NIPES

Journal of Science and Technology Research

Journal homepage: www.nipesjournals.org.ng

Design and Implementation of a Digital Video Broadcasting System Using Software-Defined Radio

***Bello N.**[a,*] **and Obasohan O.**[b]*

[a, b] Department of Electrical/Electronic Engineering, University of Benin, PMB 1154, Benin City, Edo State, Nigeria.
Corresponding Author Email: nosabello@uniben.edu, osamuwa.obasohan@uniben.edu

Article Info

***Keywords*:**
Digital Video Broadcasting, VLC, Software-defined radio, RTL-SDR, AVCONV, USRP B210, DVB-T

Received 23 October 2021
Revised 22 December 2021
Accepted 31 December 2021
Available online 8 December 2022

https://doi.org/*10.5281/zenodo.7415334*
https://nipesjournals.org.ng

Abstract

In this paper, we present the design and implementation of a digital video broadcasting system using software-defined radio. The system aimed to broadcast a live video feed wirelessly from one computer to another. The design made use of a USRP B210 for transmission utilizing GNU radio companion blocks imported into a python script designed to transmit the live video feed and an RTL SDR dongle for the reception which is done using a VLC player. The developed system achieved a 2K mode DVB-T transmission on a 6.587 MHz bandwidth across a maximum range of 20.4 meters. The maximum range was dependent on the power gain of the transmitter, which is the frontend capability of the software-defined radio. Therefore, this makes improvement and future-proofing easy.

1.0. Introduction

Broadcasting is the distribution of audio and/or video utilizing the electromagnetic spectrum via any electronic mass communications medium to relay these messages to a mass/dispersed audience [1]. Broadcasting began with audio broadcasting, which came into popular use around 1920 with the spread of vacuum tube radio transmitters and receivers.

Analogue broadcasting over the years had some shortcomings such as low image quality, inefficiency in spectrum usage and high power for transmission [2]. This necessitated the need for the use of digital broadcasting/communication which offers a better quality of picture and sound, consumes less bandwidth than the analogue, is more efficient when it comes to bandwidth usage than analogue transmission and most importantly less immune to noise. Ever since the advent of software-defined radios, researchers have attempted at implementing broadband wireless systems using these commercial off-the-shelf platforms [3]. This is duly to the affordability of the camp equipment for emergency broadcast; future-proofing with ETSI DVB-T standard evolution; support for high-complexity cell-based content-distribution networks; and potential for using DVB-T as the transport layer technology for Internet protocol (IP)-based networks [4] in rural communities to address digital-divide impairment. In [3], a real-time DVB system was successfully designed using USRP2 software-defined radio which was a development with relatively low complexity, such that it was likely to be executed within the timing constraints even on a non-real-time platform. However, USRP2 was among the first generation of software-defined radios developed by Ettus Research which has evolved greatly. Similarly, [4] designed a software-based version of an ESTI DVB-T transmitter using GNU Radio to achieve an acceptable computational resource requirement.

In this paper, we present the design and implementation of a complete DVB-T system using USRP B210 and RTL-SDR dongle software-defined radios.

1.1 The Standard ETSI DVB-T

DVB-T or Digital Video Broadcasting Terrestrial is the most widely used digital television standard in use around the globe for terrestrial television transmissions. It provides many facilities and enables far more efficient use of the available radio frequency spectrum than the previous analogue transmissions. The DVB-T system addresses the terrestrial broadcasting of MPEG-2 coded TV signals. Therefore, an appropriate adaptation of the digital coded transport stream to the different terrestrial channel characteristics requires the use of a multi-carrier modulation, the so-called Orthogonal Frequency Division Multiplex (OFDM) technique, combined with a powerful concatenated error correction coding (Coded Orthogonal Frequency Division Multiplex, COFDM) [5, 6].

The structure of the transmitter device that adapts the baseband motion picture expert group-2 (MPEG2) signal to the typical multipath radio frequency channel, briefly described in Figure 1, allows the distribution over small and large Single Frequency Networks (SFN).

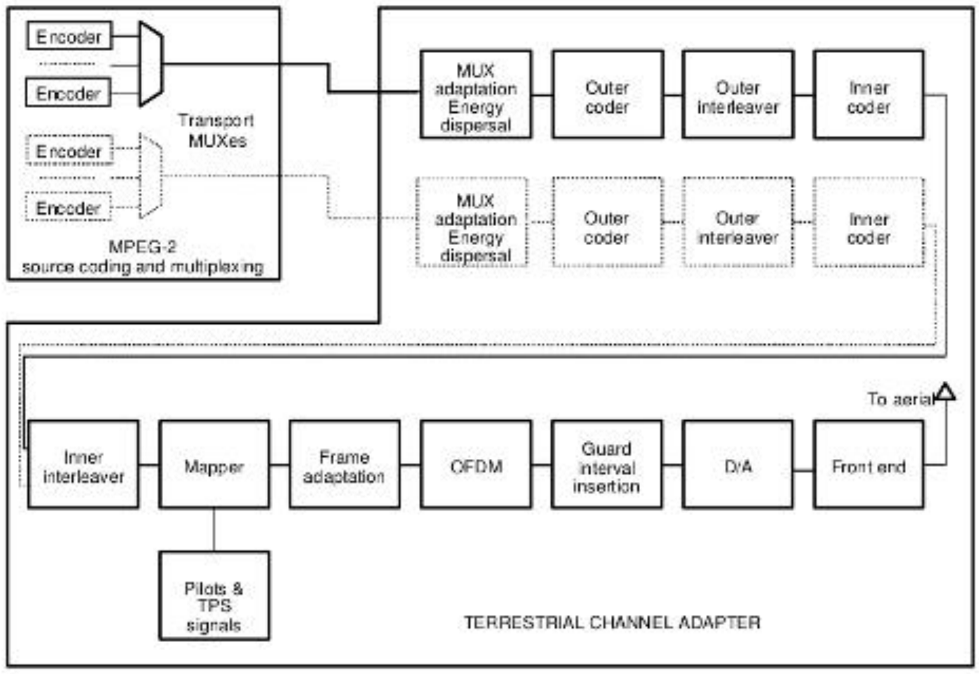


Figure 1 Functional Block Diagram of the system [7, 8]

