HELWAN UNIVERSITY

Department:
Electronics &
Communications.

Faculty of Engineering

B.Sc. Graduation Project

Title:

Multiuser Cognitive Radio (CR) Communications System,

(SDR- Implementations)

prepared by

Project Group

1- Mohamed Khaled

2-Omar Ezz-ElDeen Sayed

3-Mohamed Omar

4-Ahmed Saber

5- Essam Bahaa-ElDeen

Supervisor: Prof. Dr./ Gamal Abdel-Fadeel

May 2021

Thanks and appreciation

First of all we thank god for the most mercy for enabling us to present this project in the best form that we wanted to be, we would like to thank our supervisor of this project, **Prof. Dr./ Gamal Abdel-Fadeel** for his valuable help and advice to come out with this project. also, your office was open for us to come whenever we had a problem, thank you very much for your effort. also, we thank our faculty and teaching assistants that provided us with all the knowledge.

Thank you very much for your effort.

Project Group

1 Abstract

Cognitive Radio (CR) has evolved as a smart technology in bridging the disparity between the availability and allocation of the radio frequency spectrum amongst multiple users. With a Software Defined Radio (SDR) framework, Cognitive Radio aims to target high spectrum utilization efficiency via opportunistic spectrum access through autonomous adaptation of its communication parameters according to the network and the demands of the users. This paper presents an overview of the Cognitive Radio where can be done by using Software Defined Radio (SDR) where it can be applied, implemented, and programmed using GNU or LabVIEW. We use in programming some software languages such as C, C++, and Python. So in this project, we are trying to implement cognitive radio in two directions (LabVIEW and GNU) where a logical approach is taken to implement our CR in both directions.

In LabVIEW:

1. we design a band scanning or monitoring system that we use in order to be able to find the different PUs and Holes in this band.
2. we design a system to transmit and receive packets of any messages by using BPSK or QPSK or OPSK or DQPSK or 16-PSK or 8-PSK Modulation and Demodulation schemes and using Raised Cosine or Root Raised Cosine or Gaussian Filter.
3. we design a system that takes the reads of the scanning system to make a decision to choose the best available carrier frequency to start the transmission by the transmitter.

4. we design the Tx that transmit a sinusoidal signal to use as a PU in our testbed.

In GNU:

1. Design of PU based on OFDM Modulation
2. Design a spectrum sensing
3. Design a common channel

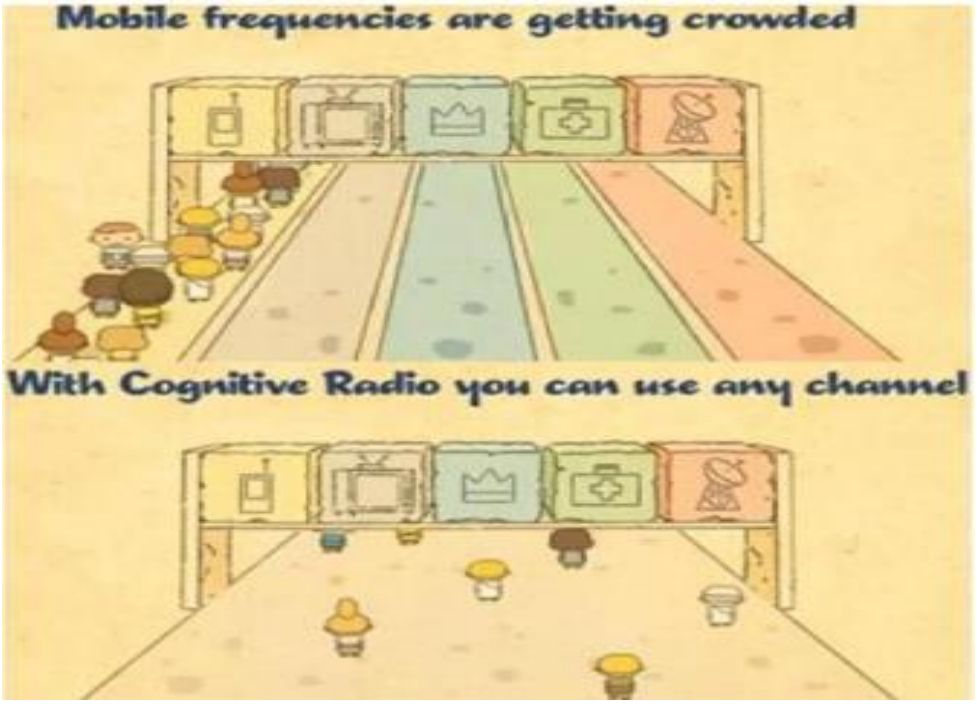


Figure 1 Cognitive Radio concept

Key Words:

Cognitive Radio (CR), Spectrum Sensing Techniques, Spectrum Monitoring, Frequency Hopping, Band Hopping, SDR Architecture, LabVIEW, GNU Radio, NI USRPs.

2 Table of contents

Contents

3 List of illustrations.

4 Introduction

The ever-growing demand in modern lives and businesses for new wireless services and applications for ubiquitous computing causes a scarcity of wireless spectrum available. The spectrum allocation is achieved by assigning the spectrum completely to a licensed user in the conventional approach, and the system has to work within that frequency range. This contributes to inefficient spectrum usage, because much of the time, a significant portion of the bands remain underused. The cognitive radio definition offers a revolutionary strategy for handling this spectrum. It provides new techniques for the utilization of the spectrum available. The key feature of cognitive radio is to take advantage of the unused spectrum to provide a new route of spectrum access. However, the two key issues impede the creation of cognitive radio. The first and most fundamental issue is the fear of primary users (PUs) regarding possible interference from secondary users (SUs) communications. If licensed users enable unlicensed users (or SUs) to enter their licensed spectrum bands in an opportunistic way, there would be a chance of intrusion from SU communications. Such intervention can weaken the QoS of the PUs, which may lead to a loss of the PU business. SUs must also have effective mechanisms to secure communications from PUs in the identification of spectrum holes by spectrum sensing. The second problem is the lack of economic incentives for spectrum sharing among PUs.

Cognitive Radio (CR) is an intelligent wireless communication system that understand the environment around him by understanding-from-learning.

CR technique is able to solve the problem of spectrum scarcity by applying the optimistic spectrum sharing techniques. In general words, CR can discover and sense Dynamically changing environments, as CR once senses any change in

the surrounding environment, such as carrier frequency, bandwidth, power, coding, schemes, and modulation.

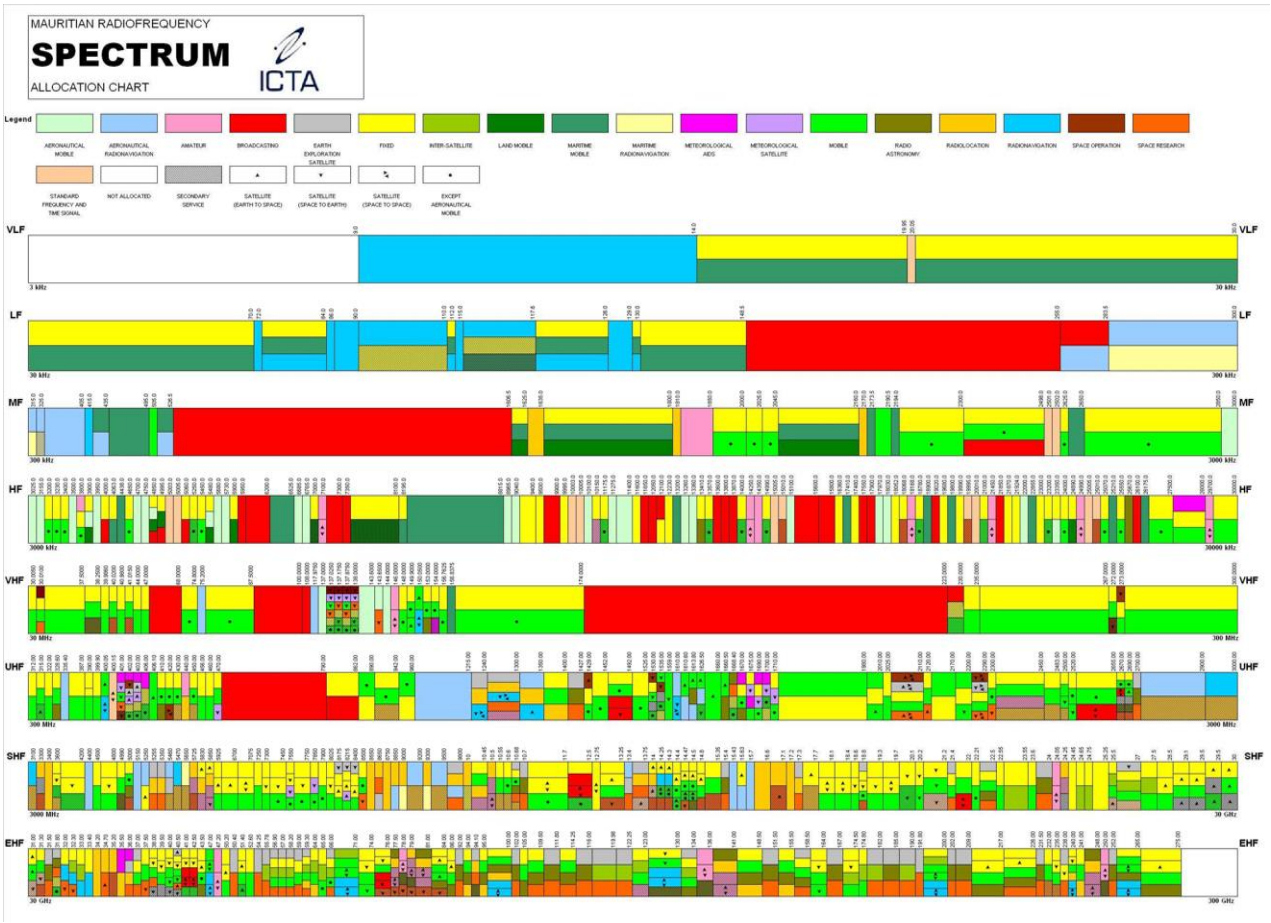


FIGURE 3 spectrum distribution Figure 2

TABEL 1. Advantages, challenges of CR

Advantages	Disadvantages
1. Overcome radio spectrum scarcity. **2. Avoid jamming.** **3. Saving power protocol.** **4. Improve QOS as CR improve SNR.**	1. Required more information about PU. 2. High interference of channel. 3. Difficult to make synchronization between Tx & Rx.

