Software Defined Radio using GNU Radio and USRP

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree

of
BACHELOR OF TECHNOLOGY
in
ELECTRICAL ENGINEERING

Submitted by:

Suman Kumar Das

Guided by:

Dr. Vimal Bhatia
Associate Professor

INDIAN INSTITUTE OF TECHNOLOGY INDORE
December 2017

CANDIDATE'S DECLARATION

I hereby declare that the project entitled "**Software Defined Radio using GNU Radio and USRP**" submitted in partial fulfilment for the award of the degree of Bachelor of Technology in Electrical Engineering completed under the supervision of "**Dr.Vimal Bhatia**, Associate Professor in Discipline of Electrical Engineering, IIT Indore" is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Suman Kumar Das
(ee140002033)

Certificate by BTP Guide

It is certified that the above statement made by the student is correct to the best of my knowledge.

Vimal Bhatia
Associate Professor
Discipline of Electrical Engineering
Indian Institute of Technology Indore
(Project Guide)

PREFACE

This report on **"Software Defined Radio using GNU Radio and USRP"** is prepared under the guidance of Dr. Vimal Bhatia, Associate Professor, Discipline of Electrical Engineering, IIT Indore.

Through this report I have tried to give a detailed description of implemented waveforms on software defined radio and tried to cover every aspect of its implementation.

I have tried to the best of my abilities and knowledge to explain the content in a lucid manner. I have also added figures to make it more illustrative.

Suman Kumar Das
B.Tech IV year
Discipline of Electrical Engineering
IIT Indore

ACKNOWLEDGEMENTS

I would like to thank my BTP supervisor **Dr. Vimal Bhatia** for his constant support and valuable feedback which helped me in the course of the project. He gave me opportunity to discover and work in this domain. He guided me thoroughly and pulled me out of the craters of failures I faced all through the period.

I also appreciate the efforts and time given by **Mr. Abhijeet Bishnu**, PhD scholar, who guided me, explained me the problems while working with software defined radio and provided me the initial pathway for starting this project in right manner. He guided me with useful direction to proceed along whenever necessary.

I would also like to extend a special thank you to all my friends who have always encouraged me to give in my best towards this project. It is their help and support, due to which I was able to complete this project and technical report. Lastly, I am grateful to my family members too for their support. Without their support this report would not have been possible.

Suman Kumar Das
B.Tech IV year
Discipline of Electrical Engineering
IIT Indore

ABSTRACT

Software Defined Radio (SDR) is defined as : " **Radio in which some or all of the physical layer functions are software defined**". SDR is an adaptive, future-proof solution that makes wireless networks highly flexible. It can replace conventional radio hardware with re-configurable, re-programmable radios, opening the way for new services and prolonging a mobile wireless device's lifespan. Due to its ability to create radios that change on demand, it is very suitable for testing dynamic wireless video performance under software controlled parameters.

GNU Radio is an open-source and free software radio toolkit for learning about building and deploying SDRs. GNU Radio is a signal processing package, which also provides functions to support implementing spectrum analyzer, an oscilloscope, concurrent multi-channel receiver for an ever-growing collection of modulators and demodulators. So, in this project, GNU Radio is chosen for development environment.

Universal Software Radio Peripheral (USRP) is a high-speed universal serial bus based board for making software radios. The USRP has an open design, with freely available schematics, drivers and free software to integrate with GNU Radio.

In this project, various implementation and testing of SDR using USRP B210 development board are being done i.e. 1080p real time DVB transmission and reception, FM Radio broadcast and reception, user datagram protocol streaming and Open-BTS local global system for mobile base-station.

The main objective of this project is to realize practical working of radio frequency (RF) communication systems. SDR with USRPs could be used for analyzing real time RF prototype or implementing some advanced algorithms to realize deviation of theoretical case from practical case.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ADC Analog to Digital Converter

ASIC Application Specific Integrated Circuit

BER Bit Error Rate

CENELEC European Committee for Electrotechnical Standardization

COFDM Coded OFDM

CP Cyclic Prefix

CPU Central Processing Unit

CR Cognitive Radio

DAB Digital Audio Broadcasting

DAC Digital to Analog Converter

DTT Digital Terrestrial Television

DVB-H Digital Video Broadcasting- Hand-held

DVB-S Digital Video Broadcasting-Satellite

DVB-T Digital Video Broadcasting-Terrestrial

DVB Digital Video Broadcasting

EBU European Broadcasting Union

ETSI European Telecommunications Standards Institute

FFT Fast Fourier Transform

FM Frequency Modulation

FPGA Field Programmable Gate Array

GMSK Gaussian Minimum Shift Keying

GNU GNU's not Unix

GRC GNU Radio Companion

GSM Global System for Mobile Communication

HD High Definition

HDTV High Definition TV

IFFT Inverse Fast Fourier Transform

JTC Joint Technical Committee

LabVIEW Laboratory Virtual Instrument Engineering Workbench

LTE Long-Term Evolution

MIMO Multiple Input and Multiple Output

MPEG Moving Picture Experts Group

OFDM Orthogonal Frequency Division Multiplexing

OTG On The Go

PC Personal Computer

PRBS Pseudo Random Binary Sequence

QCIF Quarter Common Intermediate Format

QoS Quality of Service

RFIC Radio Frequency Integrated Circuit

RF Radio Frequency

RS Reed-Solomon

SDR Software Defined Radio

SISO Single Input and Single Output

SNR Signal to Noise Ratio

SWIG Simplified Wrapper and Interface Generator

TDM Time Division Multiplex

TS Transport Stream

TV Television

UDP User Datagram Protocol

UHD USRP Hardware Driver

USB Universal Serial Bus

USRP Universal Software Radio Peripheral

WLAN Wireless Local Area Network

Chapter 1

Introduction

1.1 Overview

A software defined radio (SDR) is a wireless device that turns a PC into a next-generation wireless prototyping tool. It consists of a configurable RF front end with an FPGA or programmable system-on-chip (SoC) to perform software based signal processing functionality. A single USRP B210 [1] can operate on continuous RF coverage from 70 MHz – 6 GHz, which can be used to implement wireless standards from FM radio to Wi-Fi and LTE.

SDR is the culmination of advances on several fronts and probably the most significant area of development in radio systems today. The entire worldwide cellular system uses SDR. NASA and the US military communications are now almost exclusively using SDR. It is only possible due to continuous RF coverage, real-time bandwidth and software controlled reconfigurable FPGA.

1.2 What is SDR?

In the past, Radio systems were used to communicate using one or two signals operating at different frequencies. Thus, different groups of people using different types of radio systems could not communicate with each other due to incompatibility between their radios. This was a major problem during war. The problem of communication among people using different types of radio equipment can be solved by using Software Programmable Radios.

Figure 1.1 shows the working of SDR [2] . Wireless engineers can use software-defined radio hardware as a cost-effective, real-time platform for a range of wireless engineering tasks, including [3] :

- Over-the-air lab and field testing with live RF signals

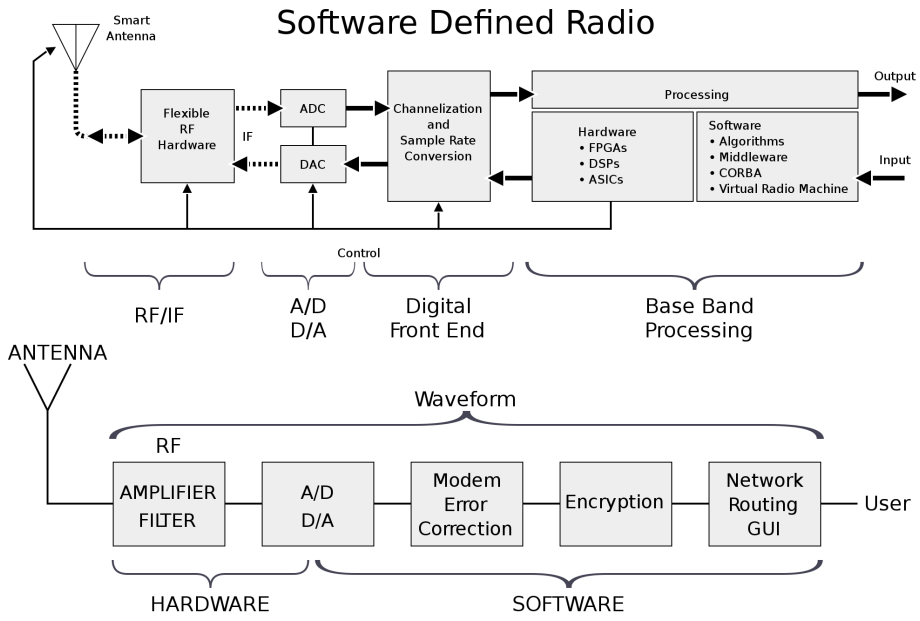


Fig. 1.1 Working of SDR.

- Rapid prototyping of custom radio functions
- Hands-on learning of wireless communications concepts and design skills

1.3 Objectives

The main objective of this project is to implement a working prototype of real time Full HD video transmission using open source technologies. Along with this the aim was to perform some tasks as a preliminary, e.g. FM Transmitter Design FM Receiver Design, Implementation of Open-BTS GSM network, UDP real time QCIF (176 × 144) video streamming and Implementation of DVB-T transmitter using GNU Radio [4] .