The ETSI DVB-T system is defined as the functional block of equipment performing the adaptation of the baseband TV signals from the output of the MPEG-2 transport multiplexer, to the terrestrial channel characteristics. The following process shall be applied to the data stream:

i) Transport multiplex adaptation and randomization for energy dispersal (MAED) which removes time domain correlation from byte-wise MPEG2 input stream by performing a bit-level XOR with a pseudo-random binary sequence (PRBS);
ii) Outer coding which performs Reed-Solomon (RS) encoding at the byte level and in systematic form. The chosen code is an RS (204,188), obtained by shortening an RS (255,239). The main purpose of the RS code is to mitigate the impact of the error bursts produced at the receiver side by the Viterbi algorithm;
iii) Outer interleaving (i.e., punctured convolutional code) which aims to minimize correlation between the residual errors at decoding time;
iv) Inner coding which performs convolutional encoding and puncturing to achieve quasi-error-free (QEF) transmission;
v) Inner interleaver which provides support for hierarchical modulation and fairness in assigning the payload bits to the OFDM carriers, i.e., it prevents certain bits from the original TS from being constantly assigned to the same set of OFDM carriers with an unsatisfactory signal-to-noise ratio (SNR);
vi) Mapping which modulates the encoded bits onto QPSK, 16-QAM, and 64-QAM constellations. Both non-hierarchical and hierarchical modulations modes are supported, thus allowing two different TSs with different error performance to be transmitted simultaneously as seen by the two entirely separate TS in Figure 1 showing the high- (thick lines) and low-priority streams (thin lines);
vii) Frame adaptation (FA) which provides insertion of all needed reference signals: pilot carriers, used for synchronization and channel estimation, and carriers conveying information given the implemented transmission parameters. This operation is called transmission parameter signalling (TPS);
viii) Orthogonal Frequency Division Multiplexing (OFDM) modulation block adds virtual carriers and performs an inverse fast Fourier transform (IFFT) with 2048 (2k) or 8192 (8k) subcarriers;

1.2 Universal Software Radio Peripheral (USRP)

Universal Software Radio Peripheral (USRP) is a range of software-defined radios designed and sold by Ettus Research. The USRP product family is intended to be a comparatively inexpensive hardware platform for software radio and is commonly used by research labs, universities, and hobbyists. Most users connect to a host computer through a high-speed link, which the host-based software uses to control the USRP hardware and transmit/receive data. Some USRP models also integrate the general functionality of a host computer with an embedded processor that allows the USRP device to operate in a stand-alone fashion.

The USRP product family includes a variety of models that use a similar architecture. A motherboard provides the following subsystems: clock generation and synchronization, FPGA, ADCs, DACs, host processor interface, and power regulation. A modular front-end called a daughterboard, is used for analogue operations such as up/down-conversion, filtering, and other signal conditioning. This modularity permits the USRP to serve applications that operate between DC and 6 GHz.

1.3 USRP Products

The USRP family has many products and they include:

i) USRP N series (Networked series)
ii) USRP E series (Embedded series)
iii) USRP X series

iv) USRP B series (BUS series)

1.3.1 Bus series

All products in Ettus Research Bus Series use a USB 2.0 or USB 3.0 interface to transfer samples to and from the host computer. Unlike the other USRP Product Series, the Bus series doesn’t make use of daughterboards. These are designed for applications that do not require the higher bandwidth and dynamic range provided by the Network Series. This series consists of USRP B100, USRP B200, USRP B210 and USRP1 [9].

1.4 RTL-SDR

The RTL-SDR is a small USB dongle used to receive radio signals. RTL-SDR is a very cheap software defined radio peripheral that uses the RTL2832U chipset.

The RTL2832U chipset is a high-performance DVB-T COFDM demodulator that supports a USB 2.0 interface. It supports 2K or 8K mode with 6MHz, 7MHz and 8MHz bandwidth. Modulation parameters, e.g., code rate, and guard interval are automatically detected [10].

The RTL2832U outputs 8-bit I/Q-samples, and the highest theoretically possible sample rate is 3.2Megasamples per second (MS/s), however, the highest sample rate without lost samples that have been tested so far is 2.56MS/s. The frequency range is highly dependent on the tuner that is used. The different tuners used are shown in Table 1 [11].

Table 1 Different tuners used in RTL-SDR dongles

TUNER	FREQUENCY RANGE
Elonics E4000	52-2200MHz with a gap from 1100MHz to 1250MHz (varies)
Rafael Micro R820T	24-1766MHz
Rafael Micro R828D	24-1766MHz
Fitipower FC0013	22-1100MHz
Fitipower FC0012	22-948.6MHz
FCI FC2580	146-308MHz and 438-924MHz (gap in between)

There are so many RTL-SDR products sold such as Silver Stick from RTL-SDR.com R820T2, Terratec R820T, Black Nooelec R820T, Blue Nooelec R820T2, and a Nooelec Nano R820T.

2.0. Material and Methods

The implementation for the digital video broadcasting was done in stages. Before the start of the various processes, electronic devices and software were made available. They include

i) LINUX UBUNTU 16.04 operating system
ii) GNU Radio
iii) AVCONV
iv) USRP B210
v) RTL SDR
vi) Two laptops
vii) Two Antennas
viii) Flash drive

In the implementation of digital video broadcasting, we had two design phases; the transmission of the live feed using USRP B210 and the reception of the live feed using RTL SDR. At the transmitter end, we used the USRP B210 device. The USRP B210 is a circuit board used to transmit and receive radio signals. The integrated RF frontend on the USRP B210 is designed with the Analogue Device AD9361, a single-chip direct-conversion transceiver, capable of streaming up to 56 MHz of real-time RF bandwidth. The B210 uses both signal chains of the AD9361, providing coherent MIMO capability. On-board signal processing and control of the AD9361 is performed by a Spartan 6 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 to the host PC for additional processing using GNU Radio or applications that use the UHD API [9].

We used the NOOELEC NESDR SMART to dongle out the available options of RTL-SDRs. NOOELEC NESDR SMART is the latest RTL-SDR dongle created by NOOELEC with improved features and the best accessory set ever offered for radio enthusiasts.

Figure 2 NOOELEC NESDR SMART

NOOELEC NESDR SMART has an RTL2832U Demodulator/USB interface IC and an R820T2 tuner IC. It supports 0.5PPM, ultra-low phase noise TCXO and has female SMA antenna input port. Its uses through-hole direct sampling pads on PCB [10].

Table 2. Comparison between RTL-SDR and USRP B210

Specs	**NOOELEC NESDR SMART**	**USRP B210**
Tune Low (MHz)	24	70
Tune Max (MHz)	1766	6000
Max. Instantaneous Bandwidth (MHz)	3.2	56
ADC Resolution (Bits)	8	12
DAC Resolution (Bits)	-	12
Pre selectors	Uses tracking RF filters on the R820T2 chip	None

The SDR software is the abstraction of signal processing in a host computer rather than in the DSP and FPGA in the SDR hardware. There is so many SDR software in use and some of them include the SDR#, SDR-Radio, HDSDR, Modified Versions of PowerSDR (NaP3 for example), SpectraVue (RF Space), Rocky (Great for Softrock), GNU Radio (Linux), MATLAB SIMULINK and so on. We used GNU Radio due to its popularity in SDR research. GNU Radio is a free and open-source cross-platform software development kit that provides signal processing blocks to implement SDRs. It can be used with readily-available low-cost RF hardware to create software-defined radios, or without hardware in a simulation-like environment [12].