5 Spectrum Sensing.

5.1 Introduction

The basic condition of the CR network is that unlicensed users should be able to identify licensed users and enter unused licensed bands, but that when the licensed user enters its CR band, space should be vacated instantaneously to counteract interference effects. One of CR's main tasks is to detect the spectrum gaps in the licensed band. New spectrum sensing methods are emerging day by day, and many researchers are seeking to find the most reliable systems for CR networks. It has always been a tiresome job to define the channel between the authorized (PU) user transmitter and the cognitive user. Table 1, summarises basic challenges of CR .

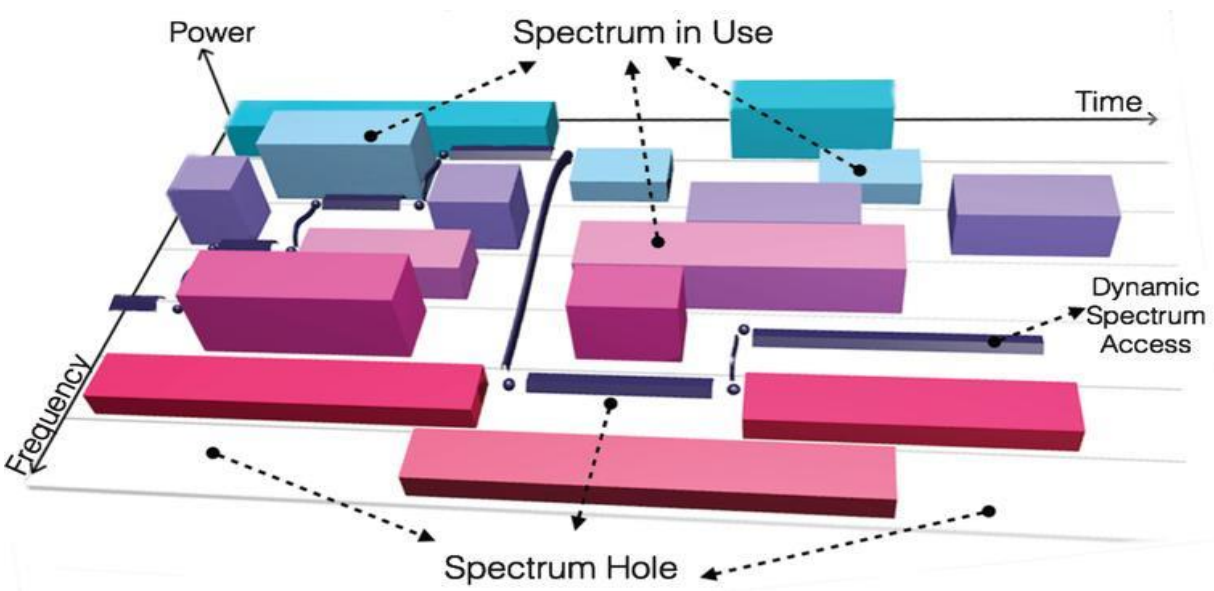


Figure 4 Dynamic spectrum acces

5.2 Spectrum sensing techniques.

Spectrum sensing methods are grouped into three types of transmitter detection, cooperative detection, and interference-based detection as seen in the figure (4). The key function of the transmitter detection technique is to analyze the received signal. Transmitter detection is also known as the non-cooperative detection technique. This scheme is further divided into three groups, Matched Filter Detection (MFD), Energy Detection (ED), and Cyclostationary Function Detection (CFD).

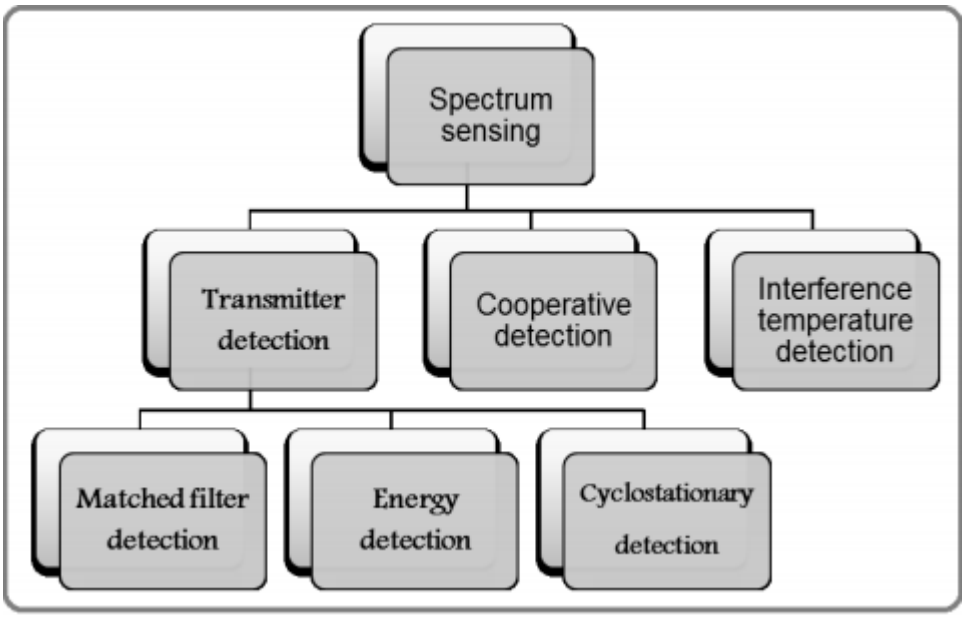


Figure 5 Classification of Spectrum Sensing techniques

6 Path loss problem & its solution.

6.1 Introduction.

Path loss, or path attenuation, is the reduction in power density (attenuation) of an electromagnetic wave as it propagates through space. Path loss is a major component in the analysis and design of the link budget of a telecommunication system. This term is commonly used in wireless communications and signal

propagation. Path loss may be due to many effects, such as free-space loss, refraction, diffraction, reflection, aperture-medium coupling loss, and absorption. Path loss is also influenced by terrain contours, environment (urban or rural, vegetation and foliage), propagation medium (dry or moist air), the distance between the transmitter and the receiver, and the height and location of antennas.

6.2 The problem caused by the path loss of CR.

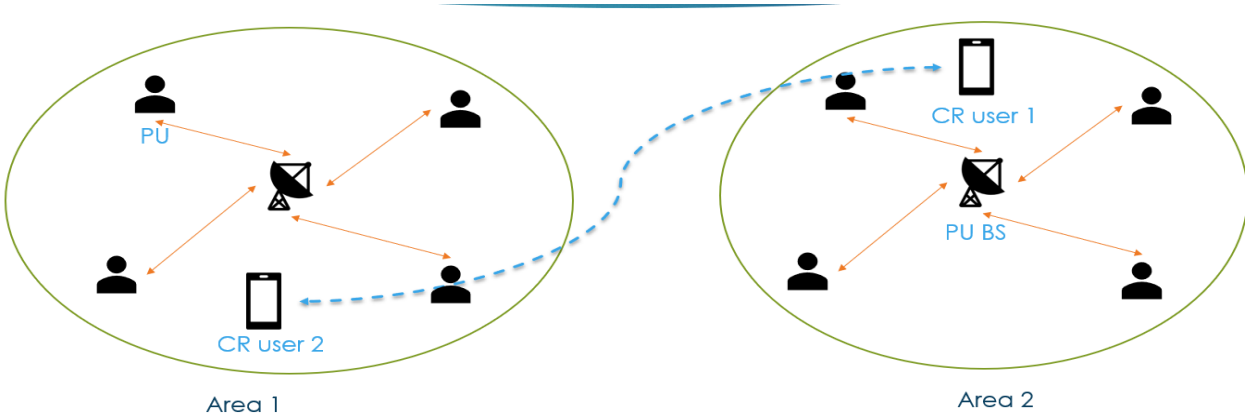
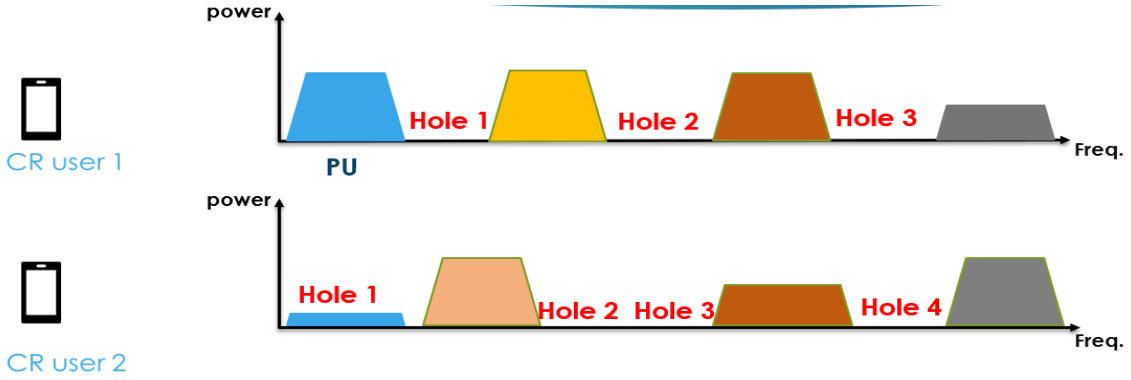


Figure 6 the long distance between the CR users



7 Figure: hole's of every user

We can see the PU that turn into hole in second user range and that because of attenuation and noise that appear with along distance.

6.3 The solution.

To solve this problem we discovered two efficient solutions which are:

- Use a common eNodeB.
- Use a common channel which is some sub-carriers that's assigned to CR users by ISP or Providers like Vodafone or Orange or Etisalat.

We discovered the first solution is very complex and expensive but more efficient and the second solution is less complex and cheaper than the first solution so we Therefore, we here into this project try to implement the second solution which is "common channel".