1.4 Methodology

At first to familiarize with GNU Radio and USRP boards some preliminary tasks were performed. In order to implement the DVB-T standard [5] [6] , DVB transmitter is taken as a reference model. The source code of this model is available on github [7] . This transmitter model is being implemented on USRP B210 board. The flow graph of this reference model contains many blocks which are connected to each other to perform signal processing. These

blocks are modified according to the DVB-T standard and the transmitter system is tested on Ettus Reseach USRP B210 board [8] .

For DVB-T receiver, DVB-T driver app is used. This driver app is available on Google play store. After demodulating the received signal, the real time video is being converted into TS file and played by VLC media player in real time.

1.5 Report Layout

The report is organized as follows:

- **Chapter 2 (Literature Review) :** This chapter presents an overview to the DVB-T2 a Digital Video Broadcasting standard which is developed for terrestrial broadcasting. This chapter describes this standard and the technical specifications as proposed by this standard.
- **Chapter 3 (Hardware and Software):** This chapter presents an overview of SDR, hardware components, software. It describes the software and hardware components which we used in our implementation with their specification. At the end, an experiment is discussed which is performed using these hardware and software components to have an idea of SDR.
- **Chapter 4 (Design and Implementation) :** This chapter details the project design and implementation. Also, it presents the design methodology and tools used to implement the system.
- **Chapter 5 (Results, Conclusion and Future Work) :** This chapter presents the objectives achieved in our experiment, in addition to the future developments that can be made to the resulting system.

Chapter 2

Literature Review

2.1 Overview

DVB (Digital Video Broadcasting) is an organization consisting of leading digital TV and technology companies. It consists of developers, manufacturers, network operators, broadcasters and regulators who work to design open technical standards for broadcast and content delivery services.

[9] Digital multimedia broadcasting is available in more and more countries with various forms. One of the most successful forms is Digital Video Broadcasting for Terrestrial (DVB-T), which has been deployed in most countries of the world for years. In order to bring the digital multimedia broadcasting services to battery-powered handheld receivers in a mobile environment, Digital Video Broadcasting for Handheld (DVB-H) has been formally adopted by ETSI. More advanced and complex digital multimedia broadcasting systems are under development, for example, the next generation of DVB-T, a.k.a. DVB-T2. Current commercial DVB-T/H receivers are usually built upon dedicated application-specific integrated circuits (ASICs). However, ASICs are not flexible for incoming evolved standards and less overall-area efficient since they cannot be efficiently reused and shared among different radio standards, when we integrate a DVB-T/H receiver into a mobile phone. This paper presents an example implementation of a DVB-T/H receiver on the prototype of Infineon Technologies' Software-Defined Radio (SDR) platform called MuSIC (Multiple SIMD Cores), which is a DSP-centered and accelerator-assisted architecture and aims at battery-powered mass-market handheld terminals.

2.2 Digital Video Broadcasting

[9] Digital Video Broadcasting (DVB) is a set of internationally open standards for digital television. DVB standards are maintained by the DVB Project an international industry consortium with more than 270 members, and are published by a Joint Technical Committee (JTC) of the European Telecommunications Standards Institute (ETSI), European Committee for Electrotechnical Standardization (CENELEC) and European Broadcasting Union (EBU).

Indian Public Broadcaster Doordarshan has launched a free DVB-T2 mobile TV service in 16 cities, aimed at the increasing number of smartphones in India. The service was officially launched on February 25 in Delhi, Mumbai, Kolkata and Chennai, Guwahati, Patna, Ranchi, Cuttack, Lucknow, Jalandhar, Raipur, Indore, Aurangabad, Bhopal, Bangalore and Ahmedabad.

2.2.1 DVB Variants

Even a quick look at DVB will reveal the fact that there are many flavours of the basic standard. In these days when there are many ways in which television can be carried from the "transmitter" to the "receiver" no one standard can be optimised for all applications. As a result there are many different forms of the Digital Video Broadcasting, DVB, standards, each designed for a given application.

The main forms of DVB are summarized below:

2.2.2 DVB-T2

What is DVB-T2?

DVB-T2 is the world's most advanced digital terrestrial television (DTT) system, offering more robustness, flexibility and 50% more efficiency than any other DTT system. It supports SD, HD, UHD, mobile TV, radio, or any combination thereof.

Background

Since its publication in 1997, over 70 countries have deployed DVB-T services but this is now rapidly changing with the massive and fast adoption of DVB-T2 around the world. Both DVB-T and T2 benefits from massive economies of scale and very low receiver prices.

Due to the European analogue switch-off and increasing scarcity of spectrum, DVB drew up Commercial Requirements for a more spectrum-efficient and updated standard. DVB-T2

Table 2.1 Major DVB standards

DVB STANDARD	MEANING	DESCRIPTION
DVB-C	Cable	The standard for delivery of video service via cable networks.
DVB-H	Handheld	DVB services to handheld devices, e.g. mobile phones, etc
DVB-RSC	Return satellite channel Satellite	DVB services with a return channel for interactivity.
DVB-S	Satellite services	DVB standard for delivery of television / video from a satellite.
DVB-SH	Satellite handheld	Delivery of DVB services from a satellite to handheld
DVB-S2	Satellite second generation	The second generation of DVB satellite broadcasting.
DVB-T	Terrestrial	The standard for Digital Terrestrial Television Broadcasting.

easily fulfils these requirements, including increased capacity, robustness and the ability to reuse existing reception antennas. The first version was published in 2009 (EN 302 755) and the 2011 update added the T2-Lite subset for mobile and portable reception [9].

Table 2.2 Comparison between DVB-T & DVB-T2.

	DVB-T	**DVB-T2**
FEC	Convolutional Coding and Reed Solomon 1/2, 2/3, 3/4, 5/6, 7/8	LDPC + BCH 1/2, 3/5, 2/3, 3/4, 4/5, 5/6
Modes	QPSK, 16QAM, 64QAM	QPSK, 16QAM, 64QAM, 256QAM
Guard Interval	1/4, 1/8, 1/16, 1/32	1/4, 19/128, 1/8 19/256, 1/16, 1/32, 1/128
FFT Size	2k, 8k	1k, 2k, 4k, 8k, 16k, 32k
Scattered Pilots	8% of total	1%, 2%, 4%, 8% of total
Continual Pilots	2.0% of total	0.4%-2.4% (0.4%-0.8% in 8K-32K)
Bandwidth	6, 7, 8 MHz	1.7, 5, 6, 7, 8, 10 MHz
Typical data rate	24 Mbit/s	40 Mbit/s
Max. data rate (@20 dB C/N)	31.7 Mbit/s (using 8 MHz)	45.5 Mbit/s (using 8 MHz)
Required C/N ratio (@24 Mbit/s)	16.7 dB	10.8 dB

How does it work?

Additional new technologies used in DVB-T2 [9] [10] are:

- **Multiple Physical Layer Pipes** allow separate adjustment of the robustness of each delivered service within a channel to meet the required reception conditions (for example in-door or roof-top antenna). It also allows receivers to save power by decoding only a single service rather than the whole multiplex of services.
- **Constellation Rotation** provides additional robustness for low order constellations.
- **Extended interleaving**, including bit, cell, time and frequency interleaving.
- **Future Extension Frames** (**FEF**) allow the standard to be compatibly enhanced in the future.

As a result, DVB-T2 can offer a much higher data rate than DVB-T OR a much more robust signal. For comparison, the two bottom rows show the maximum data rate at a fixed C/N ratio and the required C/N ratio at a fixed (useful) data rate.

Market Deployment

Like DVB-T, DVB-T2 targets not just roof-top and set-top antennas, but also PCs, laptops, in-car receivers, radios, smart phones, dongles, and a whole range of other innovative receiving devices. In countries where DVB-T services are already on air DVB-T and DVB-T2 services are likely to coexist side-by-side for some time, but many green-field countries that had not yet deployed DTT services, jumped directly to DVB-T2. A future-proof solution!

Almost all modern TV sets sold in DVB countries now have integrated DVB-T2 tuners and DVB-T2 receiver prices have rapidly dropped towards the level of DVB-T prices.

[9] The first DVB-T2 service was launched in the UK in March 2010. Sweden and Finland followed shortly and almost every European country now has far-advanced plans to switch from DVB-T to T2. In Africa DVB-T2 pay-TV services were launched in Zambia, Namibia, Nigeria, Kenya and Uganda and many other countries on the continent have followed since. The Middle-East, India and the Asia-Pacific region also have selected DVB-T2 in the past years. South-America includes multiple T2-only countries and even non-T2 countries like Argentina that also feature DVB-T2 pay-TV services.

So far, 166 countries have adopted or deployed DVB-T and DVB-T2 [9]. A true global standard!

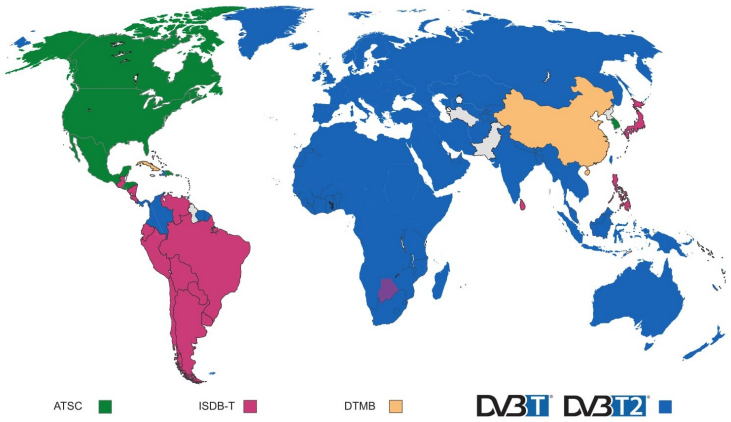


Fig. 2.1 Digital TV systems being used all over the world.

