

Real Time Voice transmission over Wireless medium using GNU Radio and USRP

EEC 687 Mobile Computing Project Report

By

Darshana Shah

Department of Electrical & Computer Engineering

Cleveland State University, OH

d.shah44@csuohio.edu

Submitted to Dr. Chansu Yu

06/07/2009

Abstract

My project is to transmit real time voice over wireless medium. For that I used two USRPs (one for transmission and one for reception) as hardware platform and GNU Radio as software environment for encoding and modulating voice at transmitter side and for decoding and demodulating voice at receiver side. First section is introduction. Second section is background of the project. Third section shows project details. It has three subsections. They are project goal, experimental setup, past work and results. Fourth section is Discussion. Fifth section is future work and sixth and last section is References.

1. Introduction

SDR (Software Defined Radio) implements everything in software. E.g. filter, amplifier, encoder/decoder, modulator/demodulator etc. Free SDR tool kit is available online i.e. GNU Radio. It has three major advantages over hardware radio. First advantage is SDR is cheap compared to hardware because we do not need to buy encoders, modulators and all other components. Second advantage is debugging is easy compared to hardware because if we run the program in software it shows error message so we can easily find it and correct it and thus it saves time. On the other hand in hardware we need to take out each component one by one and need to test it until we find the error, so it becomes quite lengthy and time consuming process. Third advantage is we can modify code in GNU radio software and make it according to our requirement. It can transmit large data or voice files using different hardware E.g. USRP. I use GNU Radio as software environment and USRP as hardware platform to transmit real time voice from one place to other over wireless medium.

2. Background of the project

GNU Radio

It is software defined radio. It has everything in software rather than hardware. It brings software code as close as possible to the antenna. In my project I use it to perform encoding, assembling and channel modulation at transmitter side and decoding, disassembling and channel demodulation at receiver side. It is available online and it is free. It uses C classes to implement different signal processing functions and python for main application programming. Swig is used as a glue to connect it all together. Swig is a Linux package that converts C classes to python compatible classes.

USRP

It supports wide range of frequencies for wireless communication as one USRP mother board can support up to four daughter boards. Thus it can cover frequency from analog range to some GHz. Following several kinds of daughter boards are available.

BasicTX -- 2 MHz to 200 MHz Transmitter

BasicRX -- 2 MHz to 300+ MHz Receiver

LFTX -- DC-30 MHz Transmitter
 LFRX -- DC-30 MHz Receiver
 TVRX -- 50 MHz to 870 MHz Receiver
 DBSRX -- 800 MHz to 2.4 GHz Receiver
 RFX400 -- 400-500 MHz Transceiver
 RFX900 -- 800-1000MHz Transceiver
 RFX1200 -- 1150 MHz - 1450 MHz Transceiver
 RFX1800 -- 1.5-2.1 GHz Transceiver
 RFX2400 -- 2.3-2.9 GHz Transceiver, 20+mW output

I use two RFX2400 daughter boards of 2.4GHz one at transmitter side and one at receiver side. Figure 1a shows hardware of USRP.

