EAI Endorsed Transactions
on Mobile Communications and Applications Research Article EAI.EU

# Implementation of Cognitive Radio Network Testbed for Multimedia Communications

Patrick DaSilva[1], Anuja Ghising[1], Siddharth Patil[1] and Honggang Wang[1,*]

[1]Department of Electrical and Computer Engineering, University of Massachusetts Dartmouth, Dartmouth, MA 02747-2300, United States

## Abstract

A cognitive radio is a form of wireless communication in which a transceiver can intelligently detect which communication channels are in use and which are not, and instantly move into vacant channels while avoiding occupied ones. This intelligently avoids interference amongst users and provides the ability to use up all available bandwidth on the RF spectrum. There are many ways in which the communication can be demonstrated. This paper examines how we have implemented a cognitive radio network testbed using software defined radios (SDRs) for multimedia communications, where components that have been traditionally implemented in hardware are instead implemented by means of software on a personal computer or embedded system to communicate and transfer a file between each other. We attempt to demonstrate potential use of SDRs for future multimedia applications.

**Keywords:** GMSK, GNU Radio, SDR, USRP, VLC, wireless streaming.

Received on 24 August 2018, accepted on 18 September 2018, published on 14 December 2018

doi: 10.4108/eai.10-7-2018.156087

## 1. Introduction

Cognitive radio (CR) is defined as “a form of wireless communication in which a transceiver can intelligently detect which communication channels are in use and which are not, and instantly move into vacant channels while avoiding occupied ones [1].” This intelligently avoids interference amongst users and provides the ability to use up all available bandwidth on the radio frequency (RF) spectrum [1]. The most popular way to do this is with a software defined radio (SDR), defined as a radio communication system where components traditionally implemented in hardware are implemented in software on a processor-based system [2]. Further, the multimedia communication industry continues to grow rapidly, especially with the wide deployment of smartphones and growing interests in multimedia social networks. There is an urgent need to train the next generation engineers and scientists in the cross-fields of multimedia and communications. In this paper, we successfully setup a testbed that supports developing a cognitive radio network for multimedia communications for the purpose.

Section 2 will discuss the background needed to complete this project. The network topology of connecting multiple software defined radios together will be discussed in Section 3. Section 4 will describe Gaussian Minimum Shift Keying, the primary modulation scheme used throughout the project. Sections 5 and 6 will discuss the two project goals completed and finally Section 7 will conclude this paper.

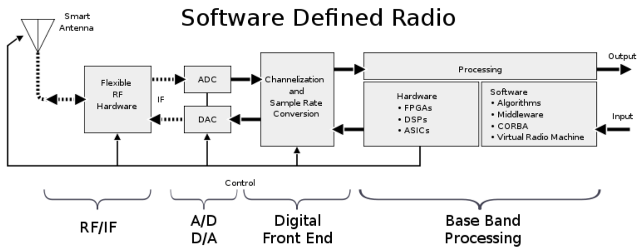
[*]Corresponding author. Email: hwang1@umassd.edu

## 2. Background

### 2.1 Software Defined Radio

Prior to SDR, a radio would need to be implemented for every protocol each time a new device or standard was released. For example, cellular, Wi-Fi, and Bluetooth need different processing to extract data, but similar RF circuitry. With the advent of SDR, each protocol can be implemented in software, allowing each radio to be reconfigurable and require less hardware.

An SDR (Fig. 1) is made up of three main components: the user interface (UI), data engine, and RF front end. The UI serves as the main interface between the end user and SDR. Mainly implemented in software, the UI can also be implemented in hardware. The primary objectives of the UI are to perform digital signal processing (DSP), modulation/demodulation, digital filtering, time-domain to frequency-domain conversions, and controlling of all hardware and hardware DSP functions. Software implementations of the UI include GNU Radio, MATLAB, Redhawk, QtRadio, and SDR#.



ADC: analog to digital converter
ASIC: application-specific integrated circuit
DAC: digital to analog converter
DSP: digital signal processing
FPGA: Field Programmable Gate Array
IF: intermediate frequency
RF: radio frequency
**Figure 1.** Software defined radio [2].

The data engine section (Digital Front End + Base Band Processing) is usually implemented using micro-processors or Field Programmable Gate Arrays (FPGAs). The data engine sits between the RF frontend and UI performing multiple necessary functions. These functions can optionally perform DSP operations on digitized pre-filtered RF data before outputting the packed data over USB or Ethernet to the UI. It also handles sample rate mismatch using digital down-converters (DDC) and digital up-converters (DUC).

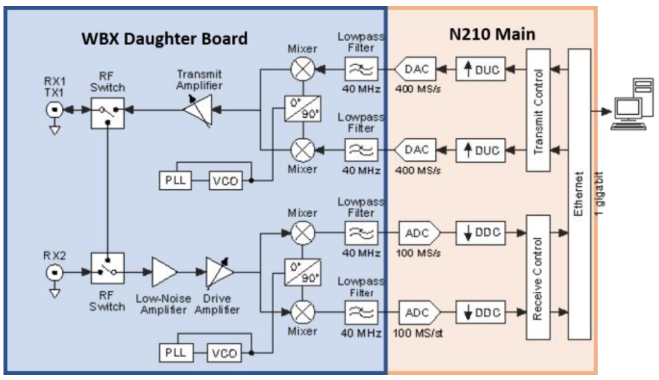
The RF section is a pure analog section apart from the analog to digital (ADC) and digital to analog (DAC) converters. This section interfaces sit between the analog outside world of RF and the rest of the digital radio. The section contains all front-end protection, filters, attenuators, transmitter filters, power amplification, and antenna switching.