2.3 OFDM Modulation Scheme

OFDM [11] is a form of transmission that uses a large number of close spaced carriers that are modulated with low rate data, maintaining total data rates similar to conventional single carrier modulation schemes in the same bandwidth. In a normal FDM system, many carriers are spaces apart in such a way that the signals can be received using conventional filter and demodulators. In such receivers, guard bands are introduced between carriers which results in a lowering of spectrum efficiency in frequency domain. To increase spectrum efficiency it is possible to arrange carriers in such a way that sidebands of individual carriers overlap and the signals are still received without interference of adjacent carrier. To do this the carriers must be orthogonal and this encoding scheme is known as OFDM.

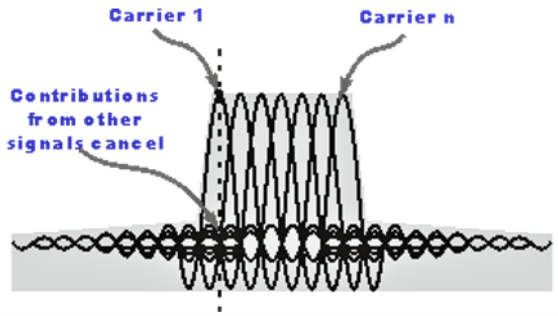


Fig. 2.2 OFDM Spectrum

The advantage of OFDM transmission scheme are:

- In this scheme channel equalization is simplified compare to single carrier system because the frequency spectrum is divided in slowly modulated narrow bands signals in place of one rapidly modulated wide band signal.
- It removes the ISI through the use of guard interval (cyclic prefix).
- This scheme provides flat fading sub-channels by dividing the channel into narrow band.
- Spectrum efficiency is increased by allowing overlap.
- OFDM makes Single Frequency Networks possible, where several adjacent transmitters send the same signal simultaneously at the same frequency, at the receiver all the signals from multiple transmitters are combined constructively, rather than interfering. In single carrier system all the signals will interfere with each other.

There are also some drawbacks of this transmission scheme which are as follows:

- This scheme is more sensitive to carrier frequency offset and phase noise.
- OFDM has a large peak to average power ratio compared to single carrier system.

Chapter 3

Hardware and Software

3.1 Overview

This chapter aims to understand how the hardware and software can be combined together to make a software defined radio. This chapter begins with a brief overview of software defined radio. Following that, the software and hardware components which we used for our implementation are explained with their specifications.

3.2 Software Defined Radio

Software Defined Radio (SDR) is a "Radio in which some or all of the physical layer functions are software defined. In this technology these software defined functions are implemented on real time hardware platform. If there is need for modification of these physical layer functions that can be done in software. There is no need to redesign hardware. The modified modules are re-implemented on same hardware. This saves a lot of cost of redesigning the hardware. SDR provides greater reconfigurability and flexibility than conventional hardware defined radios that makes them very popular in today's world. The basic block diagram of SDR is shown in Figure 2.1. SDR consists of three parts. The first one is Host PC in which our software is installed. In this block according to our module signal processing part for ex:- Modulation, Demodulation, Coding etc. is done. Second and third one are hardware components. The second block which is called as Motherboard is used to up-convert/down-convert the IF signal to/from the transmission frequency. In this block baseband and Intermediate Frequency (IF) signal is processed only in digital domain. The third block is Daughter board. This block is used for receiving or transmitting signal in analog domain. While transmitting this block amplify the up converted signal and then it

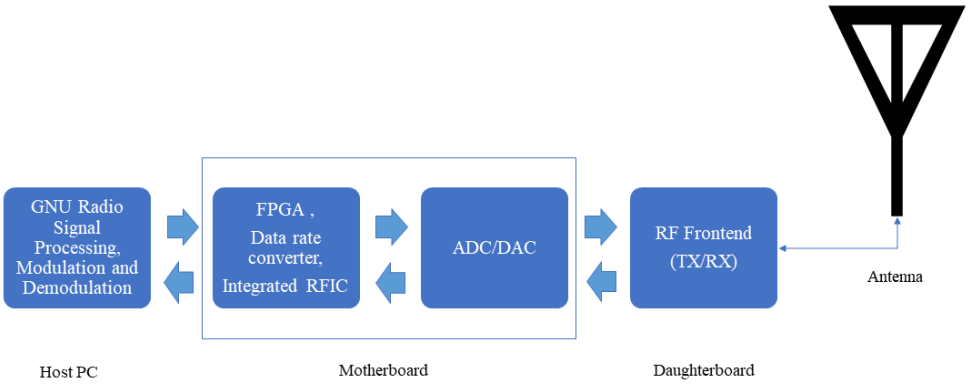


Fig. 3.1 SDR block diagram

brought to an antenna , where it is ready for transmission. When receiving the task of this block is to amplify the signal and brought the signal to the Motherboard.

3.3 Softwares

3.3.1 GNU Radio

GNU Radio [4] is a free and open source software development toolkit that provides the signal processing runtime and processing blocks to implement software radios. It can be used with low-cost external hardware to create SDR, or without hardware in a simulation like environment. In this Software all the signal processing blocks which are used in communication system can be implemented. Many of blocks like modulator, demodulator, decoders, encoders, graphical sinks (e.g. FFT), synchronization elements, equalizers etc. are already implemented. According to our application we can modify these blocks or implement new blocks.

For the implementation of new blocks GNU Radio works on two programming languages. All the applications are primarily written using the Python programming language, while the supplied, performance-critical signal processing path is implemented in C++. SWIG tool is used to create the interface between C++ and Python.

There are two ways to make an application. First one is by using GRC, a graphical programming interface. In this method we do not need to write the python code. The GRC facilitates the user to make flow graphs by dragging and dropping the blocks from library. When this flow graph is run a python file is generated in same directory. This Python file contains the flow graph which the Python interpreter runs. Second method is without using GRC. In this method we have to write our own python script to implement any model. In both methods when a python interpreter runs a flow graph it calls all the corresponding functions and libraries through SWIG tool.

3.3.2 LabVIEW

LabVIEW [12] is another software development toolkit like GNU Radio which is used to implement software radios. It is not an open source software. This software is provided by National Instruments (NI).

LabVIEW includes a compiler that produces native code for the CPU platform. This aids performance. The graphical code is translated into executable machine code by interpreting the syntax and by compiling. The LabVIEW syntax is strictly enforced during the editing process and compiled into the executable machine code when requested to run or upon saving. In the latter case, the executable and the source code are merged into a single file. The executable runs with the help of the LabVIEW run-time engine, which contains some precompiled code to perform common tasks that are defined by the G language. The run-time engine reduces compiling time and provides a consistent interface to various operating systems, graphic systems, hardware components, etc. The run-time environment makes the code portable across platforms. Generally, LabVIEW code can be slower than equivalent compiled C code, although the differences often lie more with program optimization than inherent execution speed.

3.3.3 MATLAB

MATLAB® and Simulink® connect to USRP software-defined radios (SDR) from Ettus Research to provide a radio-in-the-loop design and modeling environment. With this support package, Communications System Toolbox, and a USRP radio, you can design and verify practical SDR systems.

MATLAB and Simulink Support Package for USRP Radio includes:

- Functions and System objects for connecting MATLAB to UHD-based USRP radios
- Blocks for connecting Simulink to UHD-based USRP radios

Communications System Toolbox is also necessary for this project.

3.4 Hardwares

3.4.1 USRP B210

Overview

The USRP B210 provides a fully integrated, single-board, Universal Software Radio Peripheral (USRP™) platform with continuous frequency coverage from 70 MHz – 6 GHz [1]. Designed for low-cost experimentation, it combines the AD9361 RFIC direct-conversion transceiver providing up to 56 MHz of real-time bandwidth, an open and reprogrammable Spartan6 FPGA, and fast SuperSpeed USB 3.0 connectivity with convenient bus-power. Full support for the USRP Hardware Driver™ (UHD) software allows us to immediately begin developing with GNU Radio, prototype own GSM base station with OpenBTS, and seamless transition code from the USRP B210 to higher performance, industry-ready USRP platforms.

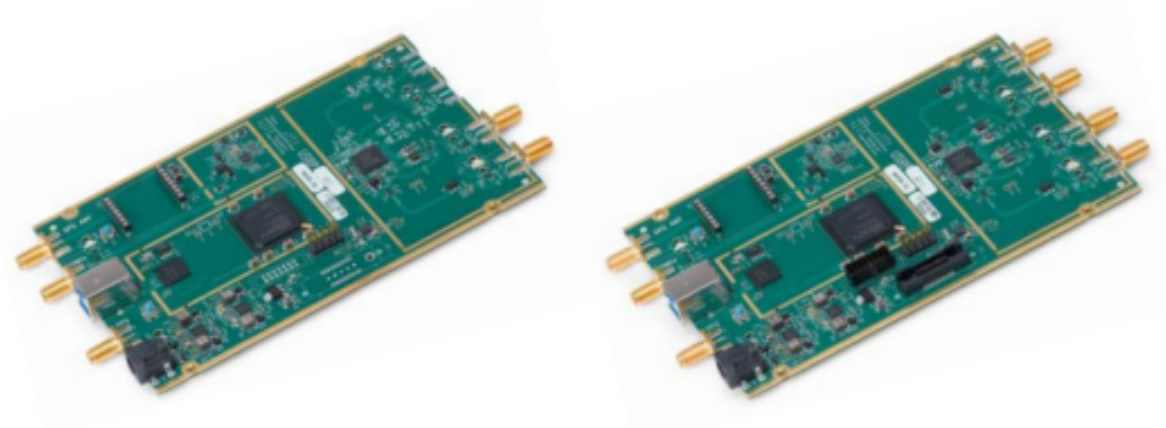


Fig. 3.2 USRP B210

Experimentation with USRP B210

Experiment with the USRP B210 across a wide range of applications including: FM and TV broadcast, cellular, GPS, WiFi, ISM, and more. Users can immediately begin prototyping in GNURadio and participate in the open-source SDR community. Full support by the UHD software allows seamless code reuse from existing designs, compatibility with open-source applications like HDSDR and OpenBTS, and an upgrade path to industry-ready USRP systems to meet application requirements. Here are some examples of what you can do with a USRP B210.