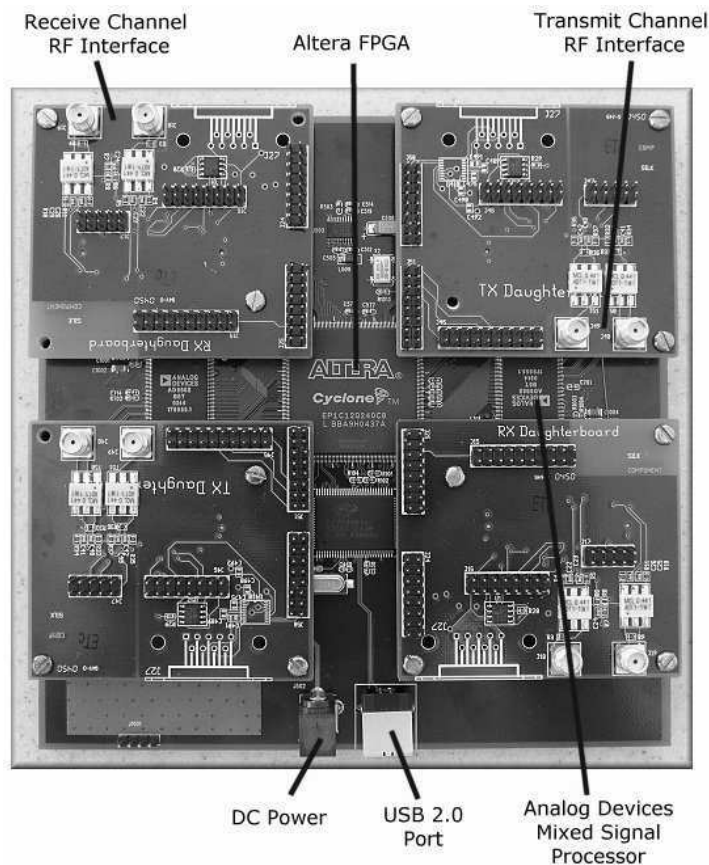


Figure 1a
Hardware Diagram of USRP

USRP has following main blocks.
RF front end, ADC/DAC, FPGA and Programmable USB controller.

How it works?

In this section I will explain how GNU Radio and USRP work together. It is shown in figure 1b. We need to install GNU Radio software in computers. Sender sends the signal from GNU Radio software installed in computer after applying required functions from program. Through USB

cable we can connect PC to USRP. The first block of USRP is FPGA which is programmable. The function of FPGA is to up convert the signal and make it suitable to DAC. DAC converts digital signal which is user code to analog signal and send it to RF front end. RF front end sets the frequency of the signal suitable to antenna and transmits it to antenna.

At receiver side antenna captures signal and sends it to RF front end. RF front end makes signal suitable to ADC by changing its frequency. ADC converts analog signal to digital and send it to FPGA. FPGA down converts the signal. Now through USB cable we can apply this digital signal to GNU driver which is installed in the PC. So we can receive signal from the USRP and can apply functions of GNU Radio and observe it from PC.

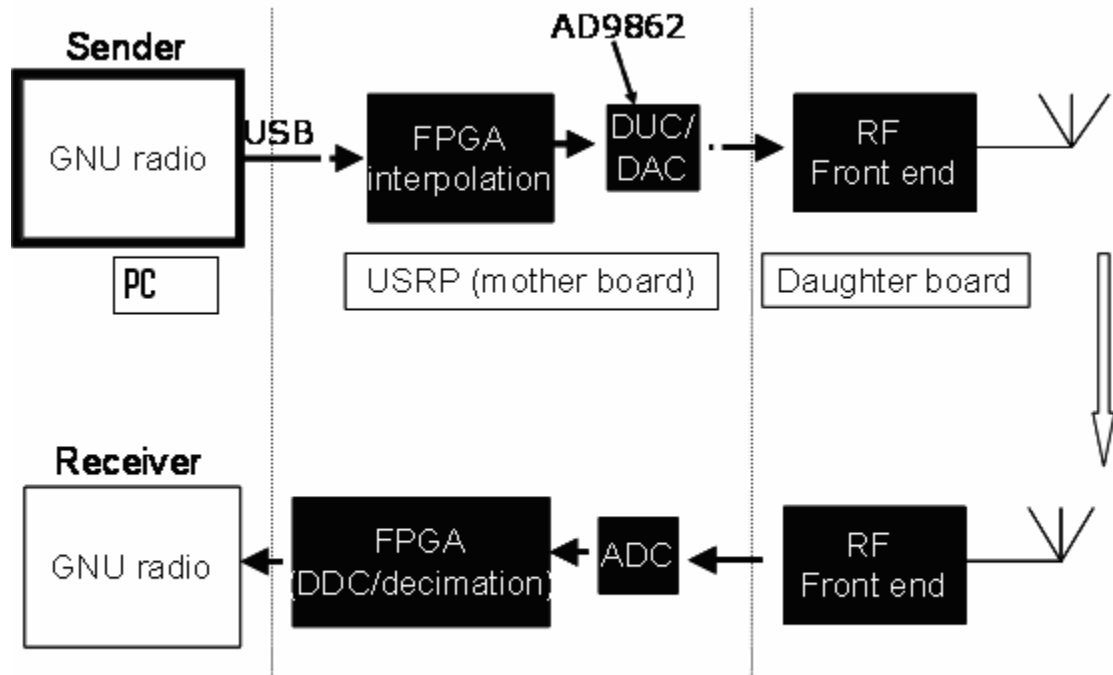


Figure 1b General Block Diagram

3. Real Time Voice Transmission over Wireless Medium

This part includes three subsections. 3.1 introduce project goal and block diagram of the project. 3.2 show experimental setup. 3.3 covers past work and finally 3.4 shows result.