2.1. Design Stages

The entire DVB-T system having the transmitter and receiver stages are shown in Figures 3 and 4.

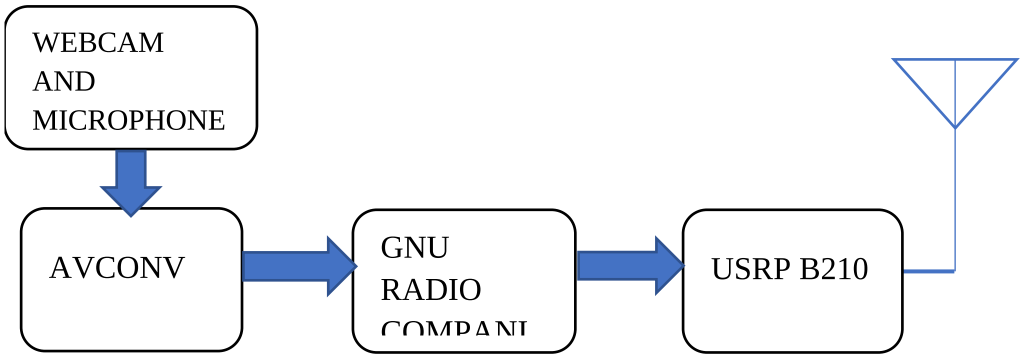


Figure 3 Transmitter stage

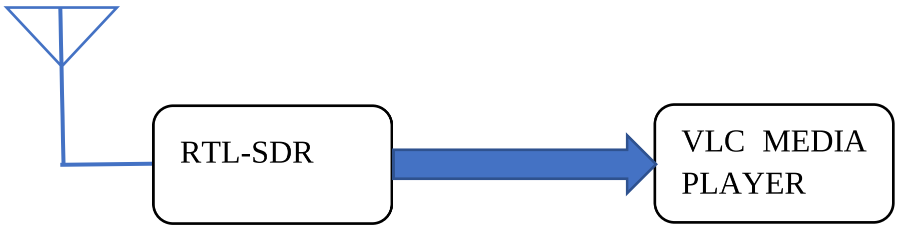


Figure 4. Receiver stage

The different stages that were involved in the implementation of the DVB-T transmitter stage using software-defined radio are as follows. The stages are presented in Figure 5.

i) Acquisition of the Input Signal
ii) Video Filtering
iii) Encoding the Input Signal
iv) MPEG 2 TS Multiplexing
v) Modulating Stage

vi) Transmitting Stage

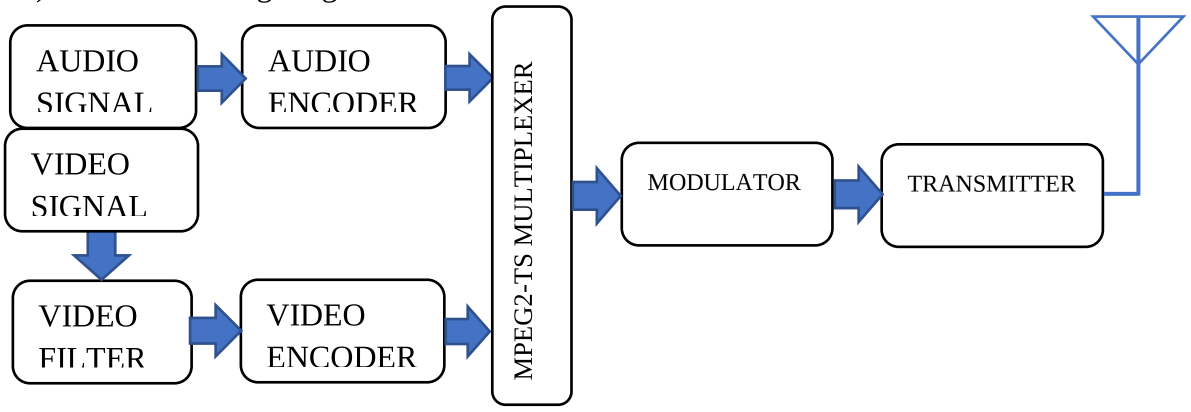


Figure 5. Transmitter design stages of DVB-T

2.2. Acquisition of the Input Signal

The input signal is made up of video and audio signals. The video signal to be transmitted is gotten from the webcam of the computer, and this video signal is made as an input to AVCONV by entering the video4linux2 command on the terminal window. The audio signal to be transmitted is gotten from the computer's microphone, and this audio signal is made as an input to AVCONV by entering the ALSA (Advanced Linux Sound Architecture) command on the terminal window.

2.3. Video Filtering

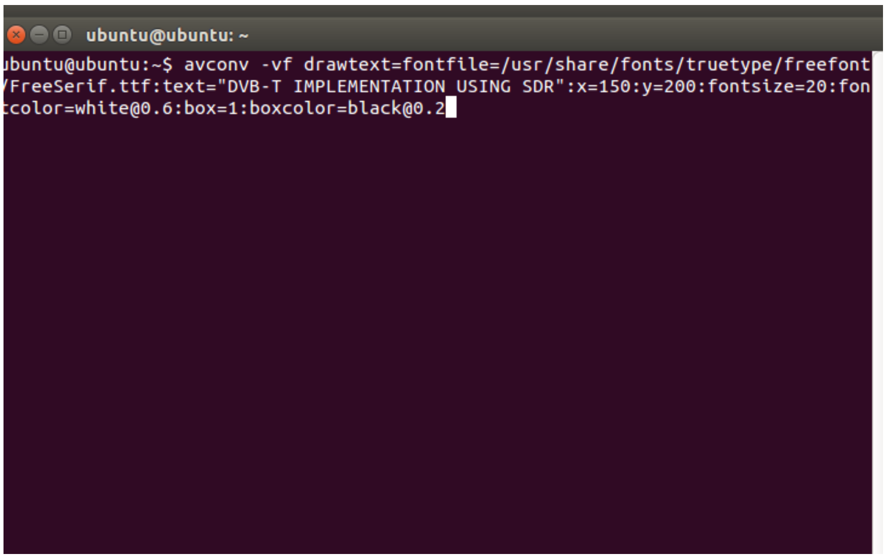
Added design requirements aside from the transmission and reception of the video signal, we utilized the tools provided by AVCONV is editing the broadcasted video signal. We attached the text "DVB-T IMPLEMENTATION USING SOFTWARE DEFINED RADIO" to the video. From the list of video filters in AVCONV, we made use of only the drawtext filter. The drawtext filter is used to draw a text or string from a specified file on a particular position of a video, using the libfreetype library This was accomplished by taking the following steps after the text has been selected;

i) Choosing the position on the screen to write the text
ii) Selecting a font size
iii) Selecting a font colour
iv) Selecting the box
v) Selecting the box colour

The complete command for the operation is shown in Figure 6.