## 2.2 Ettus Research USRP N210

The Ettus Research Universal Software Radio Peripheral (USRP) N210 (Fig. 2) is an SDR that takes in an analog RF input, samples it into the digital domain, and sends it across a User Datagram Protocol (UDP)/IP network interface to be processed using software. The USRP N210 can operate from baseband to 6 GHz in the RF spectrum via swappable daughter boards. The N210 architecture includes a customizable Xilinx Spartan 3A-DSP FPGA. The 1 Gbit/s Ethernet interface allows for streaming up to 25 million samples per second (MS/s) in full-duplex mode to and from the user interface. With the FPGA, the N210 has the potential to process up to 100 MS/s.

**Figure 2.** USRP N210 [3].

The N210 provides the RF frontend and data engine in a single enclosure. Fig. 3 shows the hardware layout of the N210, which contains the Ettus Research WBX daughter board as the analog portion of the RF section. The main N210 printed circuit board includes the hardware for the Data Engine Section. The WBX daughter board contains one transceiver antenna and one receiver antenna allowing the N210 to be placed in full duplex mode and stream up 25MS/s.



ADC: analog to digital converter
DAC: digital to analog converter
DDC: digital down converter
DUC: digital up converter
PLL: phase-locked loop
VCO: voltage-controlled oscillator
WBX: wide bandwidth transceiver
**Figure 3.** Hardware layout of USRP N210 [4].

## 2.3 GNU Radio

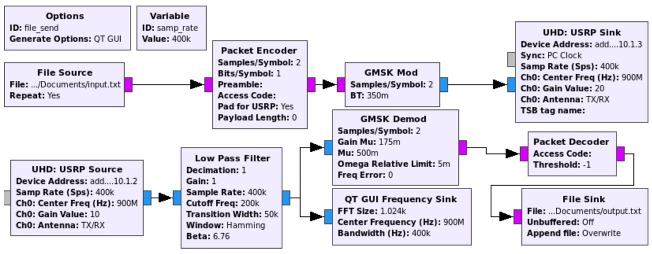
GNU Radio is a free and open source software development toolkit that provides signal processing blocks to implement software-defined radios and signal-processing systems. It is a highly modular and flowgraphoriented framework that comes with a comprehensive library of processing blocks easily combined to make complex signal processing applications. It enables users to design, simulate and deploy highly capable real-world radio systems and is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems.

GNU radio is primarily supported on the Linux platform and when installed, comes with a large variety of tools and programs which can be used with USRP software defined radios after installing the USRP Hardware Driver (UHD). On Linux these programs are typically installed into */usr/local/bin*. A few of the common tools are:

- *uhd_fft*: It is a very simple spectrum analyser tool which uses a connected UHD device (i.e., a USRP) to display the spectrum at a given frequency.
- *uhd_rx_cfile*: It records an I/Q sample stream using a connected UHD device. Samples are written to a file and can be analysed offline later, using either GNU radio.
- *uhd_siggen_gui*: It is a simple signal generator and can create the most common signals (sine, sweep, square, and noise).

GNU Radio Companion (GRC) is a graphical tool for creating signal processing flow graphs and generating flow-graph source code. On Linux systems, GRC is invoked by using the gnuradio-companion terminal command. GRC will pop up in its own window. A GNU Radio block can then be dragged into the main window and connected by clicking the edges.

Fig. 4 shows our early implementation of transferring a file via USRP N210 software defined radios. GNU radio natively supports the Ettus research line of SDRs.



**Options**
**ID:** file_send
**Generate Options:** QT GUI

**Variable**
**ID:** samp_rate
**Value:** 400k

**File Source**
**File:** .../Documents/input.txt
**Repeat:** Yes

**Packet Encoder**
**Samples/Symbol:** 2
**Bits/Symbol:** 1
**Preamble:**
**Access Code:**
**Pad for USRP:** Yes
**Payload Length:** 0

**GMSK Mod**
**Samples/Symbol:** 2
**BT:** 350m

**UHD: USRP Sink**
**Device Address:** add....10.1.3
**Sync:** PC Clock
**Samp Rate (Sps):** 400k
**Ch0: Center Freq (Hz):** 900M
**Ch0: Gain Value:** 20
**Ch0: Antenna:** TX/RX
**TSB tag name:**

**UHD: USRP Source**
**Device Address:** add....10.1.2
**Samp Rate (Sps):** 400k
**Ch0: Center Freq (Hz):** 900M
**Ch0: Gain Value:** 10
**Ch0: Antenna:** TX/RX

**Low Pass Filter**
**Decimation:** 1
**Gain:** 1
**Sample Rate:** 400k
**Cutoff Freq:** 200k
**Transition Width:** 50k
**Window:** Hamming
**Beta:** 6.76

**GMSK Demod**
**Samples/Symbol:** 2
**Gain Mu:** 175m
**Mu:** 500m
**Omega Relative Limit:** 5m
**Freq Error:** 0

**Packet Decoder**
**Access Code:**
**Threshold:** -1

**QT GUI Frequency Sink**
**FFT Size:** 1.024k
**Center Frequency (Hz):** 900M
**Bandwidth (Hz):** 400k

**File Sink**
**File:** ...Documents/output.txt
**Unbuffered:** Off
**Append file:** Overwrite

**Figure 4.** GNU Radio implementation of transferring a file over two USRP N210s.

Once the flowgraph is created, GRC creates a Python script from the graph. The Python script allows for manual manipulation of the block connections which is useful if the user wants more control over the signal flow. This provides the entire power and functionality of Python and its libraries, such as SciPy or NumPy for Python-centric processing of your signals or your favourite widget library to create any GUI you wish.