6.4 Use a common channel

Use a common channel to solve this problem, in this case we choose two subcarriers to create a downlink & uplink channel, for example we choose 712 MHz as uplink common channel & 715 MHz as downlink channel, the steps of the solution are:

- Sense holes in range and we select 2.4 – 2.42 GHz
- Collect holes in array and send array to common channel
- Common channels collect all arrays from all users in a network and compare it to gain a common hole array
- Broadcast common hole array for all user in network to use common hole to communicate to each other

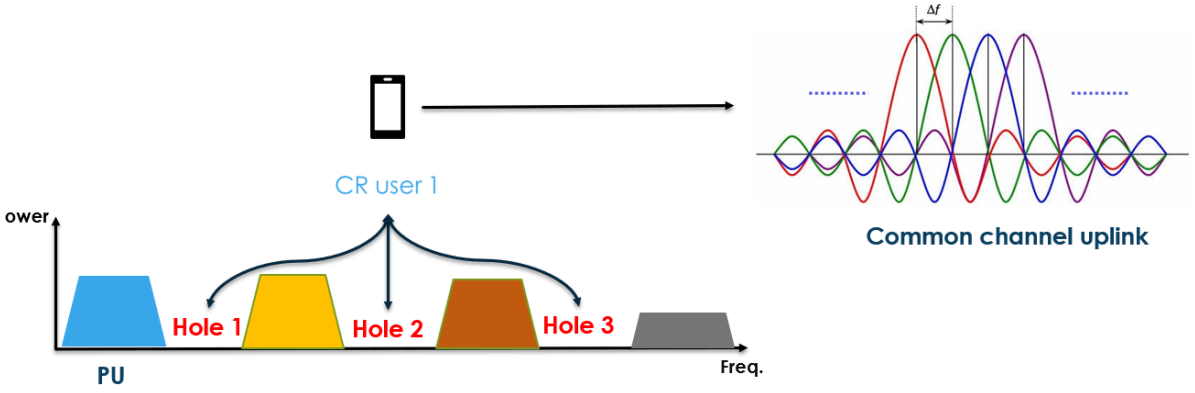


Figure 8 collect holes and send to common channel

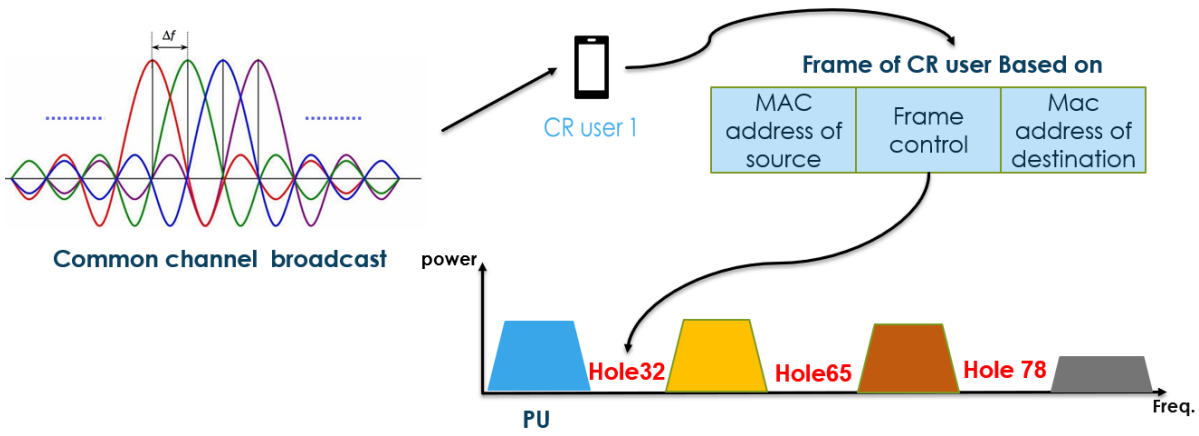


Figure 9: broadcast a common hole and select a random hole to communicate

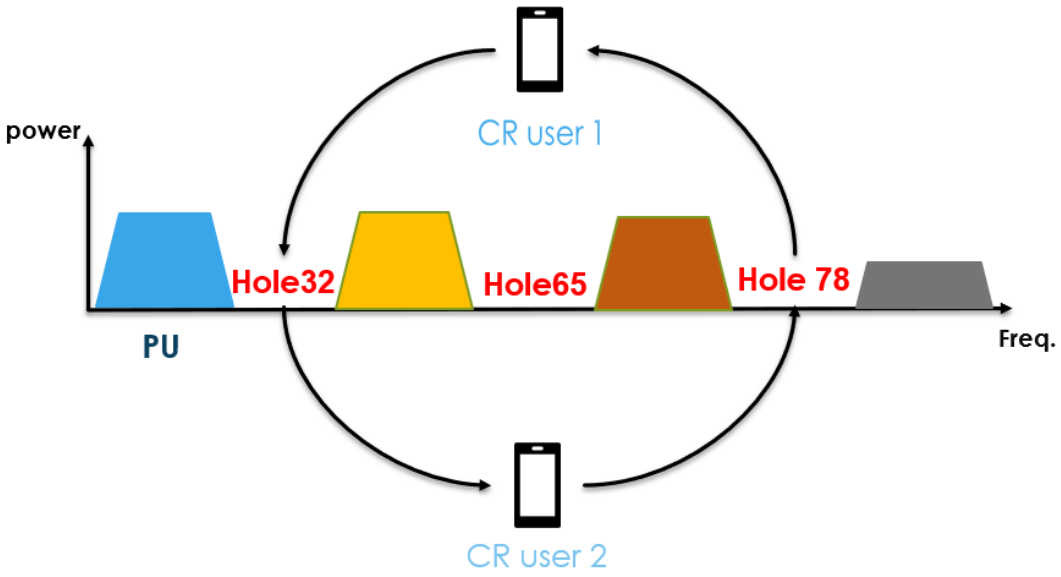


Figure 10 second user reply in another random hole

7 CR-Communication Flow Chart

Whenever a PU is not active, its (temporarily unoccupied) band is considered a free spectrum hole (SH) and hence, secondary users (unlicensed, SU) can make use of the free SH until the primary user returns back at which SU should, immediately, abandon using the hole and he/she should search for another spectrum hole to utilize.

In this project, we designed and implemented the basic functionalities of the CR-system using the SDR-basics and programming techniques. The following Figure depicts basic flow chart for the CR-communication operations

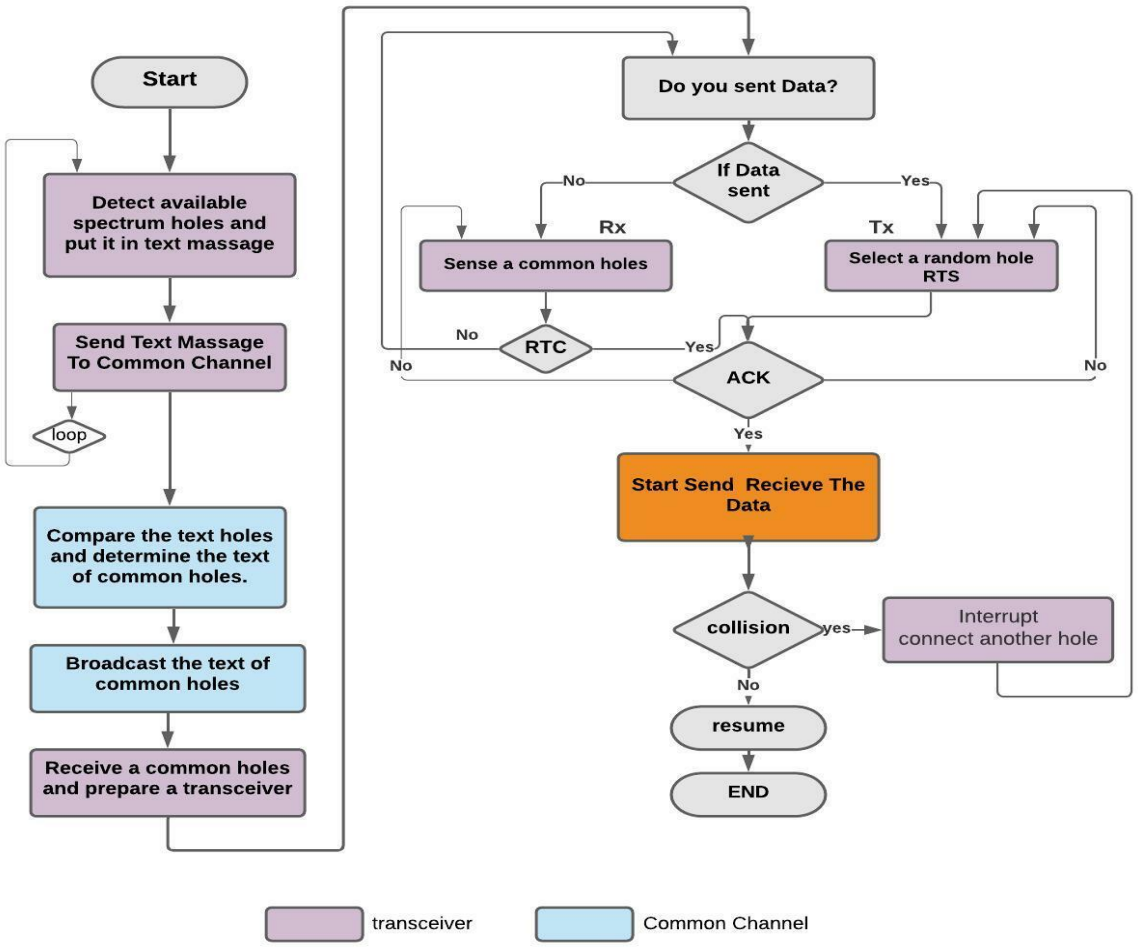


Figure 11 CR-Communication Flow Chart

The Primary users are those who have a license to operate in a specific spectrum band and when the PU does not use the band there is a free spectrum hole that secondary users can use until the primary user returns to use his spectrum again and the SUs leave that range and search in another band for another spectrum hole to access it.

All this is done by programming the CR on (Frequency & Band) Hopping and Band Selection and New Band Assignment.