2.4. Encoding of the Input Signal

Encoding the signal takes two steps; Audio Encoding and Video Encoding. In audio encoding, we set the audio file format, then the value for the audio sampling frequency, followed by the value for the audio bitrate, and lastly the number of audio channels. The audio file format we chose was the MP2. MP2 is an audio file format used with MPEG Audio Stream. It is the dominant standard for audio broadcasting. The sampling rates available for audio encoding are 32kHz, 44.1kHz and 48kH. We chose "44.1kHz" as our sampling frequency. This is because if the sampling rate is too low (32kHz), we won't get good audio quality and though 48kHz has a very good audio quality, choosing it means that we need a computer with a very high processing speed. So, the choice was a compromise between very good audio quality and the available processing strength of the computer. Next is the choice of the audio bitrate. The audio bitrates available are 32, 48, 56, 64, 80, 96, 112, 128, 160, 192, 224, 256, 320 and 384Kbps (kilobits per second). The higher the bitrate, the higher the audio quality and the higher the processing speed that is required by the computer. Again, a good compromise is struck at a choice of “128Kbps” as the bitrate. The different audio channels available are 2.0, 2.1, 5.1, 6.1, and 7.1. We chose “2.0” as our audio channel because of its simplicity.



```
ıbuntu@ubuntu:~$ avconv -vf drawtext=fontfile=/usr/share/fonts/truetype/freefont
/FreeSerif.ttf:text="DVB-T IMPLEMENTATION USING SDR":x=150:y=200:fontsize=20:fon
tcolor=white@0.6:box=1:boxcolor=black@0.2
```

Figure 6. Terminal window showing video filtering using AVCONV

Encoding the video signal involves setting the video file format, the resolution, the aspect ratio, the frame rate, and the video bitrate. We chose the video file format, “MPEG2VIDEO”. Though MPEG2VIDEO is not as efficient as newer standards such as H.264/AVC and H.265/AVC, it is used in digital video broadcasting because it is backwards compatible with existing hardware and software. The common image resolutions used are 320×240, 480×320, 640×480, 640×360, 720×480, 800×600, 1024×768, 1280×720, and 1440×1080. The higher the resolution chosen, the higher the video quality, and the higher the processing speed that is required by the computer. As a compromise for the processing speed, we choose a resolution of “320×240”. The common aspect ratios used are 3:2, 4:3 and 16:9 whose choice is based on the choice of the resolution so we chose “4:3”. We chose a frame rate of “60fps” and a video bitrate of “4Mbps”. The video bitrate for mpeg2video is within the range of 4 – 100Mbps and the lower, the less processing power required.

2.5. MPEG-2 TS Multiplexing

MPEG-2 TS combines audio and video into one file. There are certain parameter values required in multiplexing the audio and video signals to an MPEG2-TS. We needed to set the multiplexing rate (mux rate), the MPEG2-TS ID, the MPEG2 Service ID, the Packet Identifier (PID) and the Program Map Table (PMT). Table 3 presents the values used.

Table 3. System configuration for MPEG2-TS signal

System Configuration	Values
Muxrate	4524064
MPEG3-TS ID	1025
MPEG2-TS Service ID	1
PID	0x1021
PMT	0x1020

After a successful compression of the audio and video file into an MPEG2-TS, the MPEG2-TS is pipelined to GNU Radio through a TCP socket.

2.6. Modulating Stage

The important parameter values to be set for modulation includes the following, however, all the design specifications we used in the modulating stage was gotten from the "ETSI EN 300 744" standard [7, 13, 14, 15].

i) Transmission Bandwidth
ii) Transmission Mode
iii) Hierarchy Type
iv) Constellation Type, Code Rate and Guard Interval
v) FFT Size
vi) Transmission Centre Frequency and Symbol Rate

In most of the design choices in the modulation stage, the processing power of the computer was a major criterion. From the choices available for the transmission bandwidths which are 6MHz, 7MHz and 8MHz, we chose "6MHz". Also, there are two transmission modes available for selection, 2K mode and 8K mode. The 2K mode is suitable for single transmitter operation and small SFN networks with limited transmitter distances. The 8K mode can be used both for single transmitter operation and for small and large SFN networks. The design requirement for this study is to make a small SFN network that can be transmitted over a limited distance. Therefore, the 2K mode was selected. The 2K mode gives a maximum output of 1705 carriers, with symbol interleaving carried out at 2bits words mapped onto 1512 active carriers per OFDM symbol. The hierarchy types available for selection are non-hierarchical and hierarchical. The hierarchical type transmits two MPEG2-TS at the same time; a standard-definition television (SDTV) signal, and a high-definition television (HDTV) signal on the same carrier. In the non-hierarchical type, only one MPEG2-TS is transmitted on a carrier. In order not to complicate the design, we selected the non-hierarchical type. The constellation types used in ETSI EN 300 744 standards are QPSK (or 4QAM), 16QAM and 64QAM. The constellation type chosen was QPSK because it was supported for the mux rate of 4Mbps selected for the MPEG2-TS signal produced. The choice of the code rate and guard interval was guided by the values of the bitrate, code rate and guard interval for the different constellation types at 6MHz bandwidth that is shown in Table 4. We selected a "code rate of 1/2" and a "guard interval of 1/32" because of the multiplexing rate of the MPEG2-TS file.

Table 4. Useful bitrate (Kbit/s) for all combinations of guard interval, constellation and code rate for non-hierarchical systems for 6MHz channels

MODULATION	CODE RATE	GUARD INTERVAL			
		1/4	1/8	1/16	1/32
QPSK	½	3,732	4,147	4,391	4,524
	2/3	4,976	5,529	5,855	6,032
	¾	5,599	6,221	6,587	6,786
	5/6	6,221	6,912	7,318	7,540
	7/8	6,532	7,257	7,684	7,917
16-QAM	½	7,465	8,294	8,782	9,048
	2/3	9,953	11,059	11,709	12,064
	¾	11,197	12,441	13,173	13,572
	5/6	12,441	13,824	14,637	15,080
	7/8	13,063	14,515	15,369	15,834
64-QAM	½	11,197	12,441	13,173	13,572
	2/3	14,929	16,588	17,564	18,096
	¾	16,796	18,662	19,760	20,358
	5/6	18,662	20,735	21,955	22,620
	7/8	19,595	21,772	23,053	23,751

The COFDM modulation used in DVB-T creates frequency domain bins and an Inverse Fast Fourier Transform is done to convert these bins in the frequency domain to the time domain. There are two modes of FFT to be selected in the DVB-T system; the 2K FFT mode, and 8K FFT mode. The 2K FFT mode is selected when the mode of transmission is 2K, and the 8K FFT mode is selected when the mode of transmission is 8K. At 2K the FFT size is 2048 bins. The frequency ranges utilized by TV channels are the ranges of 174 – 230MHz and 470 – 862MHz. We chose a free TV channel in the bands for this study and that was the channel frequency, 441MHz.