As described in the Guided Tutorials [5], GNU Radio allows for the easy development of custom out of context blocks developed in C++ and wrapped in Python. The blocks can be used in any GNU Radio flowgraph.

## 3. Connecting Multiple N210s

The first task we set out to do was connecting multiple Ettus Research USRP N210s to a single server. According to the Ettus Research N2X0 Series manual [6], multiple N210 radios can be connected to a single host over a network so long as each radio has a separate IP. Ettus Research recommends only connecting one radio per Ethernet interface and that each interface has its own subnet with the appropriate subnet mask. It is mentioned that theoretically a network switch would allow multiple radios to connect through a single interface so long as each Ethernet interface has its own subnet, and the corresponding USRP2 device is assigned an address in that subnet.

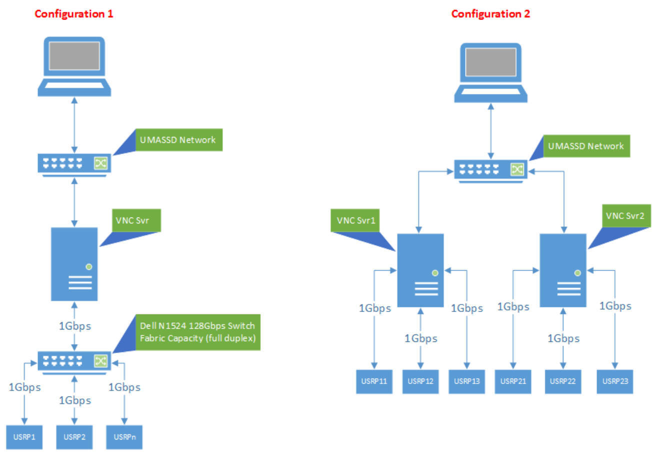
An example of setting up IP addressing for multiple N210s using separate interfaces is [6]:

- Configuration for USRP2 device 0
- Ethernet interface IPv4 address: 192.168.10.1
- Ethernet interface subnet mask: 255.255.255.0
- USRP2 device IPv4 address: 192.168.10.2
- Configuration for USRP2 device 1
- Ethernet interface IPv4 address: 192.168.20.1
- Ethernet interface subnet mask: 255.255.255.0
- USRP2 device IPv4 address: 192.168.20.2.

This example can be created with either two Ethernet interfaces on a single computer or with a layer 3 managed network switch setup with virtual local area networks (VLANs).

### 3.1 Network Configurations

To be able to connect multiple users to a testbed of N210 SDRs, two different network configurations were studied, and a list of pros and cons were generated. The configurations of interest can be seen in Fig. 5.



UMASSD: University of Massachusetts Dartmouth
USRP: Universal Software Radio Peripheral
VNC: Virtual Network Computing
**Figure 5.** N210 network configurations.

Configuration 1 contains a single XRDP host server (VNC Svr) with two Ethernet interfaces (Interface 1 and Interface 2). Interface 1 is connected to the University of Massachusetts (UMASS) Dartmouth network. Interface 1 is a dynamic host configuration protocol (DHCP) to get its information from UMASS. Interface 2 is setup with a static IP address for the cognitive radio network. The switch on the cognitive radio network is configured as a single VLAN and each N210 has a unique IP address on that VLAN.

Configuration 2 takes advantage of the four Ethernet interfaces available on the host server. This configuration contains m XRDP servers connected to the UMASSD network. Like before, Interface 1 is setup to dynamically get its IP information from the UMASS network. The remaining interfaces (2,3,4) are setup with static IP addresses on different subnets. Connected to each interface is a single SDR on the same subnet as the interface it is connected to.

The configuration settings for the host and N210 radios can be seen in Table 1.

Configuration 1 allows a single host to connect to more SDRs than Ethernet interfaces available with the use of a gigabit switch. The downside is all SDR traffic is funnelled through a single gigabit interface. The effect can be seen when multiple SDRs are communicating with the host server. As the sample rate of each radio increases, the bandwidth limit of the single gigabit interface is achieved faster, limiting the available bandwidth of each radio.

Configuration 2 overcomes this disadvantage by giving each SDR its own gigabit path to the host. The downside is multiple host servers or Ethernet interfaces are required to the host, increasing the complexity of the network configuration if additional servers are needed.

Table 1. Comparison of configurations

| | Configuration 1 | Configuration 2 |
|---|---|---|
| Host Intf 1 IP | DHCP | DHCP |
| Host Intf 1 Subnet | N/A | N/A |
| Host Intf 2 IP | 10.10.1.254 | 10.10.2.1 |
| Host Intf 2 Subnet | 255.255.255.0 | 255.255.255.0 |
| Host Intf 3 IP | Off | 10.10.3.1 |
| Host Intf 3 Subnet | Off | 255.255.255.0 |
| Host Intf 4 IP | Off | 10.10.4.1 |
| Host Intf 4 Subnet | Off | 255.255.255.0 |
| USRP 1 IP | 10.10.1.2 | 10.10.2.2 |
| USRP 2 IP | 10.10.1.3 | 10.10.3.2 |
| USRP 3 IP | 10.10.1.4 | 10.10.4.2 |

DHCP: dynamic host configuration protocol

After some preliminary testing with both configurations, Configuration 1 was picked. On Configuration 1, a single SDR could achieve 30 MS/s before the host became unresponsive. Two and three SDRs could achieve a simultaneous sample rate of 15 MS/s and 10 MS/s respectively before dropped packets were experienced.

For our application, it was estimated the SDR sample rate would not need to go above 1 MS/s and therefore Configuration 1 was chosen. Configuration 1 also allows additional SDRs to be added to the testbed without much additional configuration but to setup the SDR static IP.