8 LabVIEW Communication.

8.1 The Real Implementation & Results.

In this stage, a visual programming tool that uses visual elements in programming which is a different approach compared to text-based tools such as C, C++, Java, and MATLAB. Visual programming feature makes LabVIEW easily programmable and understandable. Because of this, LabVIEW has mostly used in SDR based studies. The most commonly used one with both is the USRP. LabVIEW easily programmable and understandable. Because of this, LabVIEW is mostly used in SDR based studies. Programming in LabVIEW is managed in two separate parts which are front panel and block diagram. In block diagram, programming is done by using elements from the functions palet and connecting them with virtual cables. Core programming structure in LabVIEW is called as virtual instrument (VI). LabVIEW contains different

function categories like programming, measurement, signal processing and data communication. With content of these functions, a communication system can be implemented in detail. LabVIEW has three element types, control, indicator and constant. With control elements, values of some functions are controlled by inserting value in the front panel. Indicator is responsible from demonstrating results in front panel. Constant elements have control element properties, but these are managed in block diagram.

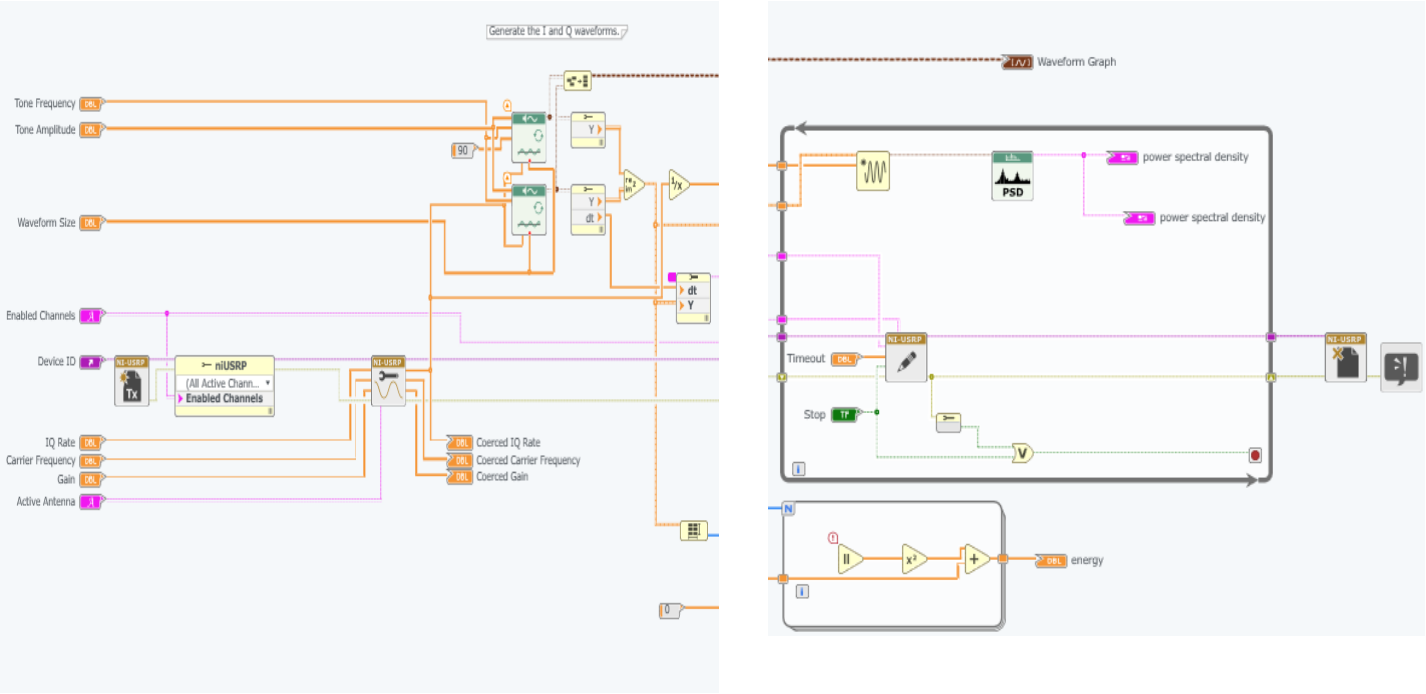


Figure 12 Tx Block diagram

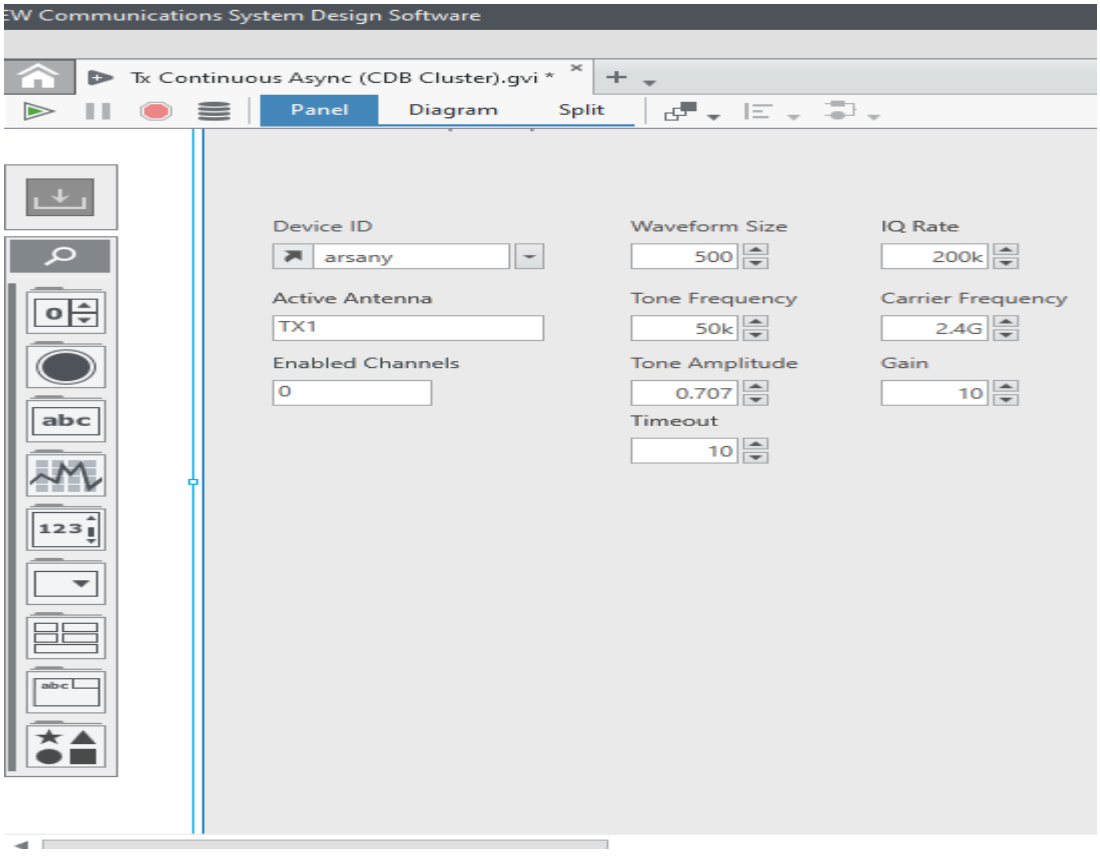


Figure 13 Tx Parameters in panel

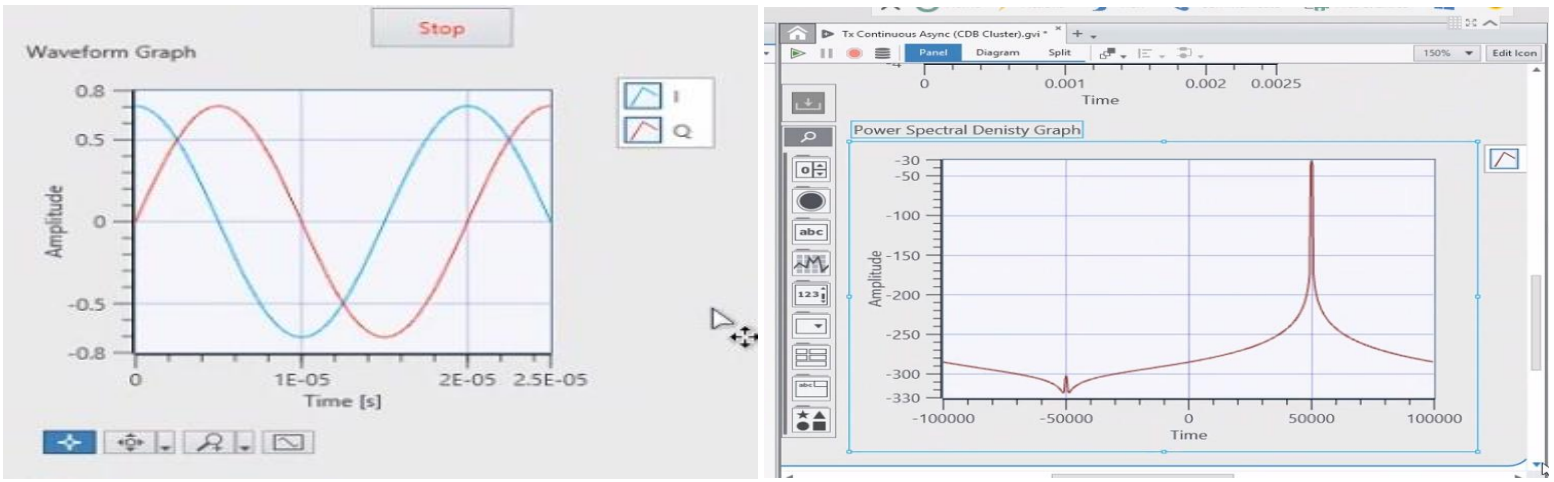


Figure 14 Tx Output

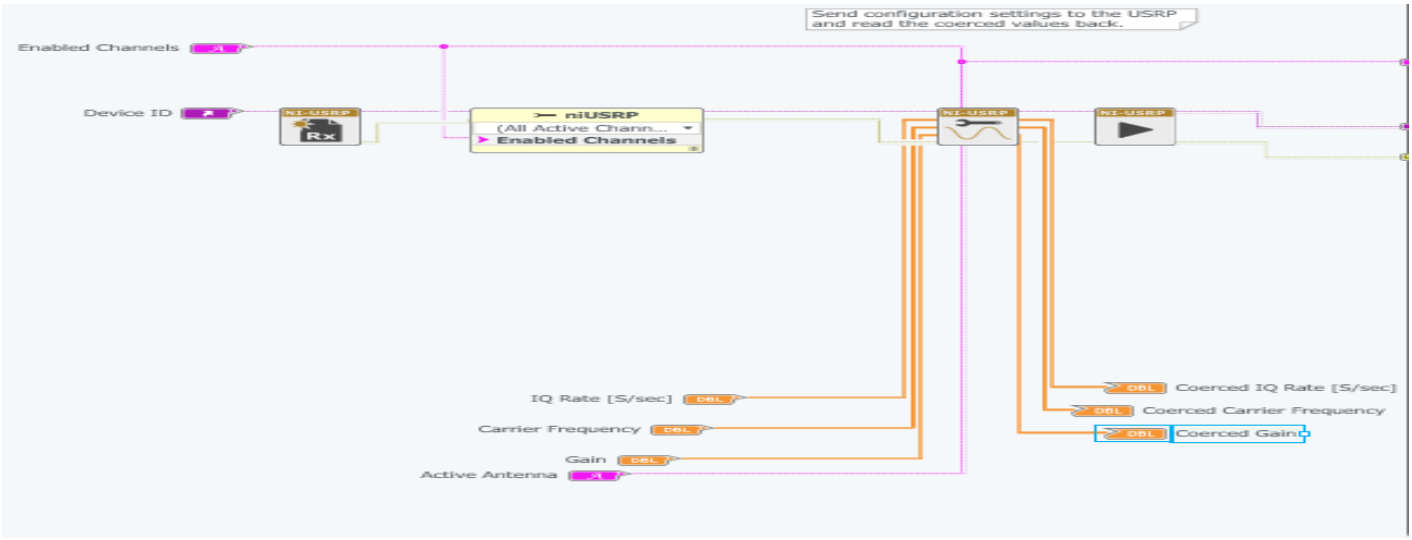


Figure 15: Rx Block diagram

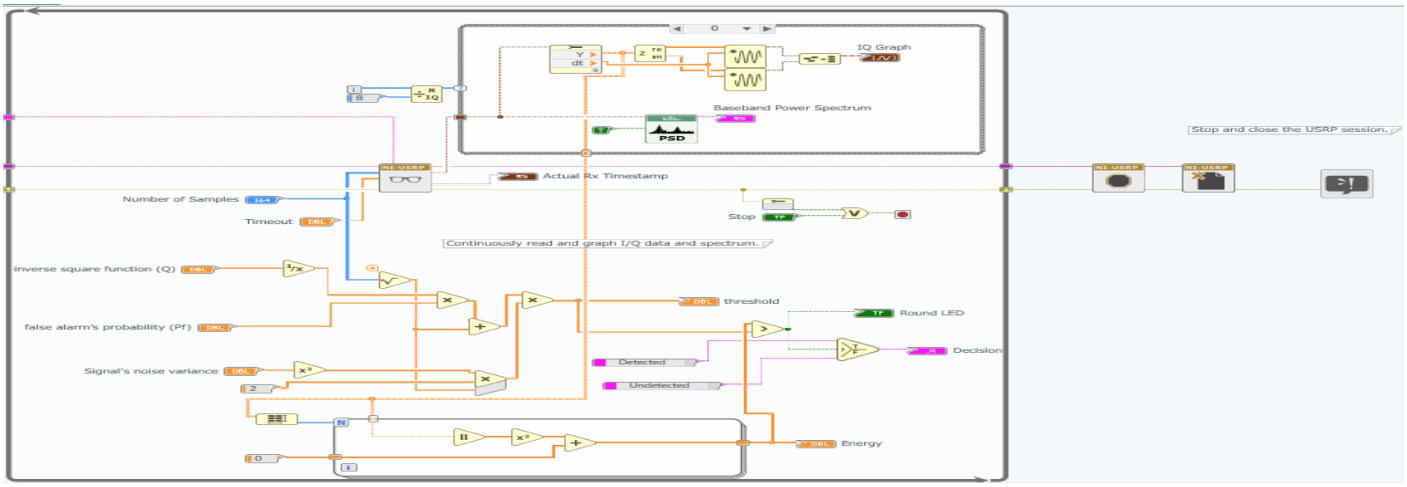


Figure 17 Rx Block diagram

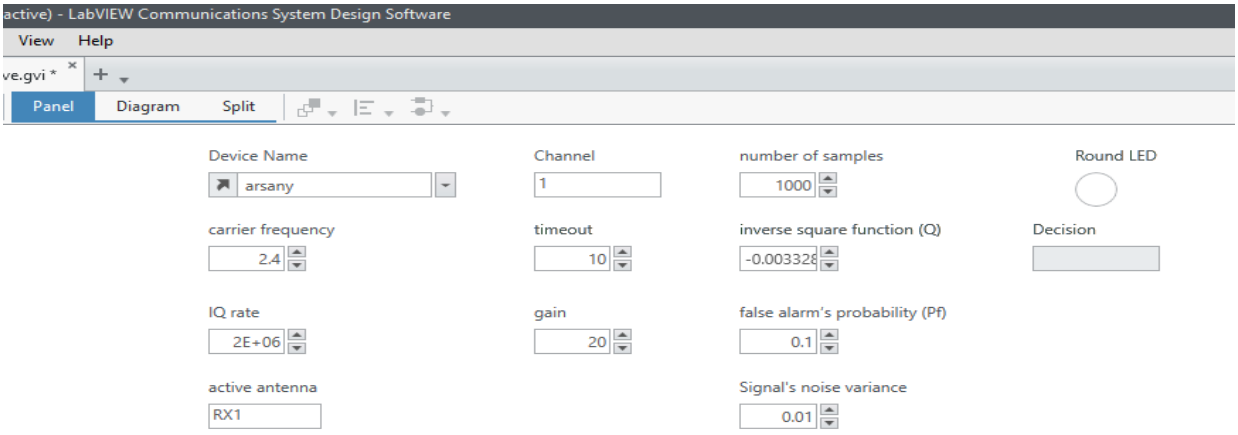


Figure 16 Rx Parameters in panel

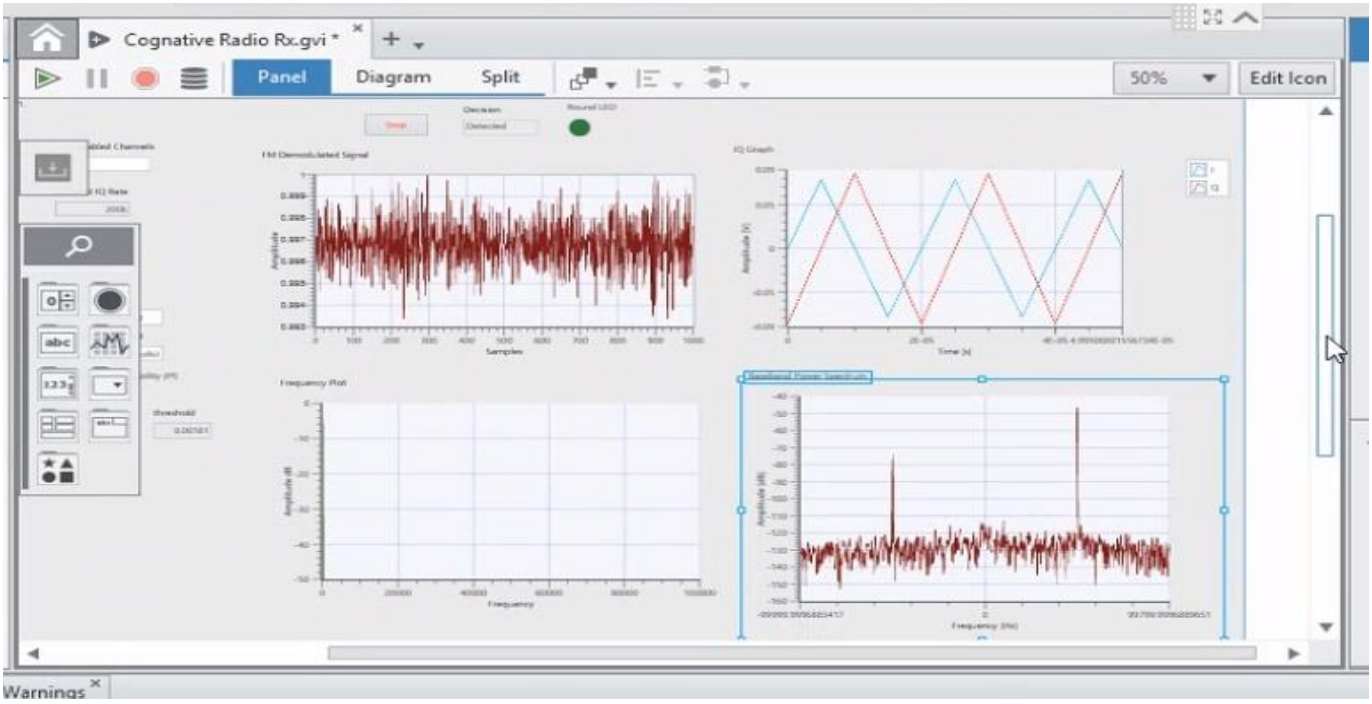


Figure 18 Output of received signal

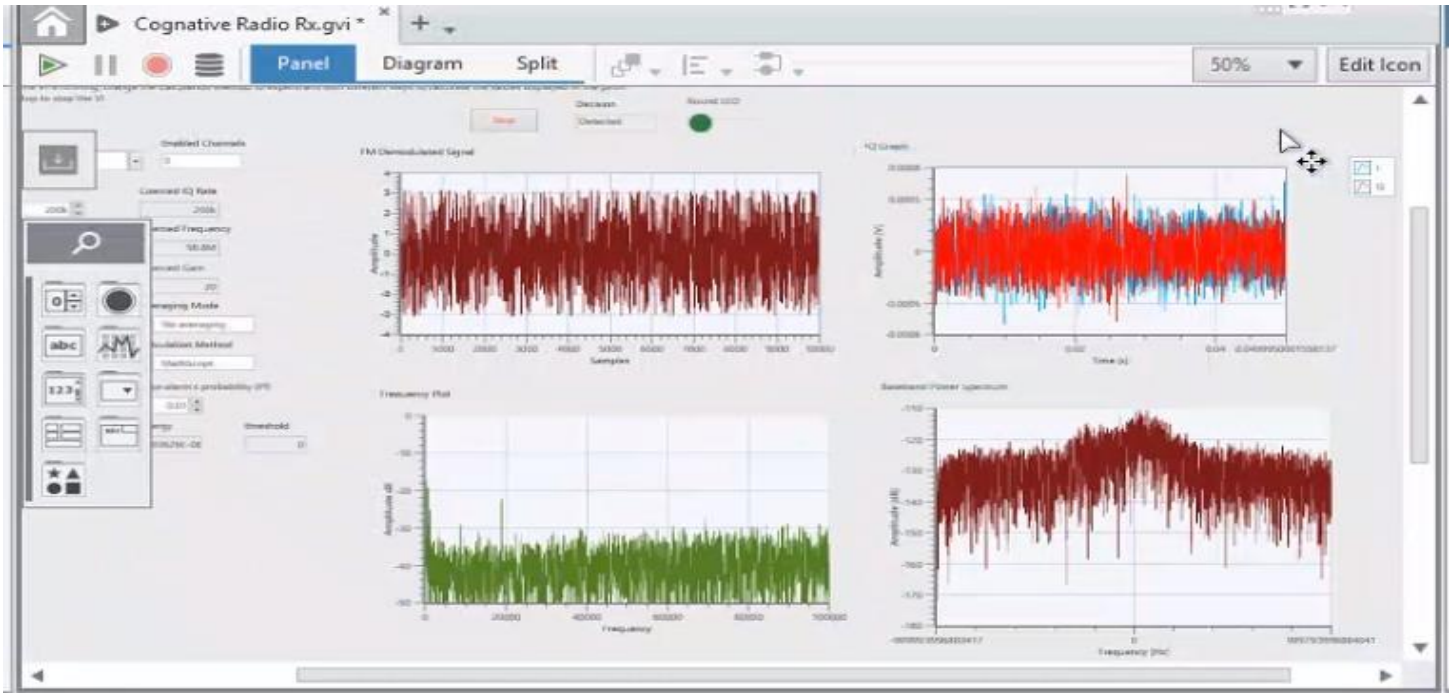


Figure 19Output of received FM radio signa

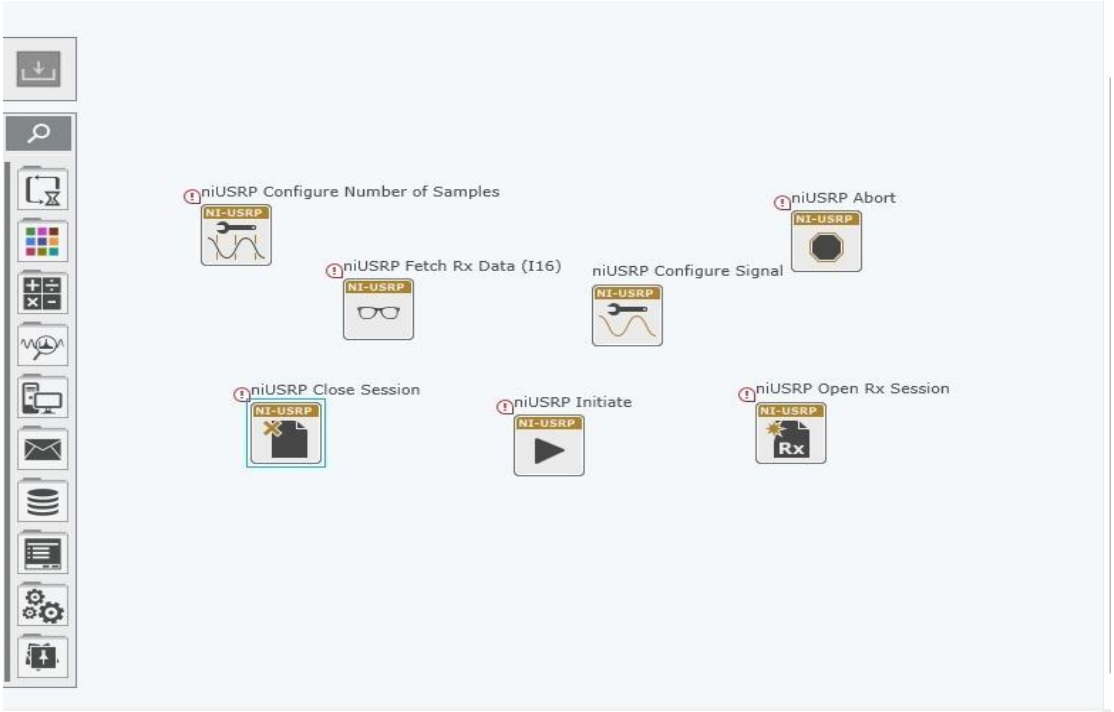


Figure 20 LabVIEW USRP library receiver VI set

Niusrp Open Rx Session: This VI is necessary to start the RF signal reception process of USRP node. In this way, USRP becomes active. **Ni USRP Configuration Property:** With this VI, USRP's RF signal reception process can be managed. Various parameters such as IQ sampling rate, carrier frequency, transmit gain, active antenna or channel