The flowgraph in GNU radio consisted of the TCP Source, Energy Dispersal, Reed Solomon Encoder, Convolutional Interleaver, Inner Coder, Bit Inner Interleaver, Symbol Inner Interleaver, DVB-T Map, FFT, OFDM Cyclic Prefixer, and UHD: USRP Sink.

The TCP block receives the MPEG2-TS from AVCONV by setting the TCP address and port number to be the same as the TCP address and port number at the output of AVCONV [16]. The energy dispersal block receives the MPEG2-TS from the TCP Source block. Energy dispersal operation aims to achieve a flat power density spectrum and avoid the occurrence of long strings 0's and 1's. Each energy dispersal block consists of 8 MUX packets of 188bytes. Thus, a block is made up of 1504bytes. 1 block is selected to disperse the energy in the MPEG2-TS. The output of the energy dispersal is delivered to a Reed Solomon encoder. The Reed

Solomon encoder acts on each block of data equal to 188bytes packets of MPEG2-TS to produce an output of 204 bytes. The Reed Solomon encoder is the first level of error correction. The data obtained from the output of the Reed Solomon encoder (N = 204bytes) is applied to a convolutional interleaver. Convolutional interleaving is used to rearrange the transmitted data sequence in such a way that it becomes more rugged to long sequence errors. The Convolutional Interleaver has the following standard parameters: the number of shift registers, J and the depth of the shift registers, M joined by Equation (1).

$$J \times M \; cells \; = \; N \qquad (1)$$

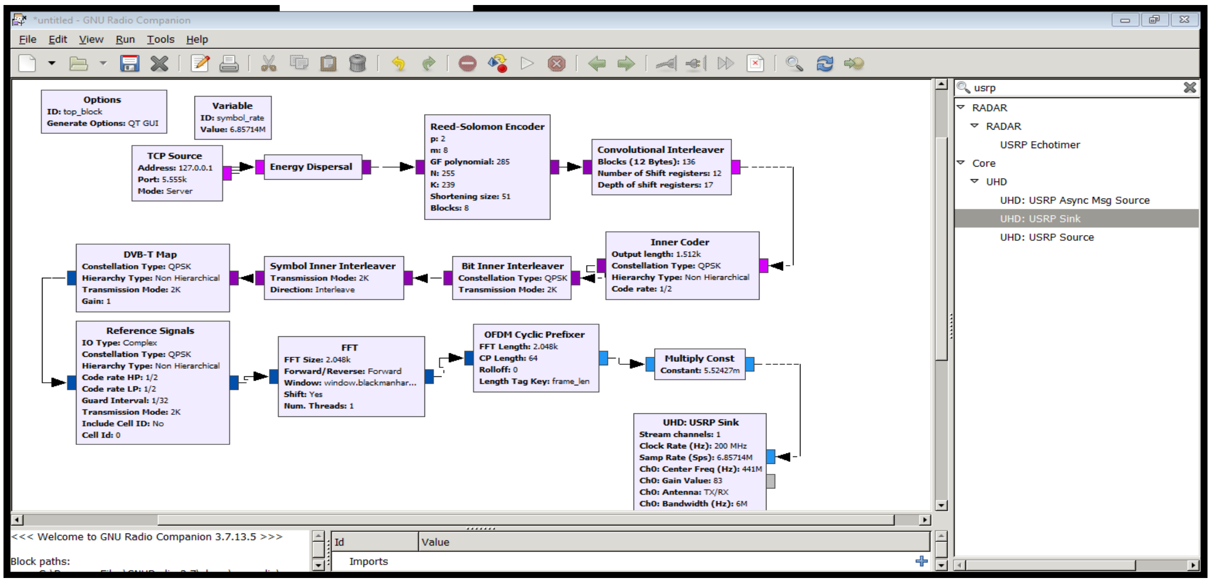
The data from the convolutional interleaver is sent to the inner coder. The inner coder is the second level of error correction. The code rate is set in this block. The data from the inner coder is sent to the bit inner interleaver. Bit interleaving is a way of rearranging the data sequence to reduce the influence of burst errors in each bit. Data from the bit inner interleaver is sent to the symbol inner interleaver. The symbol interleaver rearranges the data sequence to reduce burst errors in a group of bits. The result is prepared for the mapping of bits to the constellation according to QPSK. Data from the symbol interleaver is sent to the mapper. The digital bit sequence is mapped onto a baseband modulated sequence of complex numbers by the DVBT Map. The mapper maps the bits onto the QPSK constellation. The output of the mapper is sent to the reference signals block. Data from the reference signal block is sent to the FFT block for the inverse FFT operation. To prevent ISI and ICI in the SFN that is usually employed in the DVB-T implementations, a cyclic prefix (CP) of length 1/32 of the FFT length was used. The USRP Sink Block receives the data to be transmitted, processes the data, and sets the data transmission at a frequency of 441MHz.

2.7.Transmitting Stage

This stage is made up of the USRP B210 and an antenna. The flowgraph that is run in GRC programs the FPGA in the USRP and the USRP transmits the data at a frequency of 441MHz.

3.0.Results and Discussion

Figure 7 shows the flowgraph in GNU radio for modulating the MPEG2-TS file received from AVCONC.


*untitled - GNU Radio Companion
File Edit View Run Tools Help
usrp
RADAR
RADAR
USRP Echotimer
Core
UHD
UHD: USRP Async Msg Source
UHD: USRP Sink
UHD: USRP Source

Options
ID: top_block
Generate Options: QT GUI

Variable
ID: symbol_rate
Value: 6.85714M

TCP Source
Address: 127.0.0.1
Port: 5.555k
Mode: Server

Energy Dispersal

Reed-Solomon Encoder
p: 2
m: 8
GF polynomial: 285
N: 255
K: 239
Shortening size: 51
Blocks: 8

Convolutional Interleaver
Blocks (12 Bytes): 136
Number of Shift registers: 12
Depth of shift registers: 17

Inner Coder
Output length: 1.512k
Constellation Type: QPSK
Hierarchy Type: Non Hierarchical
Code rate: 1/2

Bit Inner Interleaver
Constellation Type: QPSK
Transmission Mode: 2K

Symbol Inner Interleaver
Transmission Mode: 2K
Direction: Interleave

DVB-T Map
Constellation Type: QPSK
Hierarchy Type: Non Hierarchical
Transmission Mode: 2K
Gain: 1

Reference Signals
IO Type: Complex
Constellation Type: QPSK
Hierarchy Type: Non Hierarchical
Code rate HP: 1/2
Code rate LP: 1/2
Guard Interval: 1/32
Transmission Mode: 2K
Include Cell ID: No
Cell Id: 0