### 3.2 Final Hardware Configuration

Our final hardware configuration for this project utilized three of the Ettus Research USRP N210 main data engine, WBX 50-2200 MHz Rx/Tx daughter board RF section, and the LP0965 Log Periodic PCB antenna. Even though we only needed 2 SDRs to communicate, we chose to utilize 3 to allow users to explore GNU Radio and the N210 in depth. Our final implementation can be seen in Fig. 6.

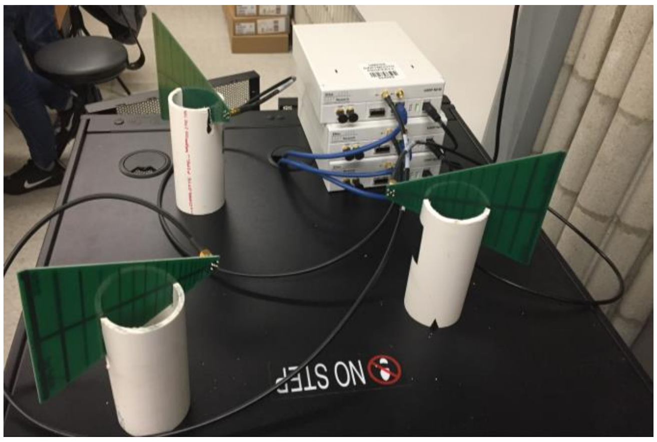


**Figure 6.** Final radio hardware configuration.

## 4. Gaussian Minimum Shift Keying Modulation

Gaussian Minimum Shift Keying (GMSK) is a form of modulation used in a variety of digital radio communications. The most widely used application known to use GMSK is GSM cellular technology used worldwide.

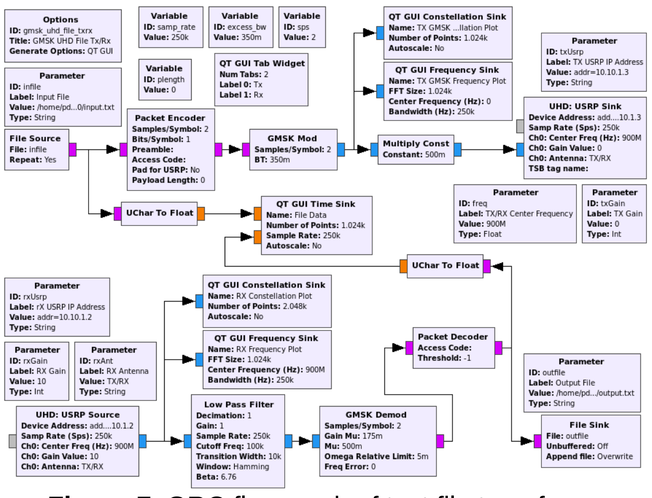
GMSK modulation is based on Minimum Shift Keying (MSK) which is a form of Continuous-Phase Frequency-Shift Keying (CPFSK). The problem with standard Phase Shift Keying (PSK) is that its sidebands extend out from the carrier interfering with adjacent digital communications. The PSK sidebands extend out from the carrier because of phase discontinuities in the modulated signal. MSK and therefore GMSK can be used to overcome these phase discontinuities. GMSK reduces its bandwidth usage by passing the MSK modulated signal through a gaussian low pass filter prior to applying it to the carrier.

Typically, GMSK is implemented using hardware components in a radio designed for a specific application. Hardware components must be picked carefully because GMSK requires great precision to work properly. With software defined radios, or more precisely, with GNU Radio and other software front ends, developers can utilize tried and true software implementations of GMSK. In GNU radio, this is as easy as dropping the GMSK Mod and GMSK Demod to transmit and receive a GMSK modulated signal. An application of GMSK digital communication can be seen in Fig. 4.

For both tasks set forth in this project, we used GMSK to send digital signals. In all our research, GMSK was the common modulation scheme users chose [7]–[10]. The other modulation scheme found in our research [11] and [12] was Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK).

## 5. Continuous File Transfer

To be able to transmit and receive a file's contents between two USRPs, the GNU Radio flowgraph (Fig. 7) was constructed. The primary data path followed a similar structure as found in the research [7]–[10] with the exception that the file sink block was set to repeat.



**Options**
**ID:** gmsk_uhd_file_txrx
**Title:** GMSK UHD File Tx/Rx
**Generate Options:** QT GUI

**Variable**
**ID:** samp_rate
**Value:** 250k

**Variable**
**ID:** excess_bw
**Value:** 350m

**Variable**
**ID:** sps
**Value:** 2

**Parameter**
**ID:** infile
**Label:** Input File
**Value:** /home/pd...0/input.txt
**Type:** String

**Variable**
**ID:** plength
**Value:** 0

**QT GUI Tab Widget**
**Num Tabs:** 2
**Label 0:** Tx
**Label 1:** Rx

**File Source**
**File:** infile
**Repeat:** Yes

**Packet Encoder**
**Samples/Symbol:** 2
**Bits/Symbol:** 1
**Preamble:**
**Access Code:**
**Pad for USRP:** No
**Payload Length:** 0

**GMSK Mod**
**Samples/Symbol:** 2
**BT:** 350m

**QT GUI Constellation Sink**
**Name:** TX GMSK ...llation Plot
**Number of Points:** 1.024k
**Autoscale:** No

**QT GUI Frequency Sink**
**Name:** TX GMSK Frequency Plot
**FFT Size:** 1.024k
**Center Frequency (Hz):** 0
**Bandwidth (Hz):** 250k