configurations for usage of multiple USRPs can be configured by using this VI and RF signal that has desired characteristic can be obtained. **Niusrp Initiate:** This VI gives feedback to USRP about parameter configurations and informs that samples can be received. niUSRP Fetch Rx Data: Starts IQ data reception by USRP node and informs that IQ data can be processed. Without this VI, processing of IQ data cannot be done. **Niusrp Abort:** This VI gives close session message to USRP and new session with different parameters can be started after this message. **Niusrp Close Session:** As last component of reception set, this VI ends data reception and allocates a memory section for this session.

In this sub-VI we can set parameters that we will use in our testbed and scan a specific band by using the code in fig20 and fig21.

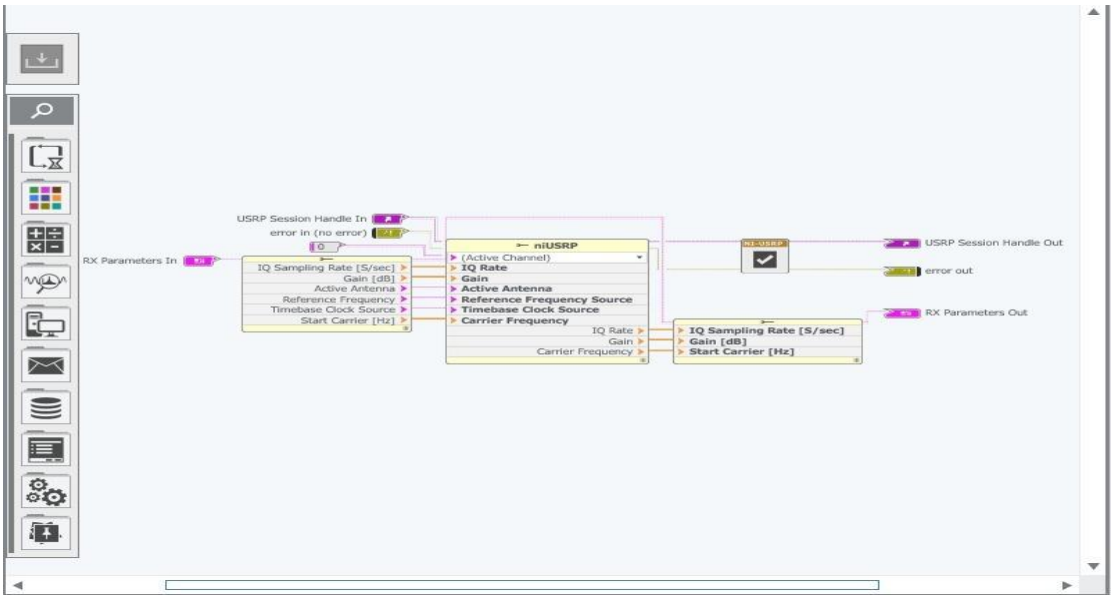


Figure 21 write Rx properties.

By this code we initiate parameters that we will use in our program such as IQ rate and gain and active antenna and start and the end of our band.

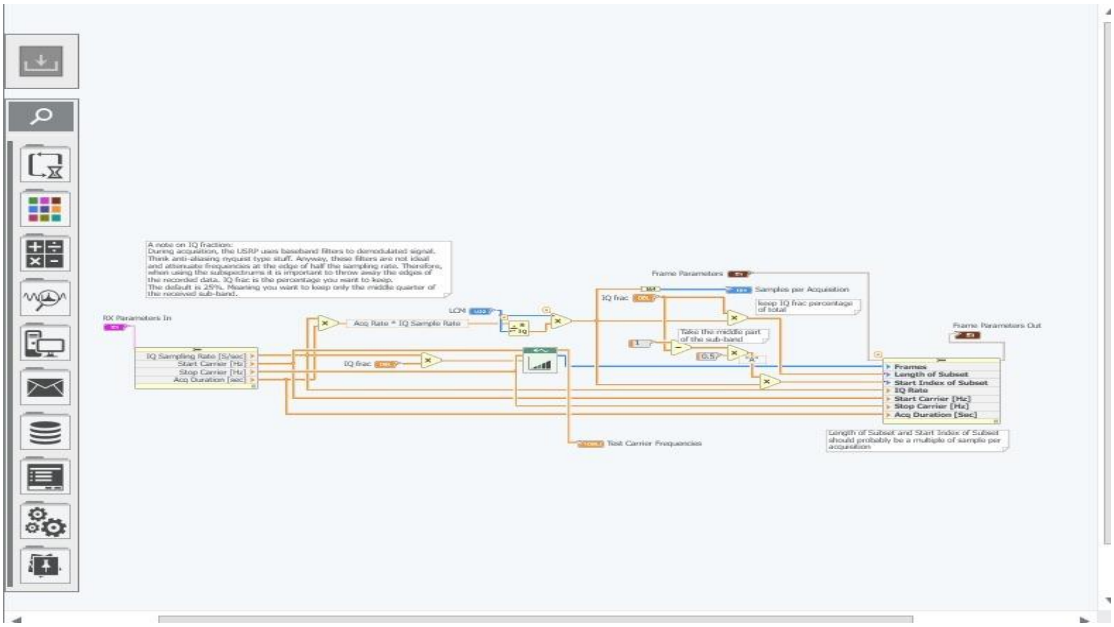


Figure 22 find carrier

During acqusition, the USRP uses baseband filters to demodulated signal.