B210 System Architecture

The integrated RF frontend on the USRP B210 is designed with the new Analog Devices AD9361, a single-chip direct-conversion transceiver, capable of streaming up to 56 MHz of real-time RF bandwidth [1]. The B210 uses both signal chains of the AD9361, providing coherent MIMO capability. Onboard signal processing and control of the AD9361 is performed by a Spartan6 XC6SLX150 FPGA connected to a host PC using SuperSpeed USB 3.0. The USRP B210 real time throughput is benchmarked at 61.44MS/s quadrature, providing the full 56 MHz of instantaneous RF bandwidth (maximum) to the host PC for additional processing using GNU Radio or applications that use the UHD API. For detailed throughput capabilities in various SISO and MIMO configurations [8]. The specification of USRP B210 is given in Table 3.1

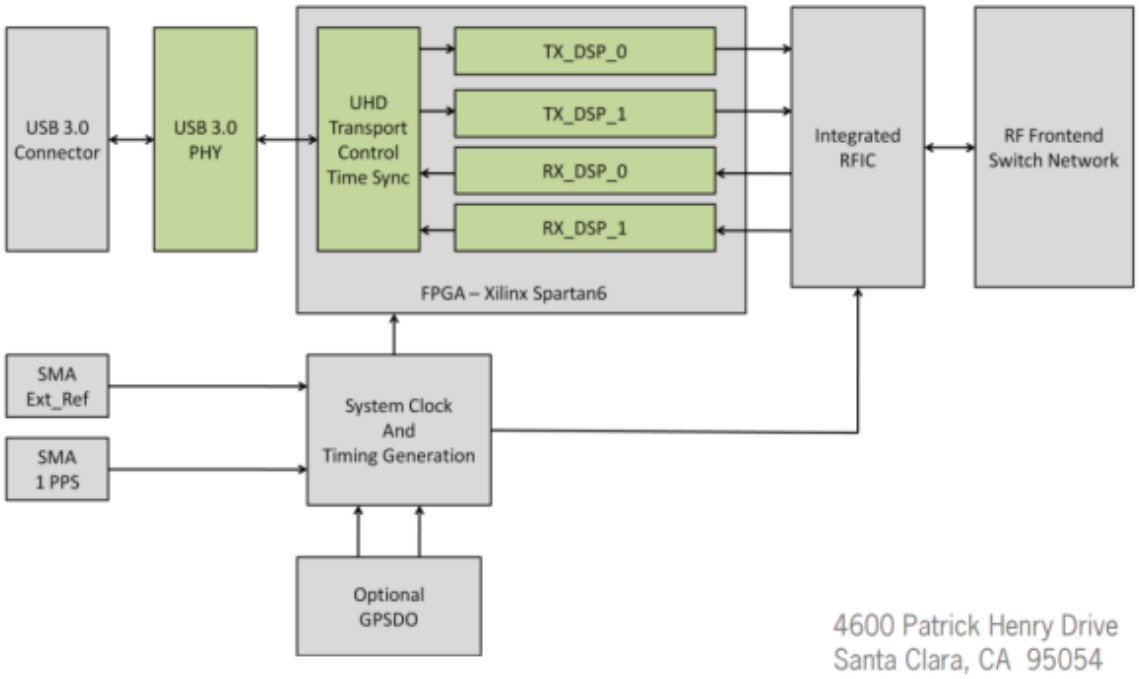


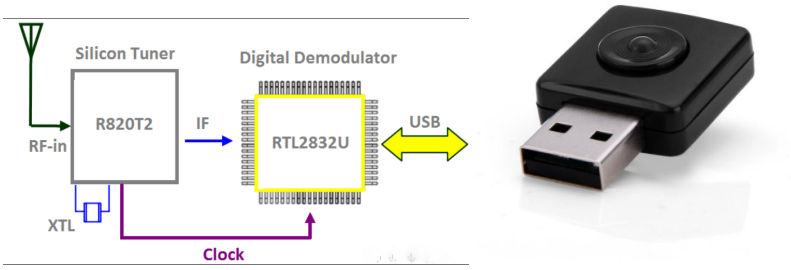
Fig. 3.3 USRP B210 block diagram.

3.4.2 RTL-2832U

The RTL2832U is a high-performance DVB-T COFDM demodulator that supports a USB 2.0 interface. The RTL2832U complies with NorDig Unified 1.0.3, D-Book 5.0, and EN300 744 (ETSI Specification). It supports 2K or 8K mode with 6, 7, and 8MHz bandwidth. Modulation parameters, e.g., code rate, and guard interval, are automatically detected.

Table 3.1 USRP B210 Specifications

S. No.	Parameter	Value	Units
1	DC Input	6	V
2	ADC/DAC Sample Rate (max)	61.44	MS/s
3	ADC/DAC Resolution	12	bits
4	Host Sample Rate	61.44	MS/s
5	Interface	USB 3.0	-
6	FPGA	Xilinx Spartan 6 XC6SLX150 FPGA	-
7	RF-Coverage	70 MHz -6 GHz	-
8	Bandwidth (max)	56	MHz



(a) Block diagram. (b) Receiver.

Fig. 3.4 RTL-2832U.

The RTL2832U supports tuners at IF (Intermediate Frequency, 36.125MHz), low-IF (4.57MHz), or Zero-IF output using a 28.8MHz crystal, and includes FM/DAB/DAB+ Radio Support. Embedded with an advanced ADC (Analog-to-Digital Converter), the RTL2832U features high stability in portable reception. The specification of RTL-2832U is given in Table 3.2

The RTL2832U features Realtek proprietary algorithms, including superior channel estimation, co-channel interface rejection, long echo channel reception, and impulse noise cancellation, and provides an ideal solution for a wide range of applications for PC-TV, such as USB dongle and embedded system via USB interface.

3.4.3 VERT2450 Antenna

In our experiments we used VERT2450 antenna. It is an Omni directional antenna and having gain of 3dBi. It operates on 2.4-2.48 GHz and 4.9-5.9 GHz dual band.

Table 3.2 RTL-2832U Specifications

S. No.	Parameter	Value	Units
1	DC Input	5	V
2	SDR frequency	24 - 1700	MHz
3	DVB frequency	230, 470, 862	MHz
4	SDR Bandwidth	3.2	MHz
5	DVB Bandwidth	6 - 8	MHz
6	Antenna input	50	Ω
7	Carrier type	2K - 8K	-
8	Modulation	QPSK, 16QAM, 64QAM	-

Fig. 3.5 VERT2450 Antenna

Another antenna having frequency range of 24MHz-1700MHz and 3.2MHz bandwidth was also used during experiment.

Chapter 4

Design and Implementation

4.1 Introduction

The aim of this chapter is to describe the DVB standard [5] and its implementation on hardware. In the beginning of the chapter, design and implementation of the reference model of DVB-T transmitter is described. Further, by considering this reference model DVB-T receiver is designed and implemented.

This chapter discusses about the hardware & software setup utilized to implement :

- DVB Transmitter & Receiver
- UDP live video streaming

4.2 Project Scope

The scope of the project is to develop a working prototype of transmitter system of DVB-T standard. The block diagram of this transmitter system is shown in Fig.4.1. To implement this project at first transmitter of DVB-T standard is developed [13] . For receiver part we are using smartphone with RTL-2832U to receive and play live stream.

After successful implementation of DVB-T and DVB-T2 transmitter and receiver system. A video of the working setup is available on the link below:

Setup video link: [14]

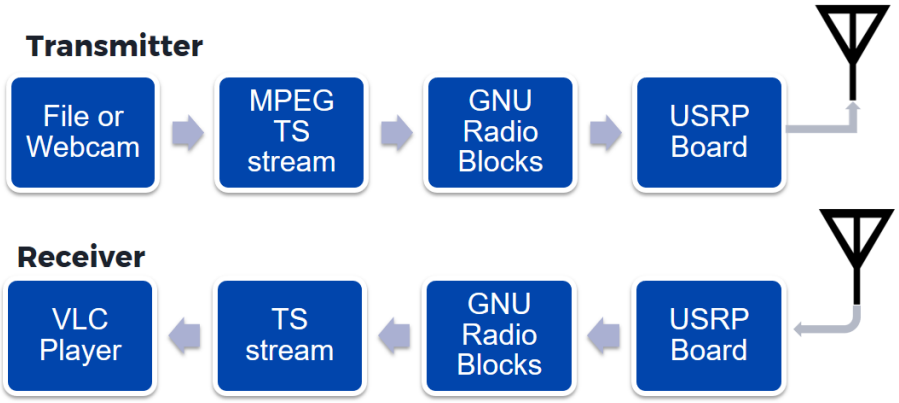


Fig. 4.1 Block diagram of DVB tx and rx.