**Multiply Const**
**Constant:** 500m

**Parameter**
**ID:** txUsrp
**Label:** TX USRP IP Address
**Value:** addr=10.10.1.3
**Type:** String

**UHD: USRP Sink**
**Device Address:** add....10.1.3
**Samp Rate (Sps):** 250k
**Ch0: Center Freq (Hz):** 900M
**Ch0: Gain Value:** 0
**Ch0: Antenna:** TX/RX
**TSB tag name:**

**UChar To Float**

**QT GUI Time Sink**
**Name:** File Data
**Number of Points:** 1.024k
**Sample Rate:** 250k
**Autoscale:** No

**Parameter**
**ID:** freq
**Label:** TX/RX Center Frequency
**Value:** 900M
**Type:** Float

**Parameter**
**ID:** txGain
**Label:** TX Gain
**Value:** 0
**Type:** Int

**UChar To Float**

**Parameter**
**ID:** rxUsrp
**Label:** rX USRP IP Address
**Value:** addr=10.10.1.2
**Type:** String

**QT GUI Constellation Sink**
**Name:** RX Constellation Plot
**Number of Points:** 2.048k
**Autoscale:** No

**Parameter**
**ID:** rxGain
**Label:** RX Gain
**Value:** 10
**Type:** Int

**Parameter**
**ID:** rxAnt
**Label:** RX Antenna
**Value:** TX/RX
**Type:** String

**QT GUI Frequency Sink**
**Name:** RX Frequency Plot
**FFT Size:** 1.024k
**Center Frequency (Hz):** 900M
**Bandwidth (Hz):** 250k

**Packet Decoder**
**Access Code:**
**Threshold:** -1

**Parameter**
**ID:** outfile
**Label:** Output File
**Value:** /home/pd.../output.txt
**Type:** String

**UHD: USRP Source**
**Device Address:** add....10.1.2
**Samp Rate (Sps):** 250k
**Ch0: Center Freq (Hz):** 900M
**Ch0: Gain Value:** 10
**Ch0: Antenna:** TX/RX

**Low Pass Filter**
**Decimation:** 1
**Gain:** 1
**Sample Rate:** 250k
**Cutoff Freq:** 100k
**Transition Width:** 10k
**Window:** Hamming
**Beta:** 6.76

**GMSK Demod**
**Samples/Symbol:** 2
**Gain Mu:** 175m
**Mu:** 500m
**Omega Relative Limit:** 5m
**Freq Error:** 0

**File Sink**
**File:** outfile
**Unbuffered:** Off
**Append file:** Overwrite

**Figure 7.** GRC flowgraph of text file transfer.

### 5.1 Data Transmission Flow

Data transmission starts with a file source block which reads raw data values from a specified file; in this case the file name is read from a parameter block, in binary format and placed on the data bus. A packet encoder block then encodes each set of bytes into a packet of a given payload length with a header. These packets are then transferred to the GMSK mod block, a hierarchical block for GMSK modulation. The input is the byte stream and the output is a complex modulated signal at baseband. To avoid saturating the USRP input, the baseband signal is fed through a multiply const where all the complex values are halved. The USRP sink block reads the incoming stream, shifts it from baseband to the specified centre frequency, and finally transmits the stream over the air through the USRP specified by its IP address.

### 5.2 Data Receiver Flow

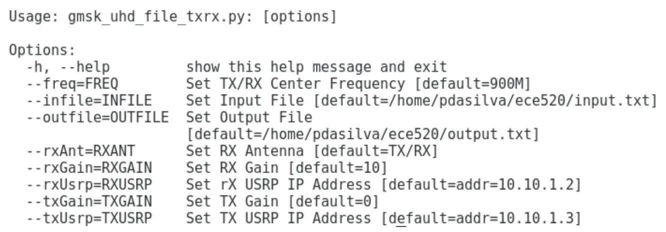
On the receiving end, the wireless transmission is sampled by the N210 and samples are fed over the network to the USRP source block. The block commands the radio specified by its IP address to be tuned to the correct centre frequency. The incoming stream is output as complex values at baseband. The complex stream is fed through a low pass filter block where all frequencies above 100 kHz are filtered out. From here on out, the stream is just fed through the inverse of the transmitting flow. The complex values are passed to a GMSK demod block, a hierarchical block for GMSK demodulation. The input is the complex modulated signal at baseband. The output is a stream of bits packed 1 bit per byte (the least significant bit). The packet decoder then checks the header information for packet payload length and reconstructs the data output. Finally, the raw data stream is written to a file in the file sink block specified by a parameter.

### 5.3 Changing Parameters from Command Line

For the file transfer blocks, we have defined a few parameter blocks. A parameter block represents a parameter to the flowgraph. A parameter can be used to pass command line arguments into a top block or instantiated hierarchical block.

Parameters blocks need a label, value, and type to be defined. The label presents a descriptive label when using the help from the python command line. To use the parameter ID as the label, leave this blank. The value is the default value for the parameter. The type is the data type of the resulting variable used in the flowgraph.

The parameter value cannot depend on any variables. When type is not None, this parameter also becomes a command line option of the form: -[short_id] --[id] [value]. The short ID field may be left blank. In this case, parameters can be fed from the command line as seen in Fig. 8.



```
Usage: gmsk_uhd_file_txrx.py: [options]

Options:
  -h, --help         show this help message and exit
  --freq=FREQ        Set TX/RX Center Frequency [default=900M]
  --infile=INFILE    Set Input File [default=/home/pdasilva/ece520/input.txt]
  --outfile=OUTFILE  Set Output File
                     [default=/home/pdasilva/ece520/output.txt]
  --rxAnt=RXANT      Set RX Antenna [default=TX/RX]
  --rxGain=RXGAIN    Set RX Gain [default=10]
  --rxUsrp=RXUSRP    Set rX USRP IP Address [default=addr=10.10.1.2]
  --txGain=TXGAIN    Set TX Gain [default=0]
  --txUsrp=TXUSRP    Set TX USRP IP Address [default=addr=10.10.1.3]
```

**Figure 8.** Changing the parameters from the command line.

## 5.4 File Transfer Plots

As shown in Figs. 9 and 10, the constellation diagrams recognize the reduction in signal amplitude prior to transmitting after modulation.