FFT
FFT Size: 2.048k
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1

OFDM Cyclic Prefixer
FFT Length: 2.048k
CP Length: 64
Rolloff: 0
Length Tag Key: frame_len

Multiply Const
Constant: 5.52427m

UHD: USRP Sink
Stream channels: 1
Clock Rate (Hz): 200 MHz
Samp Rate (Sps): 6.85714M
Ch0: Center Freq (Hz): 441M
Ch0: Gain Value: 83
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 6M

```
<<< Welcome to GNU Radio Companion 3.7.13.5 >>>

Block paths:
```

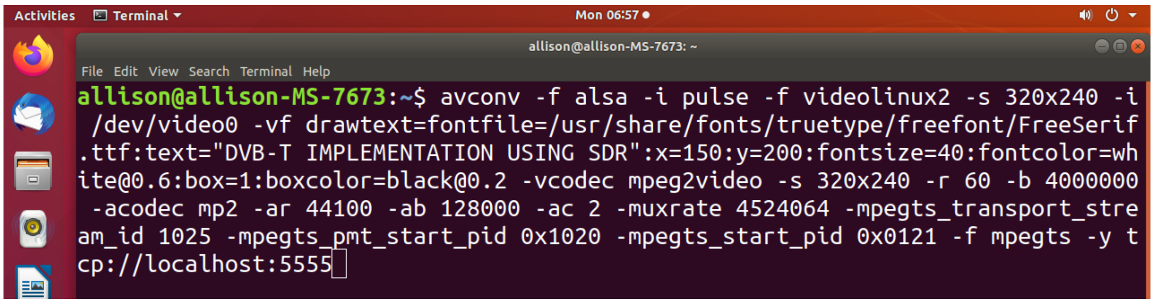
Id	Value
Imports	

Figure 7 DVB-T transmitter modulation flowgraph

Beyond the flowgraph design, we edited the generated python script to increase the flexibility of the application such as changing the channel frequency-MHz mode, code rate, constellation and guard interval variable.

Setup steps for running the transmitter end

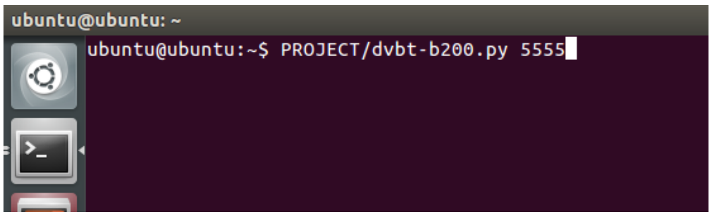
i) Connect the antenna to the USRP B210
ii) Connect the USRP B210 to the laptop using a USB cable, this serves to power the USRP B210 and also connect the device to the laptop.
iii) To let AVCONV and the python script (which was saved as dvbt-b210.py) be connected, we use localhost: 5555 (which is chosen as the default value).
iv) Once this is done, within a terminal we run the following code to convert the live stream to MPEG transport stream and send the output to localhost 5555 as shown in Figure 8.


Activities
Terminal
Mon 06:57
allison@allison-MS-7673: ~
File Edit View Search Terminal Help

```
allison@allison-MS-7673:~$ avconv -f alsa -i pulse -f videolinux2 -s 320x240 -i
 /dev/video0 -vf drawtext=fontfile=/usr/share/fonts/truetype/freefont/FreeSerif
.ttf:text="DVB-T IMPLEMENTATION USING SDR":x=150:y=200:fontsize=40:fontcolor=wh
ite@0.6:box=1:boxcolor=black@0.2 -vcodec mpeg2video -s 320x240 -r 60 -b 4000000
 -acodec mp2 -ar 44100 -ab 128000 -ac 2 -muxrate 4524064 -mpegts_transport_stre
am_id 1025 -mpegts_pmt_start_pid 0x1020 -mpegts_start_pid 0x0121 -f mpegts -y t
cp://localhost:5555
```

Figure 8 Terminal window for running the AVCONV

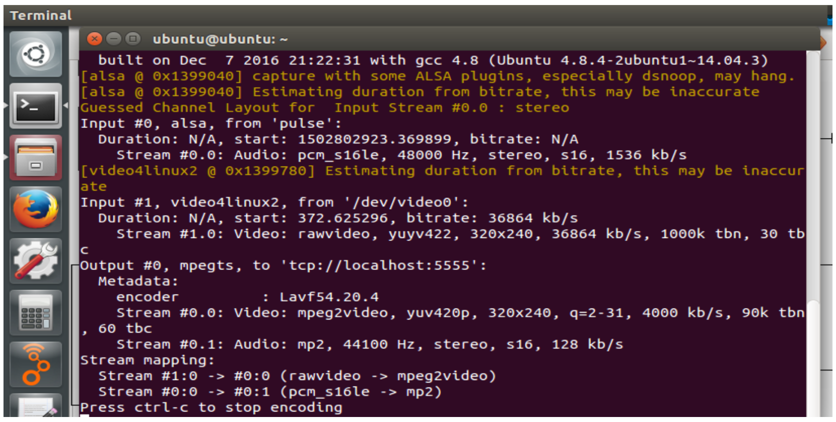
v) Then open a second terminal window to run the python script as shown in Figure 9 (the output of the AVCONV is sent to the local host 5555 which is used as in the input for the python script)



```
ubuntu@ubuntu:~$ PROJECT/dvbt-b200.py 5555
```

Figure 9 Terminal Window for running python script

vi) Then we run the AVCONV command in a second terminal window to transmit the video and the result of the command should resemble Figure 10.