Think anti-aliasing nyquist type stuff. Anyway, these filters are not ideal and attenuate frequencies at the edge of half the sampling rate. Therefore, when using the sub spectrums it is important to throw away the edges of the recorded data. IQ frac is the percentage you want to keep.

The default is 25%. Meaning you want to keep only the middle quarter of the received sub-band.

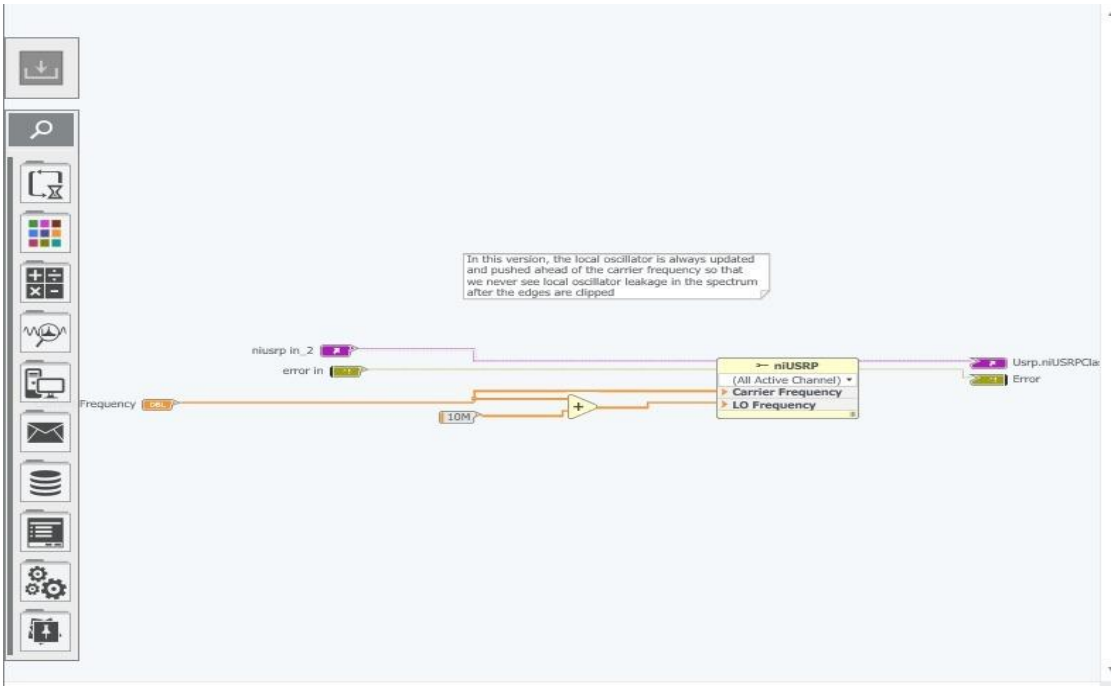


Figure 23 set carrier and local oscillator

In this version, the local oscillator is always updated and pushed ahead of the carrier frequency so that we never see local oscillator leakage in the spectrum after the edges are clipped.

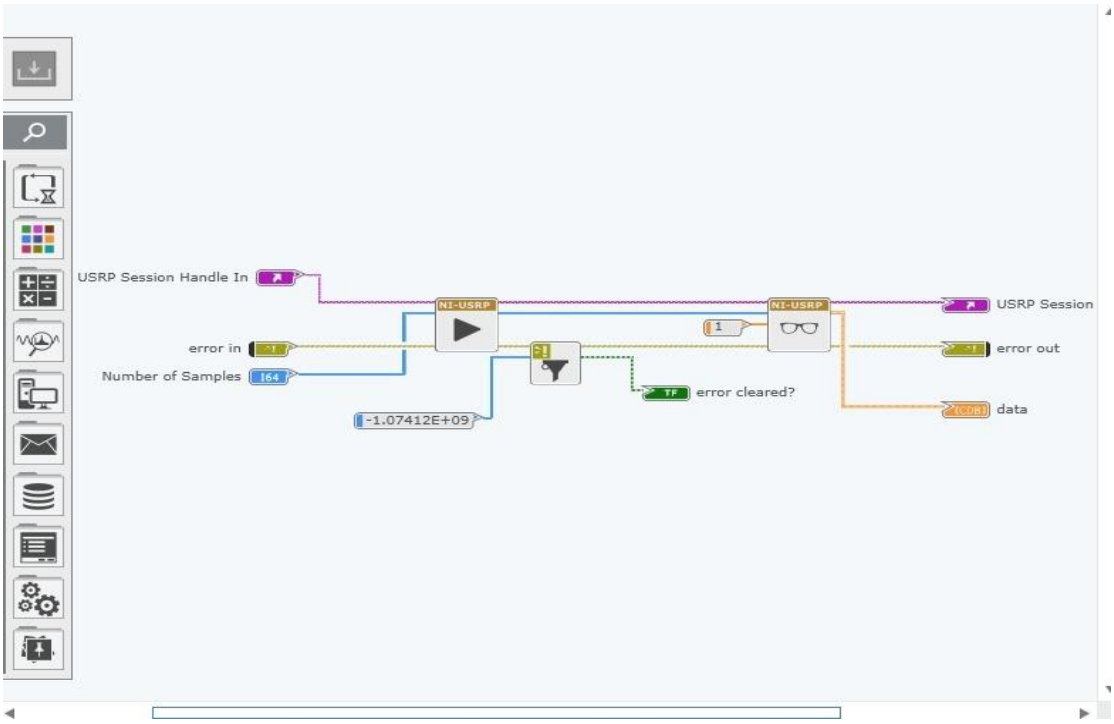


Figure 24 read data code

By using this code we can read all receiving data that received by our USRP.

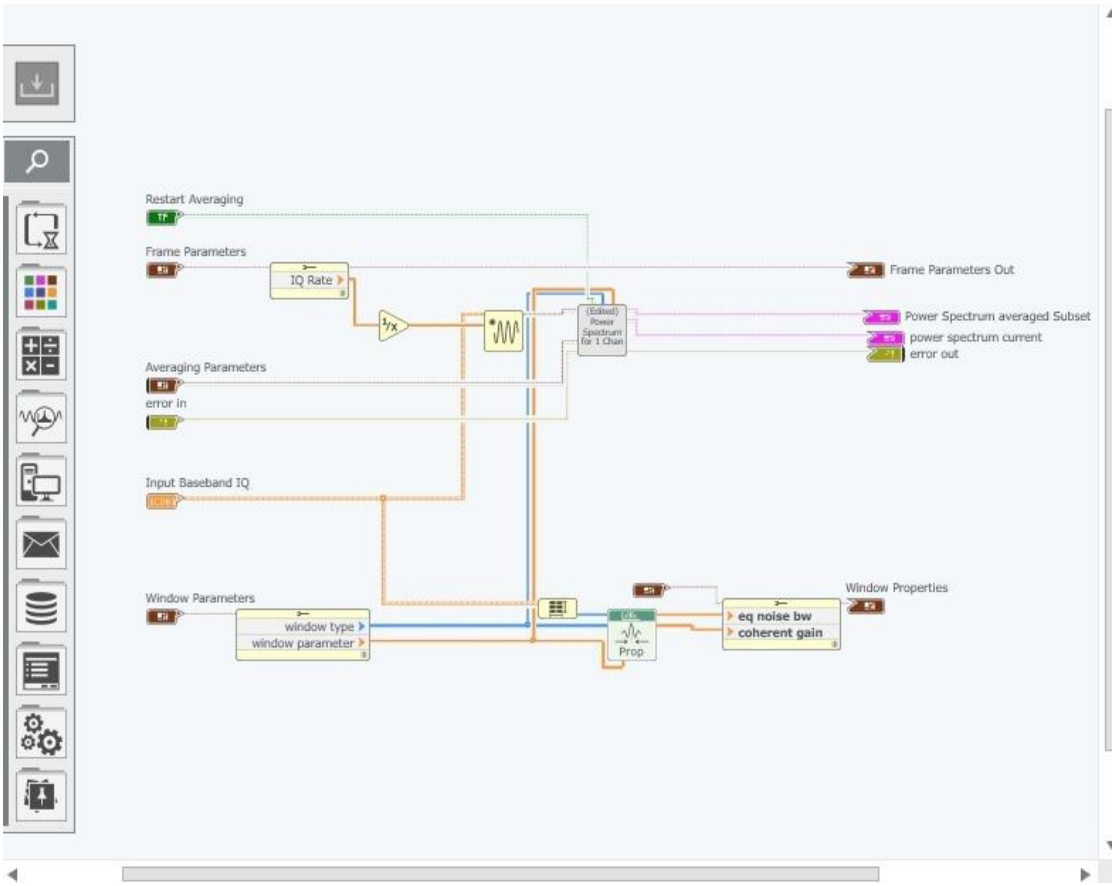


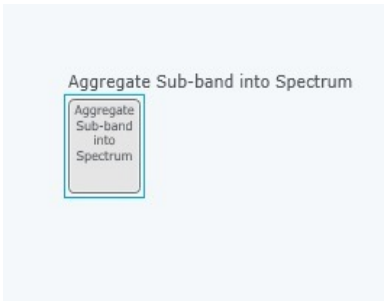
Figure 25 established sub-band

In this sub VI we take the output of frame parameters cluster as an input to the wave properties to extract the IQ and then take the output of the average parameters cluster and the error and the input baseband IQ signal and we take the output of the window parameters cluster as an input to cluster properties to extract the window type and the window parameter

All of these are taken as an input to each (Establish power spectrum for 1 Chan "sub VI") and (window properties by name VI).