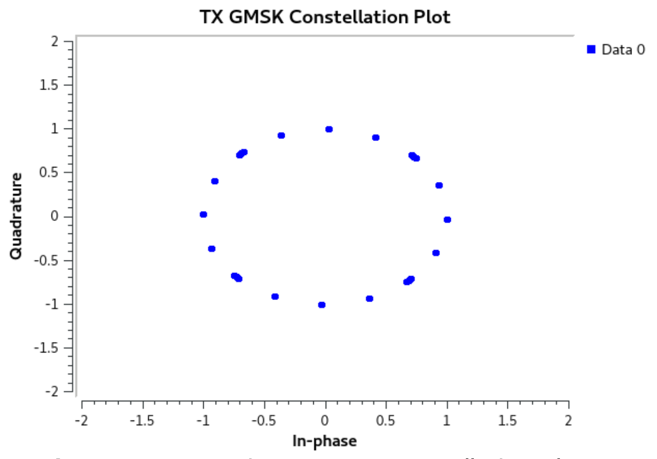


**Figure 9.** Transmitter GMSK constellation plot.

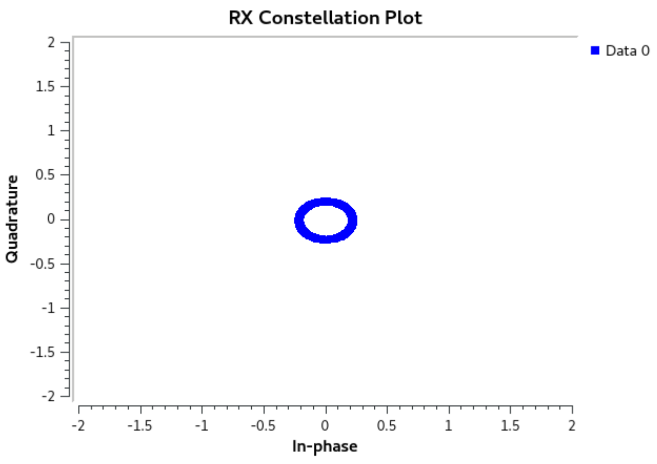


**Figure 10.** Receiver constellation plot.

Figs. 11 and 12 show the frequency plots at the transmitter and receiver ends while the file is being transferred.

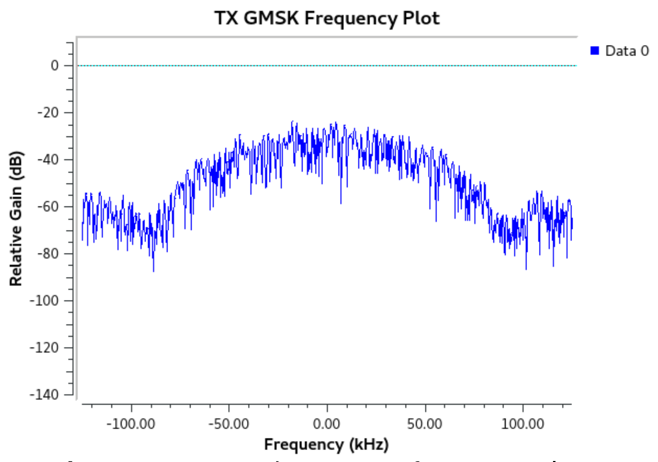


**Figure 11.** Transmitter GMSK frequency plot.

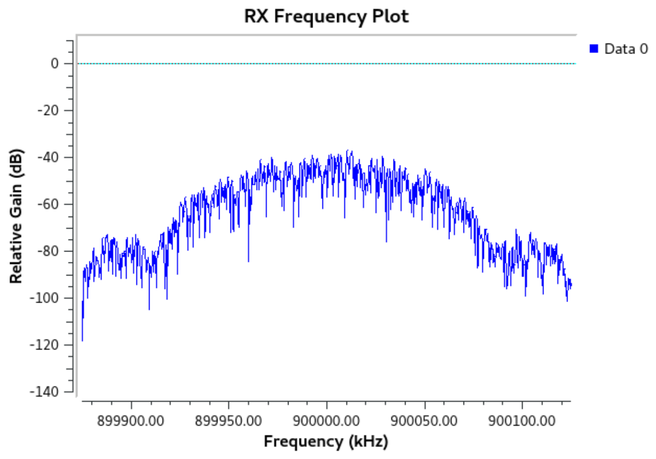


**Figure 12.** Receiver frequency plot.

## 6. Audio-Visual Media Streaming

Audio-visual media streaming can be implemented with GNU Radio and VLC using UDP (Fig. 13). VLC's primary job is to transcode the audio/video on either ends while GNU Radio takes care of wireless transmission with USRP. Gstreamer and mplayer could have been used for transcoding instead [8]–[10], but VLC was easier to setup and the research showed that VLC had not been used by a lot of people. A Linux pipe instead of UDP would have been used to send data into GNU Radio from Gstreamer or mplayer and opened in GNU Radio using the file source and sink blocks instead of UDP.

Both the continuous file transfer and the audio-visual media streaming are fundamentally the same, with the exception that file transfer uses a file sink/source whereas the audio/visual media streaming uses a UDP sink/source in conjunction with VLC to be able to transcode the media properly. The main difference between transmitting a file and a video is a video takes up more bandwidth.

