV. USRP

USRP (Universal Software Radio Peripheral) is the peripheral equipment for implementing the GNU Radio. The USRP acts as a BTS, replacing the mobile towers. This is highly efficient and economical in "small range mobile connectivity". The USRP is a hardware designed by Ettus Research to allow general purpose computers to function as high bandwidth software radios. In essence, it serves as a digital baseband and IF section of a radio communication system.

The basic design philosophy behind the USRP has been to do all of the waveform-specific processing, like modulation and demodulation, on the host CPU. All of the high-speed general purpose operations like digital up and down conversion, decimation and interpolation are done on the FPGA.

A large community of developers and users have contributed to a substantial code base and provided many practical applications for the hardware and software. The powerful combination of flexible hardware, open-source software and a community of experienced users make it the ideal platform for your software radio development.

USRP serves as interface between digital (host) and analog (RF) domain. In May 2009, the well-proven Universal Software Radio Peripheral (USRP) product became extended by an enhanced product named USRP2. USRP2 uses a different FPGA, faster ADCs and DACs with a higher dynamic range and a Gbit-Ethernet connection. All USRP daughterboard's can be used furthermore.

Daughter boards can be plugged into the USRP motherboard according to the specific frequency bands needed. These daughterboards can be hooked up to appropriate antenna's. On the receiving path (RX), a daughterboard captures the required frequency range and sends it through the PGA, possibly amplifying the signal, towards the ADC. The resulting digital signal is passed on to the FPGA. The FPGA and the host CPU both do some processing on the signal, and though the exact division of labor can be changed, standard the high speed general purpose processing, like down and up conversion, decimation, and interpolation are performed in the FPGA, while waveform-specific processing, such as modulation and demodulation, are performed at the host CPU. The USRPs have a 64 MHz crystal oscillator internal clock.

In USRP2 motherboard, an analog to digital converter (ADC) samples the received signal and converts it to digital values depending on the ADCs dynamic range of 14 bit. The digitized samples from ADC are mixed down to the desired IF by being multiplied with a sine respectively cosine function resulting in the I and Q path. The frequency is generated with a numerically-controlled oscillator (NCO) which synthesizes a discrete-time, discrete-amplitude waveform within the FPGA. Via the used NCO, very rapid frequency hopping is feasible.

Afterwards a decimation of the sampling rate is performed by an arbitrary decimation factor N. The sampling rate (fs) divided by N results in the output sample rate, sent to host. In transmit path, the same procedure is done vice versa using digital up converters (DUC) and digital analog converters (DAC).

V. Design &Implementation

This paper presents receiving AM signals. It uses a data file that contains several seconds of recorded signals from the AM broadcast band. This data file was obtained from the USRP. If you have a USRP available you could also use that as your input and receive live signals. The data file can be downloaded from the SDR Web page.

Construct the flow graph shown below consisting of a File Sink, Throttle, and FFT Sink. Set the Sample Rate in the variable block to 256000. This is the rate at which the saved data was sampled.

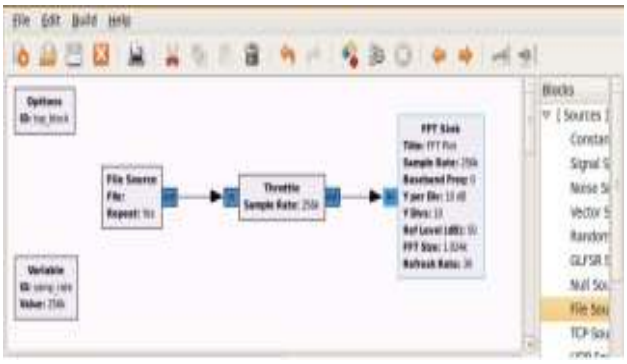

Fig2: Flow graph

When we execute the flow graph in fig2 the display spans a frequency range from just below -120KHz to just above 120KHz. This exact span is 256KHz, which

corresponds to the Sample Rate that the data was recorded at.

The peaks that we observe on the display in fig3 corresponds to the carriers for AM broadcast signals. You should also be able to observe the sidebands for the stronger waveforms.