4.3 System Requirements

In order to implement and design the project , certain software and hardware components required. Software requires to implement is GNU Radio, VLC Media Player, DVB-T driver for android.

Hardware components require are two USRP B210 boards, Vert 2450 Antenna, DVB antenna and RTL-2832U DVB-T stick.

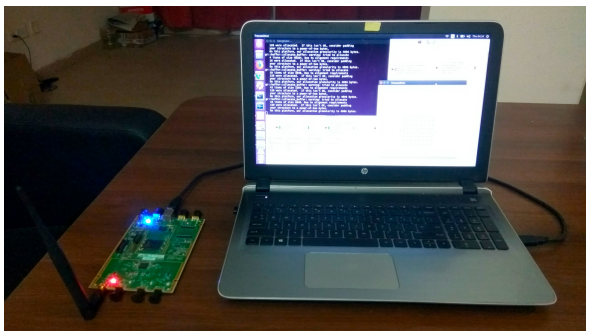
All the specification of these components are described in Chapter 2.

4.4 DVB-T transmitter & receiver

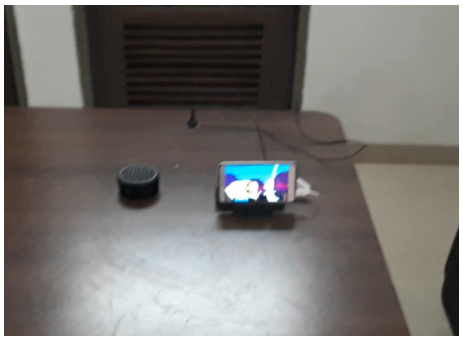
4.4.1 Experimental setup

The experimental setup for real time HD (1080p) video streaming using DVB technology, is shown in Fig. 4.2 (c).
The DVB transmitter setup is shown in Fig.4.2 (a).
The DVB receiver setup is shown in Fig. 4.2 (b).

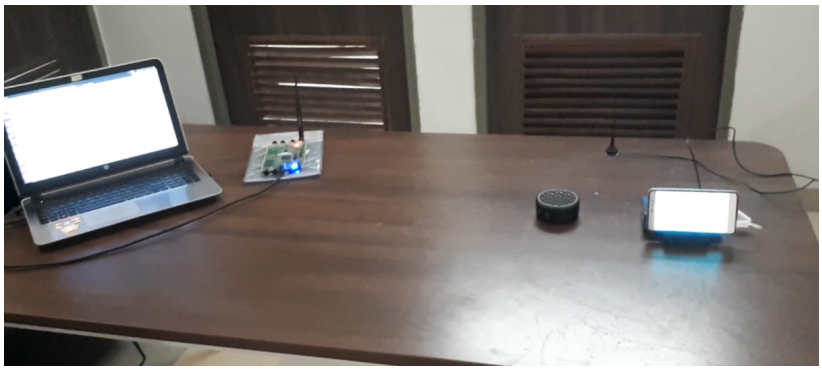
Setup video link: [14]



(a) Transmitter.



(b) Receiver.



(c) Tx & Rx setup.

Fig. 4.2 DVB-T Setup.

4.4.2 Design of DVB-T transmitter

This subsection describes the block of GNU Radio used in DVB-T transmitter, shown in Fig. 4.3.

File source

This block takes MPEG2-TS file of video to be transmitted from PC drive. This could be also be taken from webcam using VLC media player.

Energy Dispersal

In order to disperse the energy in the MPEG2-TS source, this block does a XOR of the actual data with the output of the pseudorandom binary sequence (PRBS) generator. It also creates multiplexer (MUX) frames composed of 8 MUX packets of 188bytes. The sync is done by replacing the 0x47 sync byte with 0xb8 at the beginning of the MUX frame.

Reed-Solomon Encoder

It is an outer forward error correction encoder which reduces the error probability at the receiver by adding redundant bits to the payload.

Convolutional Interleaver

Interleaver is used to improve the forward error correcting codes especially burst error correction.

Inner Coder

It is an inner forward error correction encoder which reduces the error probability at the receiver by adding redundant bits to the payload.

Bit Inner Interleaver

This interleaver is used to improve the performance of inner forward error correcting codes and then maps the bits into symbols.

Symbol Inner Interleaver

This interleaver interleaves the symbols for further improvement of inner forward error correcting codes.

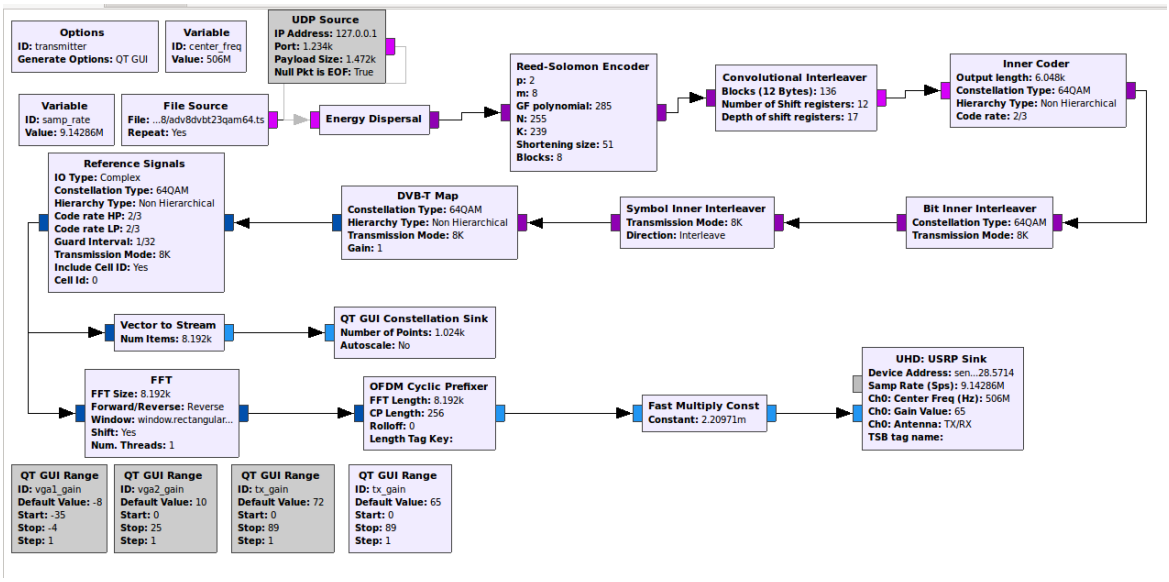


Fig. 4.3 DVB Transmitter GNU Radio Block.

DVB-T Map

This block maps the transmitted symbols according to the DVB-T standards including pilots and null subcarriers.

Reference Signals

In order to synchronization and channel estimation at the receiver, this block append the reference signal to the payload.

FFT

The COFDM (Coded OFDM) modulation used in DVB-T is creating frequency domain bins and the do an inverse fast Fourier transform to convert them to time domain.

OFDM Cyclic Prefixer

In order to mitigate the inter symbol interference, cyclic prefix (CP) is appended to each OFDM symbol , which is the last part of the OFDM symbol. The length of CP must be equal or greater than the length of multipath channel [3].

USRP sink

After modulation, OFDM Cyclic Prefixer gives data to USRP B210 board to transmit that data in air. The parameters, such as antenna gain, operating frequency and bandwidth of USRP board could be adjusted in USRP sink block in GNU Radio.

4.4.3 Design of DVB-T receiver

The working block diagram of DVB receiver is shown in Fig.4.4

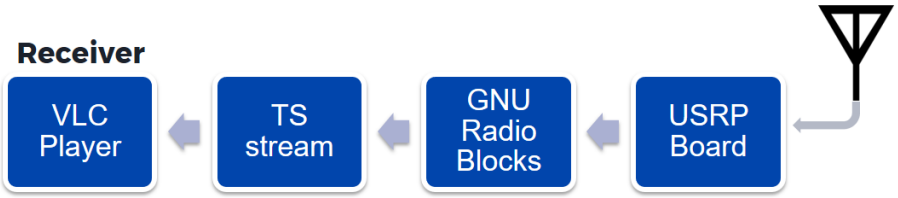
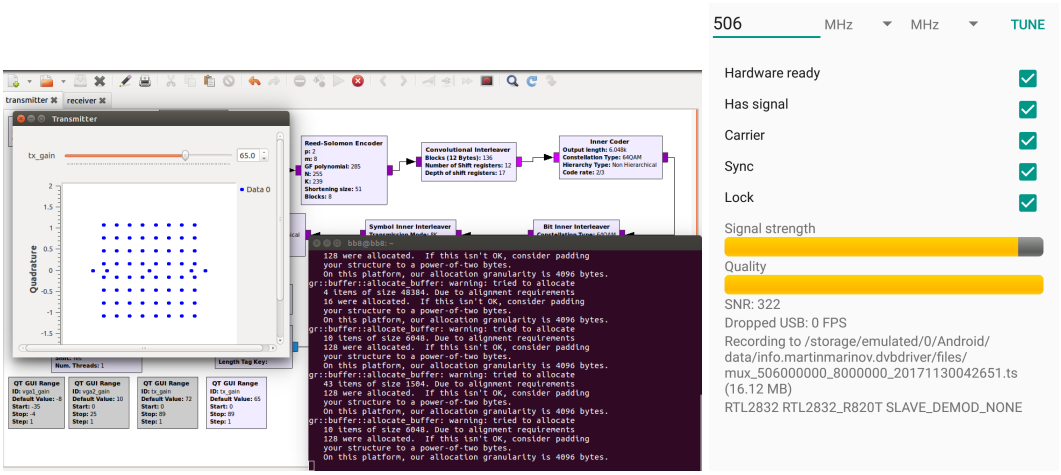


Fig. 4.4 Receiver block diagram.