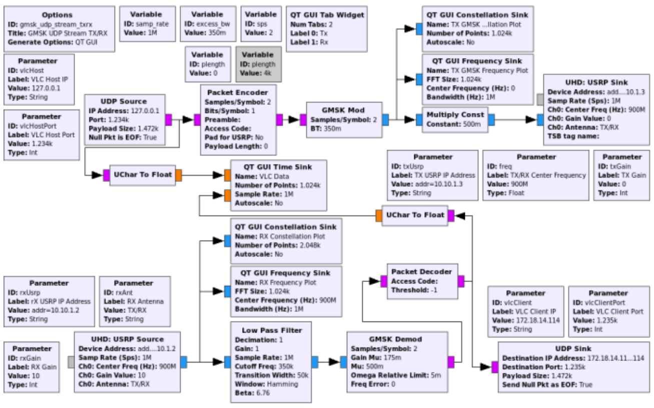


**Figure 13.** GRC Flowgraph of audio-visual media streaming.

## 6.1 Host Machine Steps

On the host machine, first we launched the GNU Radio Python script making sure to set the parameter vlcClient (Fig. 14) to the IP address of the client destined to receive the unicast stream. Then we launched VLC and set it up to stream over the network using UDP. To direct the network stream to GNU Radio, we set the IP address to 127.0.0.1 port 1234. When transcoding the media, in this case an MP4 video file, we reduced the audio and video bitrate. For this project, the audio codec was set to MPEG at a bitrate of 48kB/s, 2 channels, and a 48kHz sample rate. The video codec was set to H-264, bitrate to 120 kB/s, and a frame rate to 15 frames per second.

```
Usage: gmsk_udp_stream_txrx.py: [options]

Options:
  -h, --help            show this help message and exit
  --freq=FREQ           Set TX/RX Center Frequency [default=900M]
  --rxAnt=RXANT         Set RX Antenna [default=TX/RX]
  --rxGain=RXGAIN       Set RX Gain [default=10]
  --rxUsrp=RXUSRP       Set rX USRP IP Address [default=addr=10.10.1.2]
  --txGain=TXGAIN       Set TX Gain [default=0]
  --txUsrp=TXUSRP       Set TX USRP IP Address [default=addr=10.10.1.3]
  --vlcClient=VLCCLIENT
                        Set VLC Client IP [default=172.18.14.114]
  --vlcClientPort=VLCCLIENTPORT
                        Set VLC Client Port [default=1235]
  --vlcHost=VLCHOST     Set VLC Host IP [default=127.0.0.1]
  --vlcHostPort=VLCHOSTPORT
                        Set VLC Host Port [default=1234]
```

**Figure 14.** Audio-visual media streaming command line options.

## 6.2 Client Machine Steps

On the client machine or mobile phone, we set VLC to open a local UDP network stream on port 1235 with the URL udp://@:1235 and the stream began playing.

Upon experiencing video packet drop, we continued to reduce the audio and video codec bitrates. The low bitrates can be attributed to multiple reasons, including the host server's available resources and the capability of the GMSK implementation in GNU Radio.

## 6.3 Audio-Visual Media Streaming Plots

Transmitting wireless video can cause an enormous amount of delay attributed to many factors. This delay can be seen when overlaying the received and the transmitted data on top one another. If no delay was present, there would only be one line. This can be seen in Fig. 15.

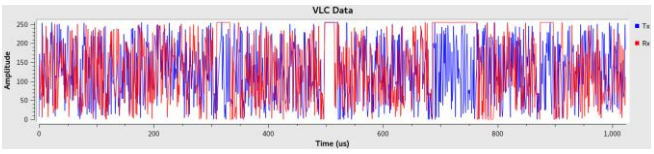


**Figure 15.** Media transmission and receive data.

## 7. Discussion on Applications

This testbed could be used for testing multimedia over wireless network applications. One application is a QoE-driven cross-layer design for CR multimedia networks. A case study of cross-layer approach has been conducted in our previous work [13], [14] for second users (SUs) that have different QoE requirements. In multimedia communications, there are many types of traffic, such as voice, image, and video (i.e., variable bitrate (VBR), constant bitrate (CBR), etc.). Some of them (e.g., voice and CBR video) are delay-sensitive, while others may not be sensitive to delay. The delay and multimedia quality can be regarded as the most important QoE requirements in these applications. In wireless CR systems, the frequent channel switching may seriously cause the transmission delay. Our studies in [13] show that sub-bands with smaller switching probability are allocated to the SUs that carry delay-sensitive multimedia traffic to improve the QoE of these SUs. We assume that there are totally k types of SUs, and are sorted from type-1 with the highest priority to type-k with the lowest priority. The type-k SUs carry non delay-sensitive multimedia traffic which is more tolerant of frequent channel switching. The channel allocation scheme needs to maintain a well-organized queuing model under different events including the arrival/leaving of any kinds of PUs or SUs. The evaluation in the case study was based on wireless fading channel models and used only a few video clips. There are challenges in capturing the dynamics of channels and behaviour of video events in complex wireless video systems. The testbed will serve as a perfect platform for the validation. The other applications could include "Personalized Health Monitoring for Multimedia eHealth". This application is to provide the means of applying CR technologies for the multimedia eHealth to mitigate the communication interferences mentioned above. The potential research objectives are to: 1) determine effective learning strategies for MBANs that allow dynamic selection of spectrum frequencies and transmission power, and, therefore mitigate interference to PU receivers, and 2) provide reliable end-to-end multimedia health data delivery in a CR network environment. The success of the project requires a qualitative CR testbed in an integrated CR communication infrastructure for multimedia eHealth transmission, algorithm design, protocol analysis and development, and experimental data set analysis. The third application could be “Networked Distributed Source Coding (DSC) for Multimedia Sensor Networks”. For DSC to be widely used in wireless multimedia sensor networks, significant research challenges still remain and are itemized below: 1) It is difficult to accurately estimate the spatial and temporal correlation in wireless multimedia sensor environments when multimedia data exchange between two correlated sensors is very difficult in real applications; 2) The lack of a framework for modelling multimedia transmission quality for DSC makes it hard to find the trade-offs between the achieved quality and the resulting energy consumption; 3) Various rate allocations of DSC lead to different levels of decoding errors in addition to the transmission errors caused by the wireless channel. To address the problems, we consider a multimedia transmission quality maximization problem with the energy consumption bound. In this optimization framework, resource allocation control parameters such as Automatic Repeat Request (ARQ), power and modulation interact with DSC control to achieve the overall high QoE and resource efficient multimedia delivery. Thus, allocating lower rate and more retry transmissions to side information packets and higher rate and less retry transmissions to those unimportant packets including value information can achieve an ideal unequal resource protection paradigm. The variable coding rates can be adjusted with the corresponding correlations among multiple correlated nodes. However, it is difficult to adjust major resource control parameters such as ARQ, power and modulation interacting with DSC control to achieve the overall high QoE multimedia delivery in current multimedia sensor settings. The proposed testbed can provide capability of the software defined radio (USRP), where the resource control can be easily implemented.