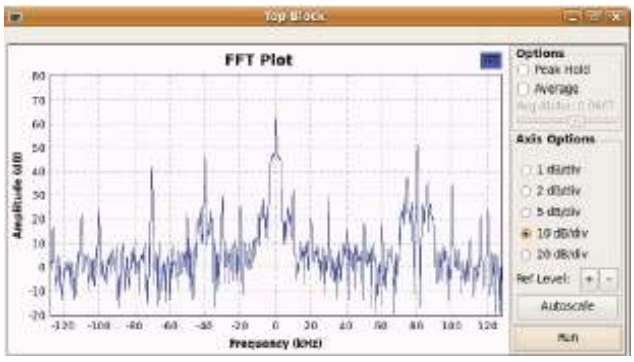

Fig3: Waveform Window

In this step we will expand the frequency scale on the FFT display so that we can view the signals with greater resolution. While we cannot change the original data, we can resample it to either increase or decrease the sample rate. We will decrease the sample rate by using decimation. Modify the flow graph as follows.

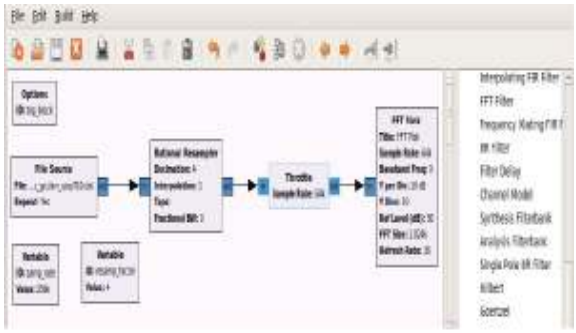

Fig4: Block Diagram for Frequency Adjustment

Add a Variable block (under Variables menu). Set the ID to resamp_factor and the Value. Add the Rational Resampler from the Filters menu. Set its decimation factor to resamp_factor. That means that it will divide the incoming data rate by the decimation factor. In this example, the incoming 256K samp/sec data will be converted down to 256K/4 = 64K samp/sec. Execute the new flow graph. You should now observe a frequency span of only 64KHz (-32KHz to +32KHz). The bandwidth of an AMbroadcast signal is 10KHz (+/- 5KHz from the carrier frequency).

VI. Conclusion

This work integrates many aspects of SDR projects based on GNU Radio and USRP/2. The given text deals with the wide application- and research-oriented field of Software-Defined Radio (SDR). With the open source project GNU Radio and the hardware platform USRP/2, very complex wireless transmission systems can be explored, even with a relatively small budget. Principally all of the needed modules and information can be found in the internet. It is the credit of this paper to bring these widespread information together.

In our paper, we have first introduced the concept of software-defined radio. Afterwards, we describe the related hardware support and development environment respectively. Then, we have explained about what we have implemented based on our demo. Through this Research, we have build up knowledge and experience for developing GNU Radio.

Although the SDR technological advances are promising, SDR technology suitable for use by the public safety community is still in an early stage. This is due to several factors. SDR technology has the potential to cause interference with other existing radio systems. Software radio is an exciting field, and GNU Radio provides the tools to start exploring. A deep understanding of software radio requires knowledge from many domains. We're doing our best to lower the barriers to entry.

VII. Acknowledgments

Through the course of my research i have been fortunate to enjoy unwavering support and encouragement of my husband M.Sunny Sheldon & my Daughter Sherlyn Andrea, my parents and my Inlaws. All the Authors are also thankful to Directors of KMIT for Encouraging and supporting us by facilitating all amenities required.

References

[1] www.gnu.org/software/gnuradio/gnuradio.html

[2] J.Mitola III Software Radio Architecture. Wiley-Interscience, 2000.

[3] Blossom, Eric. *Exploring GNU Radio.* http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html.

[4] Ettus, Matt (et al.). *USRP2GenFAQ.* http://www.gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ

[5] *The OpenBTS Project* http://www.openbts.sourceforge.net.

[6] European Defence Agency, "Back-ground on Software Defined Radio," Nov. 2007.

[7] Abidi, "The path to the software-defined radio receiver," Solid- State Circuits, IEEE J., vol. 42, no. 5, pp. 954– 966, 2007.

[8] E. Buracchini, "The software radio concept," Commun. Mag., IEEE, vol. 38, no. 9, pp. 138–143, Sept. 2000.