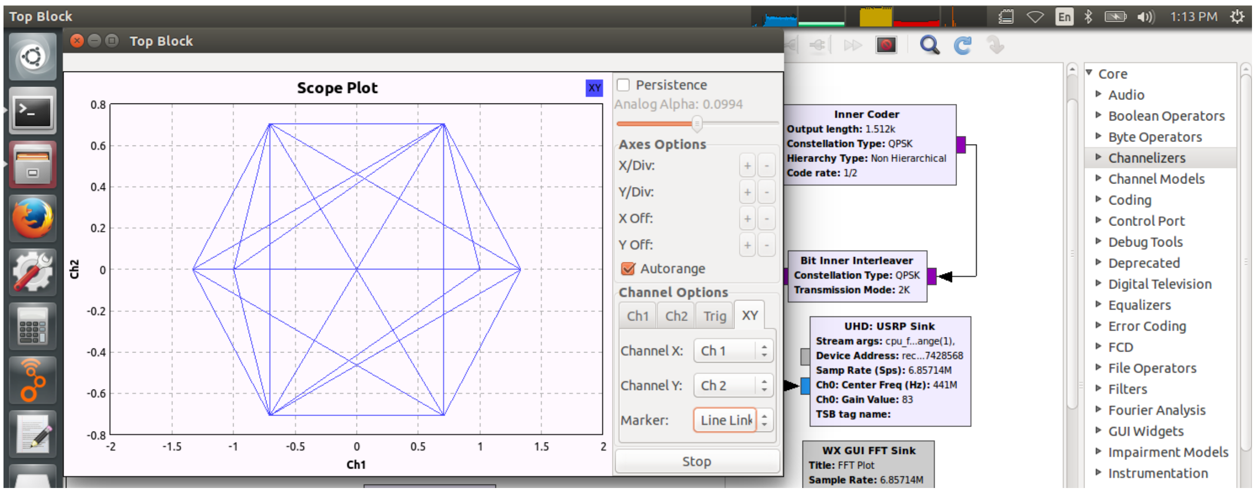


```
  built on Dec  7 2016 21:22:31 with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04.3)
[alsa @ 0x1399040] capture with some ALSA plugins, especially dsnoop, may hang.
[alsa @ 0x1399040] Estimating duration from bitrate, this may be inaccurate
Guessed Channel Layout for  Input Stream #0.0 : stereo
Input #0, alsa, from 'pulse':
  Duration: N/A, start: 1502802923.369899, bitrate: N/A
    Stream #0.0: Audio: pcm_s16le, 48000 Hz, stereo, s16, 1536 kb/s
[video4linux2 @ 0x1399780] Estimating duration from bitrate, this may be inaccur
ate
Input #1, video4linux2, from '/dev/video0':
  Duration: N/A, start: 372.625296, bitrate: 36864 kb/s
    Stream #1.0: Video: rawvideo, yuyv422, 320x240, 36864 kb/s, 1000k tbn, 30 tb
c
Output #0, mpegts, to 'tcp://localhost:5555':
  Metadata:
    encoder         : Lavf54.20.4
    Stream #0.0: Video: mpeg2video, yuv420p, 320x240, q=2-31, 4000 kb/s, 90k tbn
, 60 tbc
    Stream #0.1: Audio: mp2, 44100 Hz, stereo, s16, 128 kb/s
Stream mapping:
  Stream #1:0 -> #0:0 (rawvideo -> mpeg2video)
  Stream #0:0 -> #0:1 (pcm_s16le -> mp2)
Press ctrl-c to stop encoding
```

Figure 10 Terminal Window showing working AVCONV

vii) Thus, the live feed is transmitted and is to be received via the RTL SDR on the receiving laptop.

The constellation and FFT plot are shown in Figures 11 and 12 respectively.



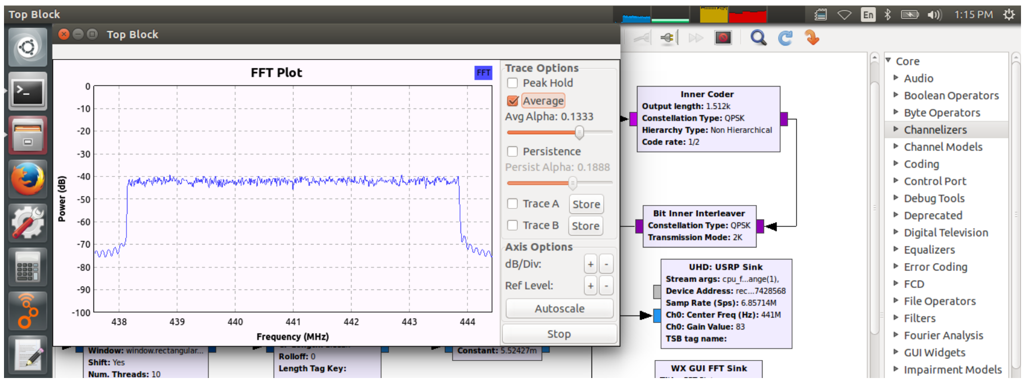
Inner Coder
Output length: 1.512k
Constellation Type: QPSK
Hierarchy Type: Non Hierarchical
Code rate: 1/2

Bit Inner Interleaver
Constellation Type: QPSK
Transmission Mode: 2K

UHD: USRP Sink
Stream args: cpu_f...ange(1),
Device Address: rec...7428568
Samp Rate (Sps): 6.85714M
Ch0: Center Freq (Hz): 441M
Ch0: Gain Value: 83
TSB tag name:

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 6.85714M

Figure 11 Constellation Plot



Inner Coder
Output length: 1.512k
Constellation Type: QPSK
Hierarchy Type: Non Hierarchical
Code rate: 1/2

Bit Inner Interleaver
Constellation Type: QPSK
Transmission Mode: 2K

UHD: USRP Sink
Stream args: cpu_f...ange(1),
Device Address: rec...7428568
Samp Rate (Sps): 6.85714M
Ch0: Center Freq (Hz): 441M
Ch0: Gain Value: 83
TSB tag name:

WX GUI FFT Sink

Window: window.rectangular...
Shift: Yes
Num. Threads: 10

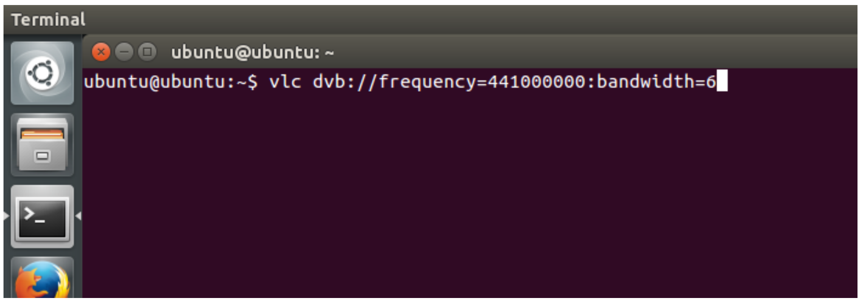
Rolloff: 0
Length Tag Key:

Constant: 5.52427m

Figure 12 FFT Plot

The reception of live feed using RTL SDR is carried out on another laptop having Ubuntu 16.04 operating system. The following steps are carried out to receive the live video stream from the transmitting computer:

i) Connect the antenna to the RTL SDR dongle.
ii) Connect the dongle into the USB port of the laptop
iii) From the terminal window VLC media player runs as shown in Figure 13; whereby the variables set the parameters required by the VLC to receive the digital video broadcast using the RTL SDR



```
ubuntu@ubuntu:~$ vlc dvb://frequency=441000000:bandwidth=6
```

Figure 13 Terminal Window Command to run VLC media player

Figure 14 shows the final implementation of the DVB-T system showing the transmitter and receiver streaming a real-time video on a channel frequency of 441 MHz at a bandwidth of 6.857 MHz. The implementation is demonstrated with an 8-PSK constellation at a ½ code rate. The developed system supports the following DVB-T transmission modes:

i) Code rates: 1/2, 2/3, 5/6;
ii) Modulation schemes: 2k (2048 subcarriers);
iii) Cyclic prefix sizes: 1/4. 1/8, 1/16, 1/32;
iv) Constellations: QPSK, 16QAM, 64QAM;
v) Channel bandwidth: 8MHz, 7MHz, 5MHz.