## 8. Conclusions

We successfully continuously transmitted and received the contents of a text file. Also, we were able to complete our stretch goal of streaming audio-visual media utilizing the same flowgraph in GNU Radio.

Since the USRPs (transmitter and the receiver) do not synchronize instantaneously, the file source block in the file transfer is set to repeat the contents of the file being read. This allows all the input file contents to be transferred at the cost of it repeating at the output file.

A follow-on approach to transferring a file would be to work on synchronizing the two radios so the file need only be sent once like [11]. An improvement to this would be send information prior to the transfer to allow the receiver to save the file under the correct filename and extension.

Another takeaway was with the media streaming. As expected video streaming takes much more bandwidth than audio streaming. There are multiple reasons for this including quality of the video and audio being streamed simultaneously taking up more bandwidth.

When video files are streamed, the video may lag, which is an expected outcome of streaming anything wirelessly. The ultimate way to stream anything using this implementation is to have the transmitting USRP and GNU Radio transmit flowgraph on its own host computer. This allows the host to only worry about prepping the video stream for wireless transmission without the overhead of also receiving the transmission and packaging it onto the network. Multicast can also work with this implementation so long as the receiving network has a broadcast IP address setup. We believe the proposed testbed will potentially benefit the research community for the study of various multimedia applications over networks.

### Acknowledgements.

The equipment used in the proposal is supported by NSF MRI grant (award# 1429120).

## References

[1] M. Rouse, "Tech Target: Cognitive Radio Definition," November 2008. [Online]. Available: http://searchnetworking.techtarget.com/definition/cognitive-radio.

[2] "Wikipedia: Software-defined radio," 30 September 2017. [Online]. Available: https://en.wikipedia.org/wiki/Software-defined_radio.

[3] Ettus Research. [Online]. Available: https://www.ettus.com/. [Accessed 8 December 2017].

[4] “About USRP Bandwidths and Sampling Rates,” Ettus Knowledge Base, 16 May 2016. [Online]. Available: https://kb.ettus.com/About_USRP_Bandwidths_and_Sampling_Rates. [Accessed 8 December 2017].

[5] GNU Radio, "Guided GNU Radio Tutorials," 5 July 2017. [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorials. [Accessed 7 December 2017].

[6] Ettus Research, "USRP Hardware Driver and USRP Manual," [Online]. Available: http://files.ettus.com/manual/page_usrp2.html.

[7] Z. Chen and J. Xu, "Cross-layer wireless video testbed," [Online]. Available: http://www.wu.ece.ufl.edu/projects/wirelessVideo/project/H264_USRP/index.htm. [Accessed 8 December 2017].

[8] A. M. Iype and S. C. D., "Video Transmission Using USRP," 16 September 2016. [Online]. Available: http://academic.csuohio.edu/yuc/mobile/mcproj/z_Shashanka_Asha.pdf. [Accessed 8 December 2017].

[9] S. Nimmi, V. Saranya and Theerthadas, "Real-time video streaming using GStreamer in GNU Radio platform," in 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), Coimbatore, India, 2014.

[10] A. Singh, S. Rani and S. Kakkar, "Video Transmission through GMSK using GNU Radio," in IJCA Proceedings on International Conference on Advances in Emerging Technology, 2016.

[11] S. Jordan and B. Patel, "Image transfer and Software Defined Radio using USRP and GNU Radio," 19 December

2016. [Online]. Available: http://academic.csuohio.edu/yuc/mobile/mcproj/3d-GNU%20Radio_Steve%20&%20Bhaumil.pdf. [Accessed 8 December 2017].

[12] G. Kaur, A. Thakur and H. Kaur, "Implementation of File Transfer with GNU-RADIO Tool on SDR Platform," in International Conference on Soft Computing Applications in Wireless Communication - SCAWC 2017, Punjab, India, 2017.

[13] T. Jiang, H. Wang, and S. Leng, “Channel allocation and reallocation for cognitive radio networks,” Wireless Communications and Mobile Computing, vol. 13, no. 12, pp. 1073-1081, 2013.

[14] T. Jiang, H. Wang, A V. Vasi: QoE-Driven Channel Allocation Schemes for Multimedia Transmission of Priority-Based Secondary Users over Cognitive Radio Networks. IEEE Journal on Selected Areas in Communications30(7): 1215-1224 (2012)