3.1 Project Goal

My project goal is to transmit real time voice through wireless medium. Basic idea is shown in figure 2. For that I use two USRP connected to two different PCs one at transmitter side and other at receiver side. I installed GNU Radio software in both PCs. My project goal is to speak something in mike attached with one pc at transmitter side and it should be hear from speaker attached to other PC at the same time. So I need encoder at transmitter side which encodes voice and decoder at receiver side to decode voice. I use GSM-FR encoder and decoder at transmitter

side and receiver side respectively. It is already available in GNU Radio tool kit. I use tx_voice.py and rx_voice.py from GNU Radio examples for this project. It is for real time voice application. GSM FR was the first digital speech-coding standard which was used in GSM digital mobile phone system. I use GMSK modulation and demodulation. In GMSK modulation signal is first applied to Gaussian filter and then to the modulator. Its spectral efficiency is high means it can transmit faster than QPSK and other modulation, which is necessary for real voice transmission.

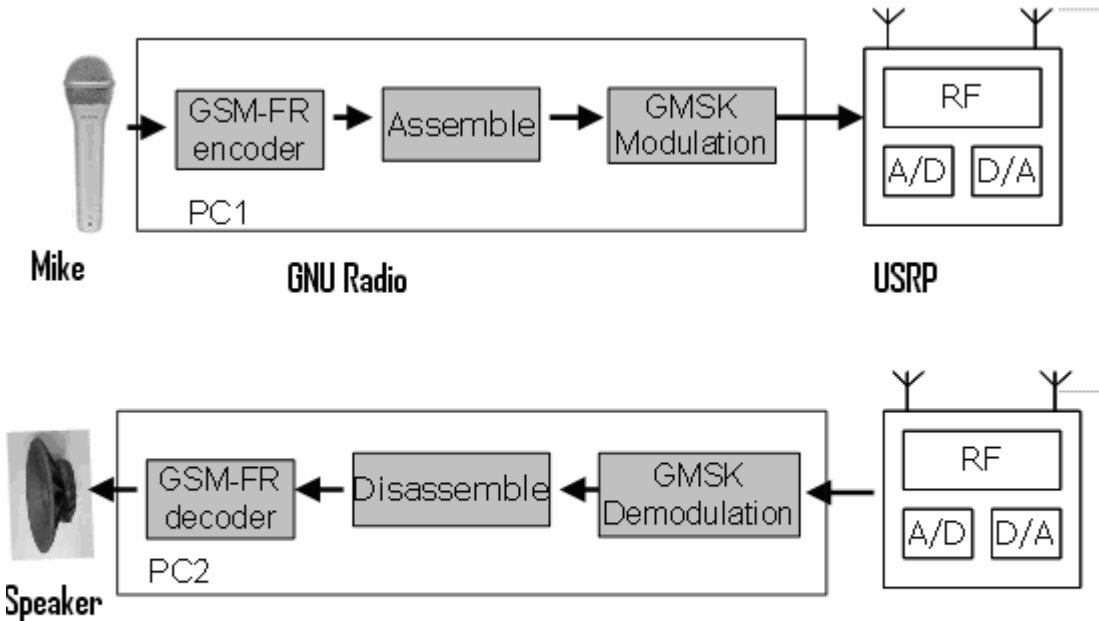


Figure 2 Block Diagram for Real Time Voice Transmission

3.2 Experimental Setup

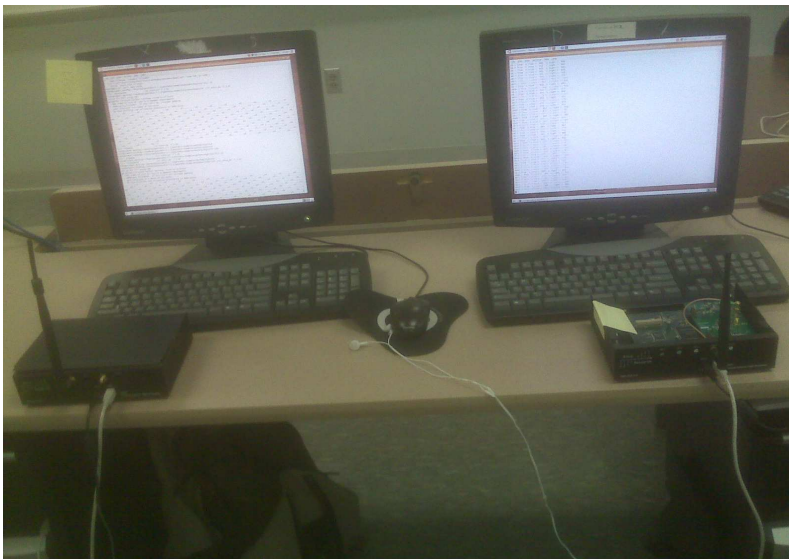


Figure 3 Project Setup

Figure 3 shows experimental setup of the project. I have connected one USRP at transmitter side to which mike is connected and one USRP at receiver side to which speaker is connected.

3.3 Past Work

- I had run different audio files and USRP files from GNU Radio examples and it works fine.
- I tried to do data communication between two USRP and it was also working. For that I used `benchmark_tx.py` and `benchmark_rx.py` from GNU Radio examples.
- I tried to transfer real time voice using `tx_voice.py` and `rx_voice.py`. But it was not working. I always got false packets and sometimes did not receive anything.

3.4 Results

- Now I am able to transmit real time voice over wireless medium. If I speak in microphone connected to one PC I can hear it from speaker connected to other PC.
- I tried to play online Internet radio on one PC and I can hear it from other PC. So I can play any online voice application using this project.

4. Discussion