Transmitted signal is received using antenna attached to USRP B210 board. After after processing and converting received signal from analog to digital. It goes to the GNU radio blocks it performs the reverse operation of the transmitter discussed in section 4.4.2. After converting received signal into MPEG2-TS, this file is saved in temporary file. This file is being played by VLC media player.

This DVB-T receiver could be also implemented using smart phone and RTL-2832U, shown in Fig.3.4. For this, connect RTL-2832U to smart phone using OTG cable. Install DVB-T driver app, this app is available on play store. Set the frequency and receiving mode in app, shown in Fig.4.5 (b). After that start recording received TS file and play received file in real time using VLC media player. The trasmitter and receiver interface is shown in Fig.4.5.



(a) Transmitter. (b) Receiver.

Fig. 4.5 DVB Transmitter & Receiver interface.

4.5 Live Video Streaming Using UDP Source

4.5.1 Experimental setup

The experimental setup of live video streaming using UDP source is explained and shown in Fig.4.6.

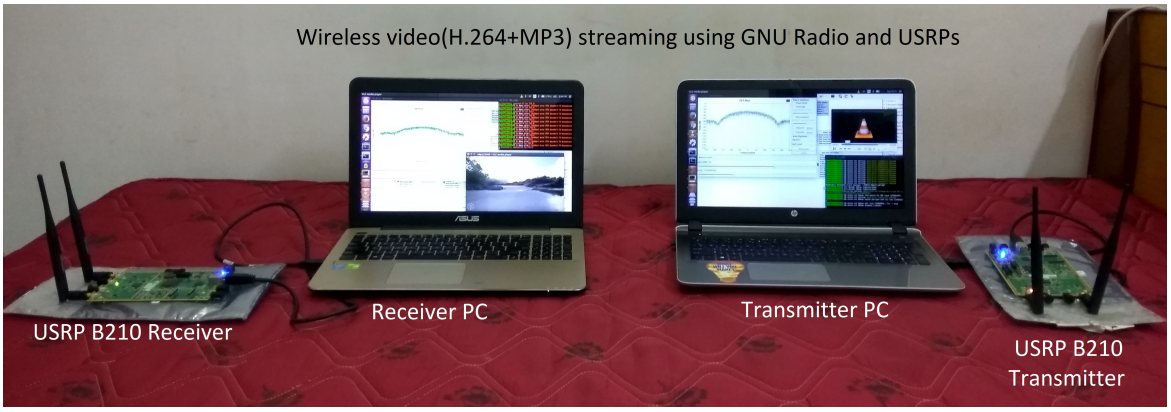


Fig. 4.6 Real time video streaming setup.

4.5.2 Design of live video streamer

Local PC test

In order to transmit a good quality video, It is recommended to perform this local PC test. This test is performed on GNU radio and VLC media player. To create UDP file source follow the following steps:

- Open VLC media player app.
- Type CTRL+C
- In file section add file to be transmitted.
- Select stream option.
- In new destination select UDP and click on add.
- In address type 127.0.0.1 and port 5003
- In transcoding options select Video-H.264+MP3(TS)
- Click next and start the stream.

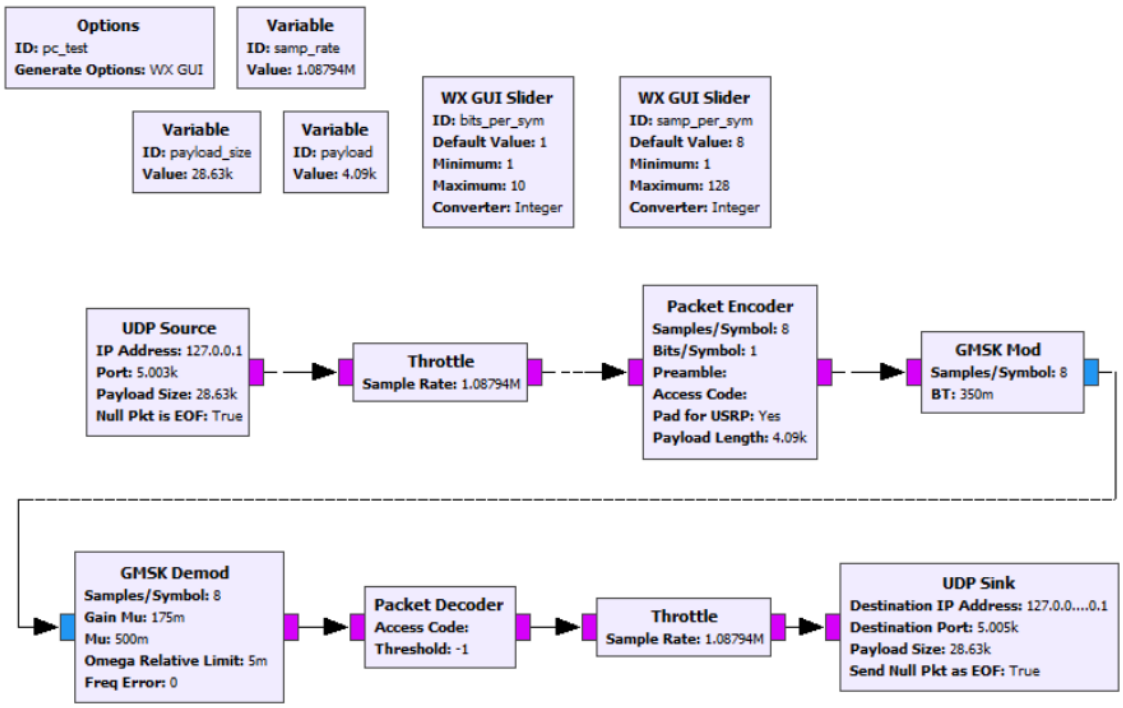


Fig. 4.7 Local PC test (UDP).

Now UDP source is ready.

Fig.4.7 shows the GNU Radio blocks used in local PC test. UDP source is connected to Throttle for controlling the input UDP source. Packet takes in 4009 bytes at a time, and these bytes get wrapped into a packet with a header, access code, and preamble, and the packet is sent on to the next block. Gaussian Minimum Shift Keying (GMSK) modulation is performed by GMSK mod block. After that Demodulation is done by GMSK demod block and demodulated data is sent to Packet decoder, It performs the reverse operation of packet encoder. Then decoded data is passed through Throttle block to UDP sink.

To play data sent to UDP sink, Open VLC media player on another tab type CTRL+N and enter the udp sink url: udp://@5005 . It plays the media present at UDP sink at local host port:5005.

This PC test is done on PC to check the modulation, demodulation, sampling rate and payload size required for the selected media file. After successful test we can design transmitter and receiver for streaming media file. It is shown in Fig.4.8.

Transmiter

The working of transmitter is shown in Fig.4.8. File source is converted into UDP stream using VLC media player. GNU Radio transmitter block receives this stream on localhost:5003. After performing packet encoding and GMSK modulation, data is sent to USRP B210 for RF transmission.

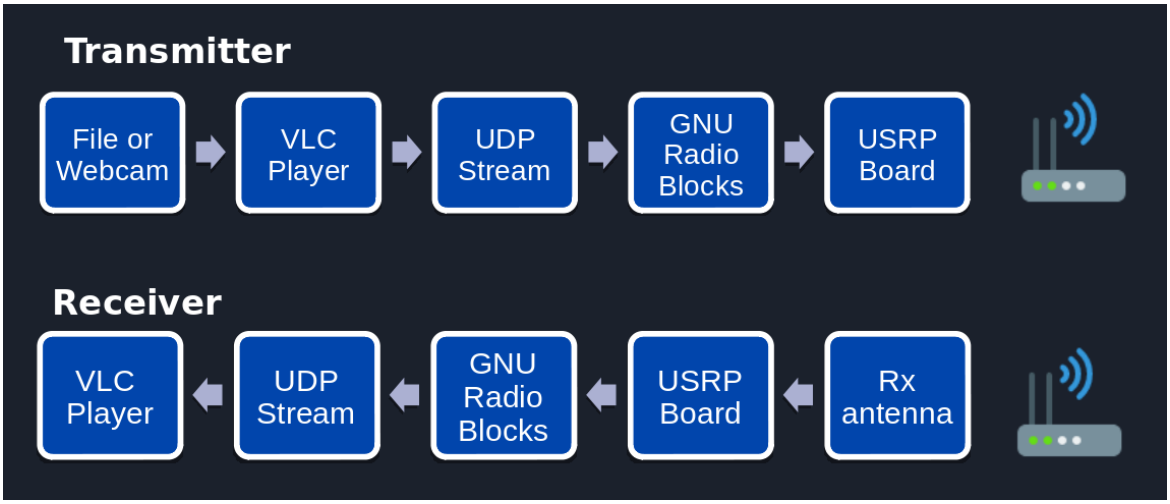


Fig. 4.8 Block diagram

In local PC test, shown in Fig.4.7, Adding USRP sink block after GMSK modulation. It gives the transmitter block diagram shown in Fig.4.9.