As seen, the spectrum plot of the transmitted signal is an OFDM signal. The system attains a maximum transmission range of 20.4 m with manageable end-to-end delay and packet loss at a maximum gain of 89.75 dB.

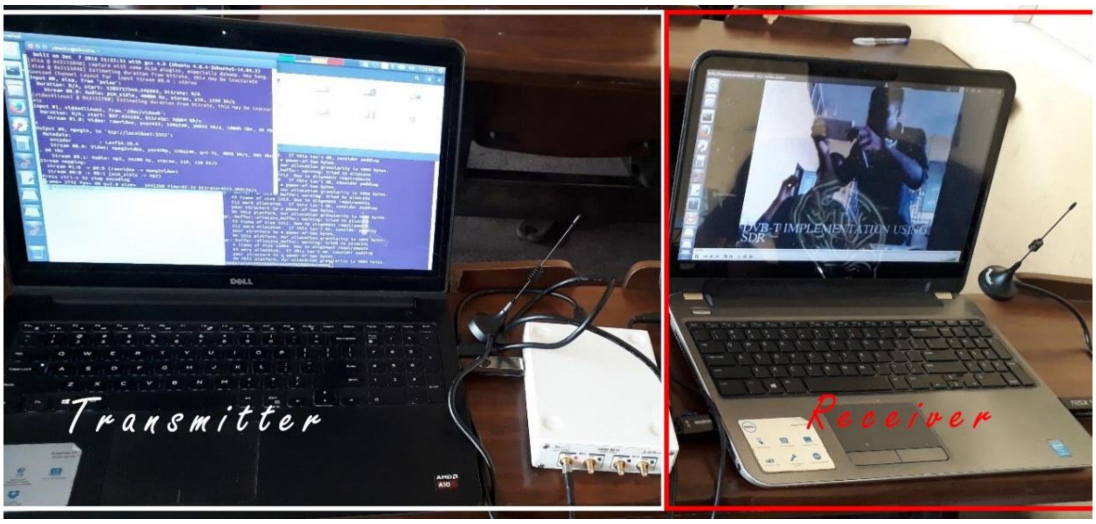


Figure 14 DVB-T Implementation showing both the transmitter and receiver ends

In comparison with past DVB-T works referenced in this study, [4] had a DVB-T transmitter with extreme CPU requirements due to the presence of several computational-intensive blocks in the developed program, though the memory usage was moderate. In their review, the program could be improved through the perspective of the improved processing speed of the underlying framework, GNU Radio. GNU Radio presently implements a parallel architecture and the proposed program in this study ran on two DELL computers. An Inspiron 14 with an AMD and an Inspiron 15 Core i3 with a low computational resource of about 40%.

During the test that was carried out, it was observed that the transmission range determined the power gain of the transmitter and the receiving ends. Given that the USRP B210 achieved a maximum of 89.75 dB, the maximum distance was 20.4 m. Also, there was an occasional USB interface error during the test which was solved by unplugging and plugging the USB cable.

4. Conclusion

Current multicore CPU and cheap SDR kits enable a pure software design of DVB-T testbeds. In this study, we have designed a DVB-T SDR transmitter using USRP B210 as a transmitter with the codes written in python script. The designed DVB-T transmitter can run on a multilevel core system processor using a selected percentage of computational power and maximum amount of RAM. It has been shown that the DVB-T transmitted signal can be successfully received by an ad-hoc receiver such as RTL SDR implemented using a VLC player in a Linux operating system which was the aim of this study. Thus, following the problem related to analogue broadcasting such as flexibility, immunity to noise, etc. it can be said the study met these requirements.

References

[1] Edeko F. O. "Electronic communication systems". First Edition, August 1997. pp 18-19,22-35

[2] Grob. B and C E Herndon," Basic Television and Video Systems".6th Edition. New York: McGraw Hill, 1999

[3] Crespi, F.L., Maglioli, M., Benco, S. and Perotti, A., (2012). A real-time video broadcasting system based on the GNU Radio-USRP2 platform. In *Proc. Karlsruhe Workshop on Software Radios (WSR)*.

[4] Pellegrini, V., Bacci, G. and Luise, M., (2008). Soft-DVB: A fully-software GNURadio-based ETSI DVB-T modulator. *Proc. WSR*, *8*.

[5] Digital Video Broadcasting (DVB); Frame Structure Channel Coding and modulation for Digital Terrestrial Television; ETSI EN300 744 V.1.6.1; European Telecommunications Standard Institute: France 2009.

[6] Digital Video Broadcasting (DVB); Frame Structure Channel Coding and modulation for a second-generation digital terrestrial television Broadcasting System (DVB-T2). DVB Consortium. February 2011

[7] ETSI Standard: EN 300 744 V 1.5.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and Modulation for Digital Terrestrial Television (2004-2006).

[8] GR-DVBT, 2017. DVB-T implementation in GNUradio – part 2. [Online] Available at: https://yo3iiu.ro/blog/?p=1220 (accessed May 5th, 2017)

[9] Ettus Research, 2017. USRP Bus series: USRP B210 (Board Only). [Online] Available at: https://www.ettus.com/product/details/UB210-KIT (accessed May 25th, 2017)

[10] Nooelec, 2017. Nooelec NESDR SMArt v4 Bundle - Premium RTL-SDR w/ Aluminum Enclosure, 0.5PPM TCXO, SMA Input & 3 Antennas. RTL2832U & R820T2-Based. [Online] Available at: https://www.nooelec.com/store/nesdr-smart.html. (Accessed May 24th, 2017)

[11] Osmocom, 201.Software-defined radio (SDR): RTL-SDR [Online] https://osmocom.org/projects/rtl-sdr/wiki (accessed June 1st, 2017)

[12] GNU Radio, 2021. GNU Radio [Online] https://wiki.gnuradio.org/index.php/Main_Page (accessed May 26th, 2017)

[13] ETSI Standard: EN 300 744 V 1.3.1, Digital Video Broadcasting (DVB); Framing structure, channel coding and Modulation for Digital Terrestrial Television (2008-2010).

[14] Reimans, Ulrich; "A Guideline for the Use of DVB Specifications and Standards, DVB-TM 1694 rev.1 September 1996

[15] Seranted D, Julian O, Ries L, Thevenon P. "The Digital TV Case: Positioning Using Signals-of-Opportunity based on OFDM modulation. In proceedings of the 10N GNSS, Portland, OR, USA, 19-23 September 2011.

[16] AVCONV Documentation, 2017. AVCONV Documentation: Description [Online] Available at: https://libav.org/avconv.html#Description (Accessed June 4th, 2017).