At the time of midterm project presentation I got false packets and I was not able to transmit real time voice. So I thought there is some frequency matching problem so it does not work. Then I tried different daughter boards but got the same problem. Then I tried to test both USRP one by one by running some audio files from GNU Radio examples and it was working. So I got confused what to do. I tried to use different RF channel antenna. I got many errors to make this program run and I solved it one by one. Finally it starts working. I was able to transmit real time voice and then I am thinking about to transmit online Internet radio. It is also little bit tricky and I got true result means I am able to transmit online radio.

5. Future Work

- We can make compact version of USRP like Ethernet card and can replace it because USRP is compatible with GNU Radio and it is programmable so we can do lots of things compared to Ethernet card. Obviously not for home application because USRP is costly but at industry level it becomes very useful.
- Transmission over wireless medium using router needs file sharing. That is security hole. Once computer port becomes open hijacker can easily hijack it. If you want to access file system without file sharing using routers we need to buy software which is very costly still not programmable. Using USRP there is no security hole. No file sharing is needed. So for secured data and voice communication we can use this application. We can program USRP using FPGA according to our requirement but can not program router accept some parameter settings.

6. References

- [1] <http://www.wu.ece.ufl.edu/projects/softwareRadio/#Project%20paper>
- [2] http://users.ecel.ufl.edu/~zhifeng/project/GNU_Radio_USRP/index.htm
- [3] Software Defined Radio: with GNU Radio and USRP by Cory Clark
- [4] <http://www.gnuradio.org/trac>
- [5] Lecture notes of Dr. Yu, which is available at,
http://academic.csuohio.edu/yuc/mobile09/09_lab04_week05_usrp_ipaq.pdf
- [6] <http://www.ettus.com/>
- [7] <http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html>
- [8] http://en.wikipedia.org/wiki/GMSK#Gaussian_minimum-shift_keying

Appendix

I have used tx_voice.py and rx_voice.py from GNU Radio examples for this project. If you get any problem while running this project please let me now.