Receiver

The working of receiver is shown in Fig.4.8. USRP source data is sent to low pass filter for removing unwanted noise from air. The output of low pass filter is sent to GSMSK demod for demodulation. The demodulated data is sent to packet decoder, It performs the reverse operation of packet encoder discussed in local PC test section. After that decoded data is sent to UDP sink at localhost:5005. VLC media player receives stream at localhost:5005 (udp://@5005).

In local PC test, shown in Fig.4.7, Adding USRP source and low pass filter block before GMSK demod. It gives the receiver block diagram shown in Fig.4.10.

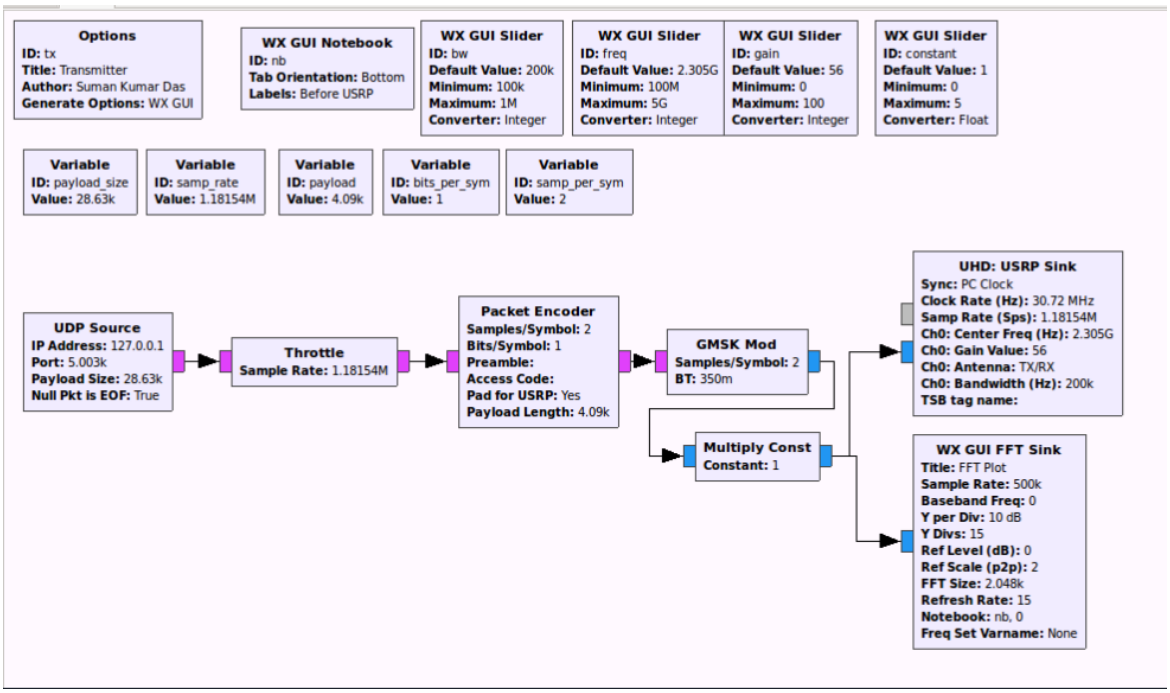


Fig. 4.9 GNU Radio Tx block (UDP).

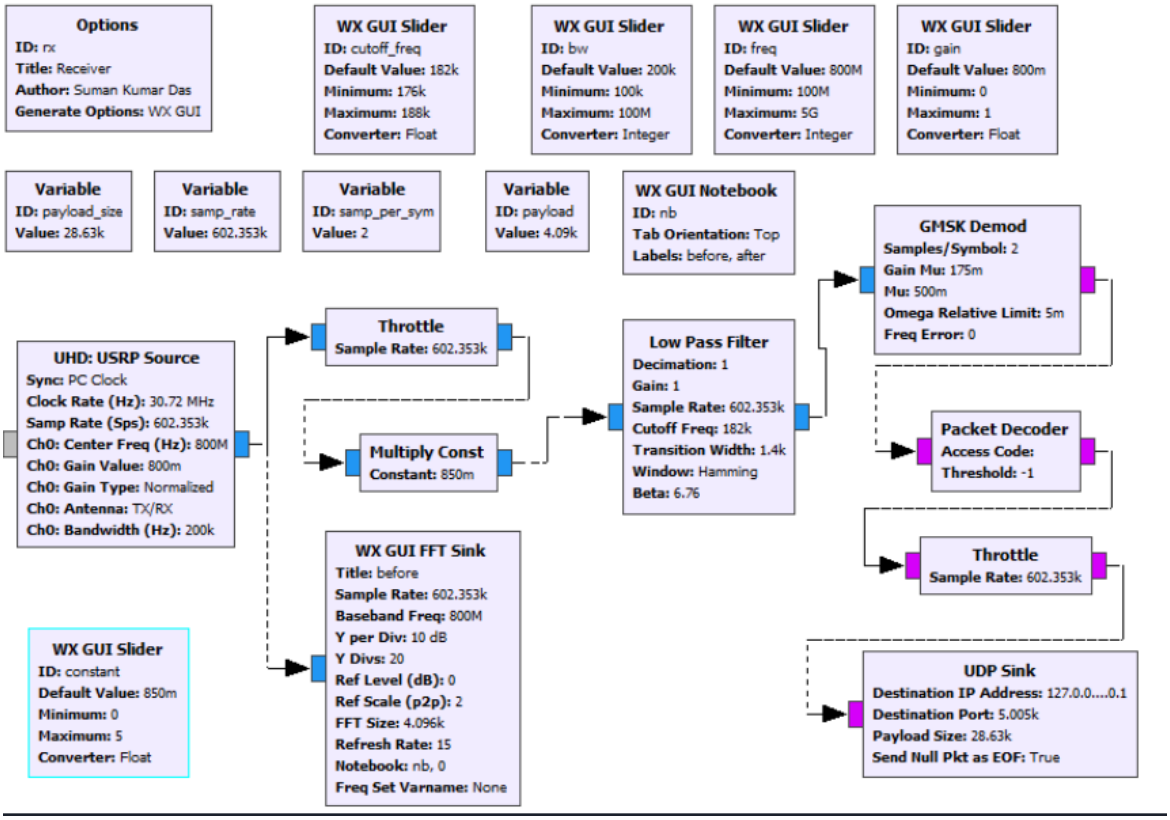
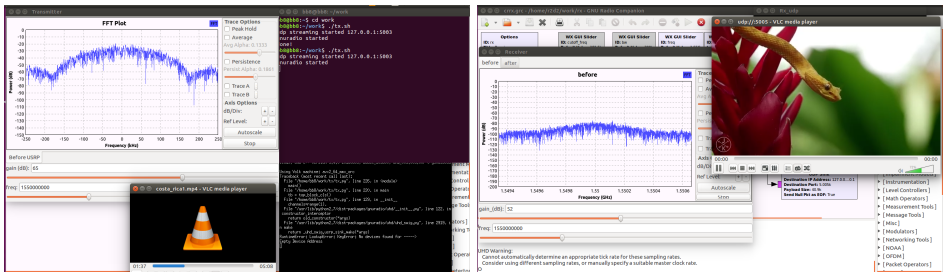


Fig. 4.10 GNU Radio Rx block (UDP).

The working transmitter and receiver is shown in Fig.4.11.



(a) Transmitter. (b) Receiver.

Fig. 4.11 Transmitter & Receiver (UDP).

Chapter 5

Result, Conclusion and Future Work

5.1 Result

After implementation of SDR with USRPs, following results were obtained from the experiments:

- Using USRP and GNU Radio, successful DVB-T transmission is being implemented.
- The resolution of transmitted video was 1920x1080 with frame rate 30 fps.
- Zero frame drop in DVB achieved (keeping transmitter and receiver stationary).
- Receiver is able to receive good signal strength and quality upto two concrete walls.

5.2 Conclusion

This setup can be used in local video broadcasting after amplifying its signal strength and installing outdoor antenna.

After installing outdoor antenna and increasing Tx power, local area DVB broadcast can be done.

Frequency, bandwidth and antenna gain of USRP can be software controlled in real time. So it could be used in advanced RF communication research work.

The vision of SDR is implementing a single radio device containing software controlled digital hardware that can emulate any radio signal of evolving or already existing wireless standards simply by updating software without replacing the underlying hardware platform. The growth of SDR technology depends on the development of the performance of its participating components such as continuous enhancement of the resolution and sampling rate of

signal converters ADCs/DACs, the better performance of filters, amplifiers and improvement of processors in terms of their functionality, performance, speed, power consumption etc. Although SDR technology faces several hurdles, many positive approaches are being introduced to overcome them. These make it suitable to contribute in the development of future generations of wireless communication technology by resolving the limitations of traditional hardware structure.

5.3 Future Work

There are many features that can be added to this transceiver model. These features are:

- DVB plays a very special role in global media broadcasting. So recent DVB standards e.g. DVB-S2X, DVB-SI, DVB-IPTV etc. can be implemented using SDR.
- Right now its a static transmitter and receiver, this can be upgraded to moving transmitter and receiver.
- Multichannel broadcast with MIMO support in SDR.
- SDR frequency, gain and bandwitdh can be changed in real time, These special features could be used in 5G and advanced RF research work.

References

[1] Ettus Research. B210 Specifications. https://www.ettus.com/content/files/kb/b200-b210_spec_sheet.pdf, 2015. [Online].

[2] Wikipedia. SDR. https://en.wikipedia.org/wiki/Software-defined_radio, 2017. [Online].

[3] A. Bishnu and V. Bhatia. An IEEE 802.22 Transceiver Framework and its Performance Analysis on Software Defined Radio for TVWS. Submitted to Telecommunication System, Springer, 2017.

[4] GNU Radio. . https://www.gnuradio.org/, 2017. [Online].

[5] ETSI. DVB-T2. http://www.etsi.org/deliver/etsi_en/302700_302799/302755/01.03.01_60/en_302755v010301p.pdf, 2012. [Online].

[6] C. Kocks, A. Viessmann, A. Waadt, C. Spiegel, A. Burnic, G. H. Bruck, P. Jung, Jaeyoel Kim, YeonJu Lim, and Hyeon Woo Lee. A dvb-t2 receiver realization based on a software-defined radio concept. In *2010 4th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 1–4, March 2010.

[7] Ron Economos. Github source. https://github.com/drmpeg/gr-dvbt2, 2016. [Online].

[8] Ettus Research. USRP B210. https://www.ettus.com/product/details/UB210-KIT, 2017. [Online].

[9] Dvb.org. Second Generation Terrestrial DVB. https://www.dvb.org/resources/public/factsheets/dvb-t2_factsheet.pdf, 2016. [Online].

[10] L. Polak and T. Kratochvil. Measurement and evaluation of iq-imbalances in dvb-t and dvb-t2-lite ofdm modulators. In *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, pages 555–558, July 2017.

[11] Drakshayini M N and A. V. Singh. A review on reconfigurable orthogonal frequency division multiplexing (ofdm) system for wireless communication. In *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pages 81–84, July 2016.

[12] NI. LabVIEW. https://en.wikipedia.org/wiki/LabVIEW, 2014. [Online].

[13] F. Moumtadi and Á. L. Lobaina. Developing of a practical teaching method of digital terrestrial television based on software. In *2015 International Conference on Computing Systems and Telematics (ICCSAT)*, pages 1–6, Oct 2015.

[14] Setup video. DVB-T setup. https://youtu.be/UzHbrAK-78o, 2017. [Online].

Appendix A

Building and Installing the UHD and GNU Radio

This Application Note provides a comprehensive guide for building, installing, and maintaining the open-source toolchain for the USRP (UHD and GNU Radio) from source code on the Linux platform.

Install Linux

Download and install Ubuntu from the link below. Download the appropriate ISO image, and write it to a USB flash drive. Be sure to verify that the ISO file was not corrupted during the download process by checking the MD5 and/or SHA1 hash.

Ubuntu download page
Create bootable USB drive

Be sure to use a USB flash drive with at least 8 GB capacity, and use a USB 3.0 flash drive, not a USB 2.0 flash drive. If you use a slower USB 2.0 flash drive, then the install process will take significantly longer.

Update and Install dependencies

Before building UHD and GNU Radio, make sure that all the dependencies are updated

```
sudo apt-get update
```

To make sure that all the dependencies are installed run following command:

sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.13-0v5 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk3.0 git-core libqt4-dev python-numpy ccache python-opengl libgsl-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq-dev libzmq1 python-requests python-sphinx libcomedi-dev python-zmq

After installing the dependencies, you should reboot the system. If the installation of the dependencies completes without any errors, then you can proceed to build and install UHD and GNU Radio.

Building and installing UHD from source code

UHD is open-source, and is hosted on GitHub.There are several good reasons to build GNU Radio from source code, especially for doing development and prototyping. It it enables an easy way to customize the location of the installation, and to install multiple UHD versions in parallel, and switch between them. It also provides much more flexibility in upgrading and downgrading versions, and allows the user to modify the code and create customized versions, which could possibly include a patch or other bug-fix.

To build UHD from source code, clone the GitHub repository, check out a branch or tagged release of the repository, and build and install. Please follow the steps below. Make sure that no USRP device is connected to the system at this point.
First, make a folder to hold the repository.

```
cd $HOME
mkdir workarea-uhd
cd workarea-uhd
```

Next, clone the repository and change into the cloned directory.

```
git clone https://github.com/EttusResearch/uhd
cd uhd
```

Next, checkout the desired UHD version. You can get a full listing of tagged releases by running the command:

```
git tag -l
```

Example: For UHD 3.9.5:

```
git checkout release_003_009_005
```

Next, create a build folder within the repository.

```
cd host
mkdir build
cd build
```

Next, invoke CMake to create the Makefiles.

```
cmake ../
```

Next, run Make to build UHD.

```
make
```

Next, you can optionally run some basic tests to verify that the build process completed properly.

```
make test
```

Next, install UHD, using the default install prefix, which will install UHD under the /usr/local/lib folder. You need to run this as root due to the permissions on that folder.

```
sudo make install
```

Next, update the system's shared library cache.

```
sudo ldconfig
```

Finally, make sure that the environment variable is defined and includes the folder under which UHD was installed.

```
export LD_LIBRARY_PATH=/usr/local/lib
```

At this point, UHD should be installed and ready to use.

Building and installing GNU Radio from source code

As with UHD, there are several good reasons to build GNU Radio from source code, especially for doing development and prototyping. It it enables an easy way to customize the location of the installation, and to install multiple GNU Radio versions in parallel, and switch between them. It also provides much more flexibility in upgrading and downgrading versions, and allows the user to modify the code and create customized versions, which could possibly include a patch or other bug-fix.

Similar to the process for UHD, to build GNU Radio from source code, clone the GitHub repository, check out a branch or tagged release of the repository, and build and install. Please follow the steps below. Make sure that no USRP device is connected to the system at this point.
First, make a folder to hold the repository.

```
cd $HOME
mkdir workarea-gnuradio
cd workarea-gnuradio
```

Next, clone the repository and change into the cloned directory.

```
git clone --recursive https://github.com/gnuradio/gnuradio
```

Next, go into the repository and check out the desired GNU Radio version.

```
cd gnuradio
git checkout v3.7.10.1
```

Next, create a build folder within the repository.

```
mkdir build
cd build
```

Next, invoke CMake to create the Makefiles.

```
cmake ../
```

Next, run Make to build GNU Radio.

```
make
```

Next, you can optionally run some basic tests to verify that the build process completed properly.

```
make test
```

Next, install GNU Radio, using the default install prefix, which will install GNU Radio under the /usr/local/lib folder. You need to run this as root due to the permissions on that folder.

```
sudo make install
```

Finally, update the system's shared library cache.

```
sudo ldconfig
```

At this point, GNU Radio should be installed and ready to use. You can try launching the GNU Radio Companion (GRC) tool, a visual tool for building and running GNU Radio flowgraphs.

```
gnuradio-companion
```

Configuring USB

On Linux, udev handles USB plug and unplug events. The following commands install a udev rule so that non-root users may access the device. This step is only necessary for devices that use USB to connect to the host computer, such as the B200, B210, and B200mini. This setting should take effect immediately and does not require a reboot or logout/login. Be sure that no USRP device is connected via USB when running these commands.

```
cd $HOME/workarea-uhd/uhd/host/utils
sudo cp uhd-usrp.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
```

Connect the USRP

The installation of UHD and GNU Radio should now be complete. At this point, connect the USRP to the host computer.

```
$ uhd_find_devices
$ uhd_usrp_probe
```

Appendix B

Building and Installing Using linux script

This method uses linux script to install all tools including latest the USRP Hardware Driver™ (UHD) and GNU Radio. It saves alot of time during environment setup.

Update and Install dependencies

Before building UHD and GNU Radio, It is necessary to make sure that all the dependencies are first installed. For this we need to create a script file which will update and install dependencies.
create a file using terminal:

```
sudo nano beforeboot.sh
```

After creating file paste following:

```
#!/bin/bash
sudo apt-get update
```

sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.13-0v5 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++

automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk3.0 git-core libqt4-dev python-numpy ccache python-opengl libgsl-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq-dev libzmq1 python-requests python-sphinx libcomedi-dev python-zmq

```
echo "Please reboot and run next afterboot.sh"
echo " "
echo " Reboot Now ?"
echo " Enter 1 for Yes  "
echo " Enter 2 for No  "
echo " "
select yn in "Yes" "No"; do
    case $yn in
        Yes ) sudo reboot break;;
        No ) exit;;
    esac
done
```

Save this file and to make this file executable run:

```
sudo chmod +x beforeboot.sh
```

Run this script.

```
sudo ./beforeboot.sh
```

It will ask to reboot after installing dependencies. Enter 1 to reboot.

Install UHD and GNU Radio

For installing UHD and GNU Radio, create another script "afterboot.sh"using terminal:

```
sudo nano afterboot.sh
```

After creating file paste following:

```
#!/bin/bash
sudo apt-get update
#Building and installing UHD from source code
mkdir workarea-uhd
cd workarea-uhd
git clone https://github.com/EttusResearch/uhd
cd uhd
git tag -l
echo "Example: For UHD 3.9.5:"
echo "write release_003_009_005"
echo -e "Please enter release version : "
read version
git checkout $version
cd host
mkdir build
cd build
cmake ../
make
make test
sudo make install
sudo ldconfig
export LD_LIBRARY_PATH=/usr/local/lib
sudo apt-get update
sudo apt-get install gnuradio
echo "Done"
```

Save this file and to make this file executable run:

```
sudo chmod +x afterboot.sh
```

Run this script.

```
sudo ./afterboot.sh
```

Adjust shared memory by using following command:

```
sudo sysctl -w kernel.shmmax=2147483648
```