We get from Establish power spectrum for 1 Chan "sub VI" the power of each spectrum and the average for each sub-carrier in our band.

We get from window properties by name VI the equation of noise and the coherent gain of each channel in our band.



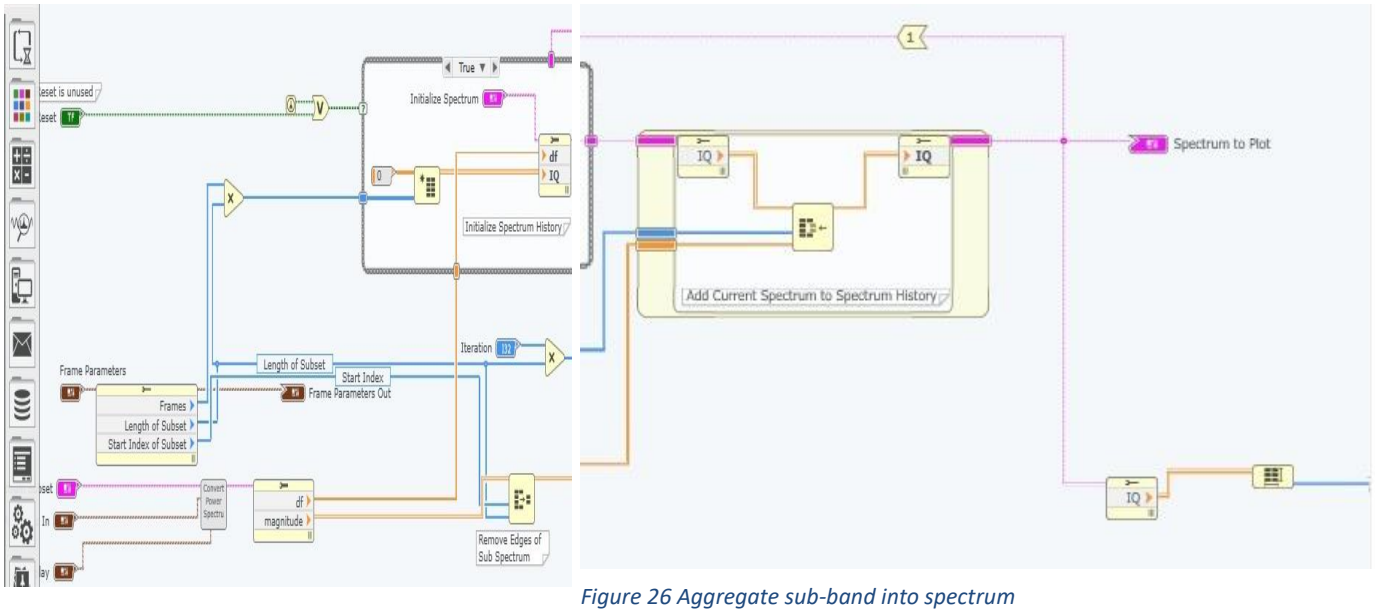


Figure 26 Aggregate sub-band into spectrum

Here in this sub-VI we take the output from the Read Data stage & Establish Sub-band stage as an input to this sub-VI to aggregate the read of each channel into one window.

By using:

1. **Initialize Array VI:** Creates an n-dimensional array in which every element is initialized to the value of element. Use the Positioning tool to resize the function and increase the number of dimensions (element, row, column, page, and so on) of the output array. The connector pane displays the default data types for this polymorphic function.

2. **Array Subset VI:** Returns a portion of array starting at index and containing length elements. When you wire an array to this function, the function resizes automatically to display index inputs for each dimension in the array you wire to array. The connector pane displays the default data types for this polymorphic function.
3. **Replace Array Subset VI:** Replaces an element or sub-array in an array at the point you specify in index. When you wire an array to this function, the function resizes automatically to display index inputs for each dimension in the array you wired. The connector pane displays the default data types for this polymorphic function**.**

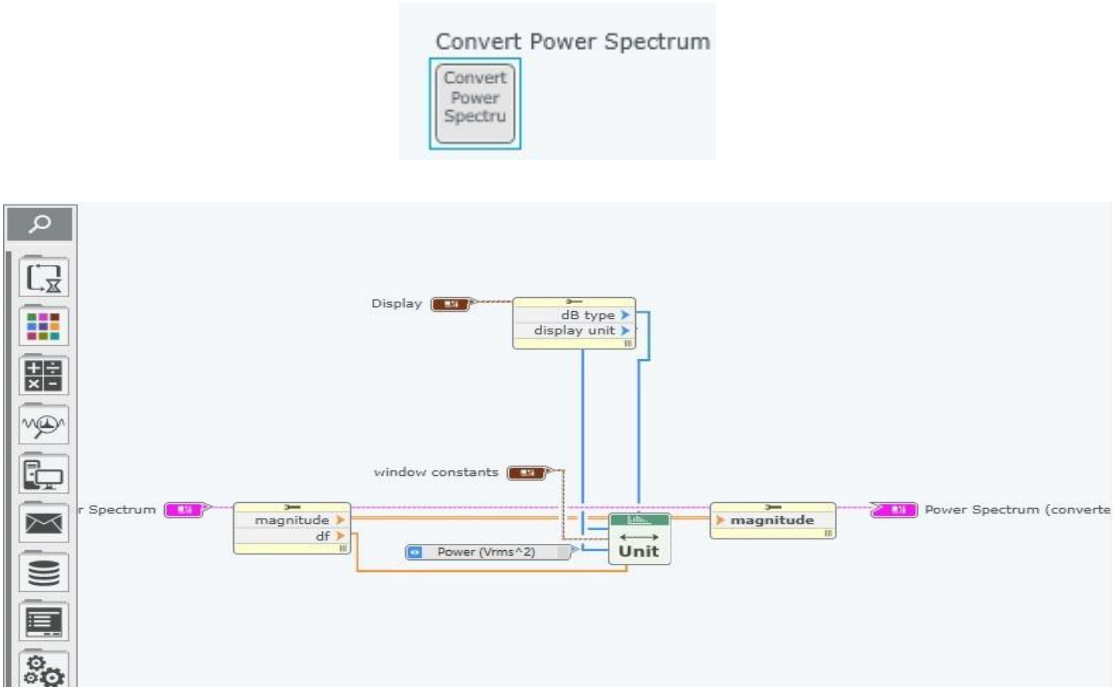
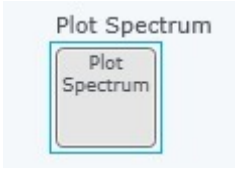


Figure 27 convert power Spectrum code

Here we use **spectrum unit conversion VI** that its main function is converts either the power, amplitude, or gain (amplitude ratio) spectrum to alternate formats including Log (decibel and dbm) and spectral density.



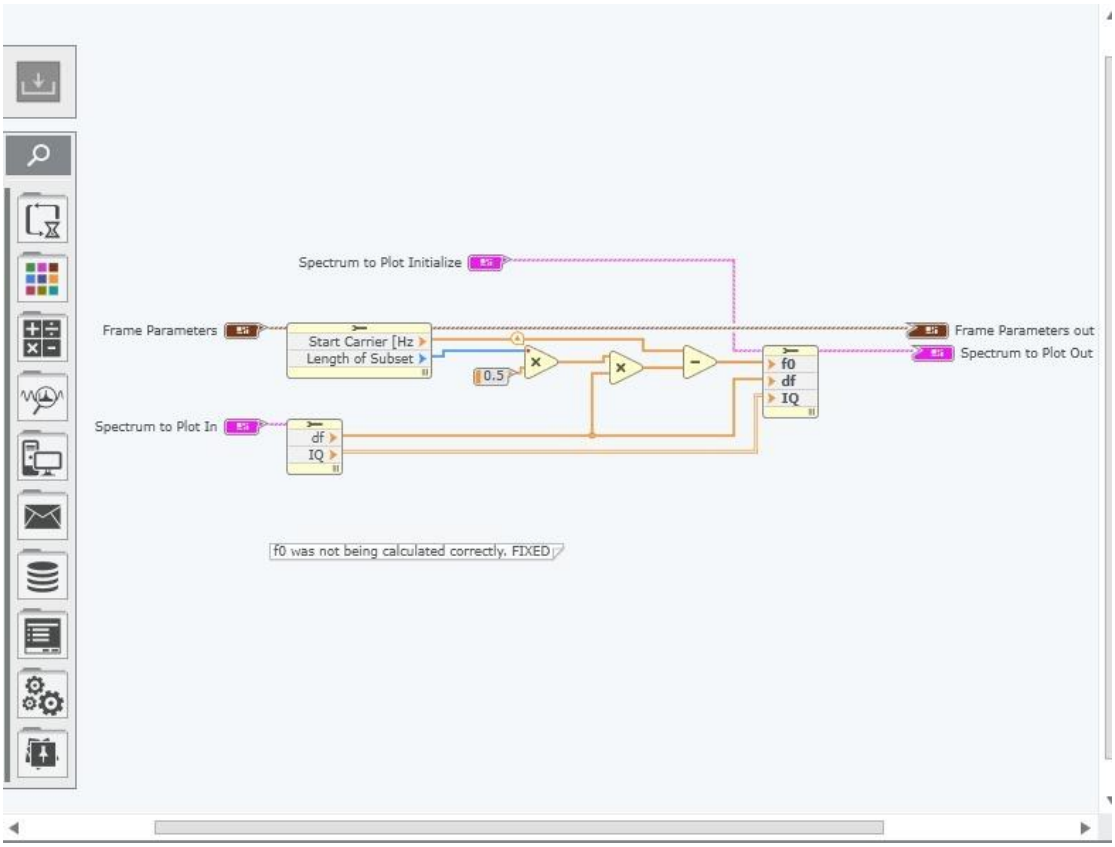
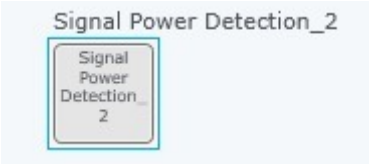


Figure 28 Plot Spectrum code

In this sub-VI code we take the output power from the Aggregate sub-VI and plot it into a graph in the front panel.



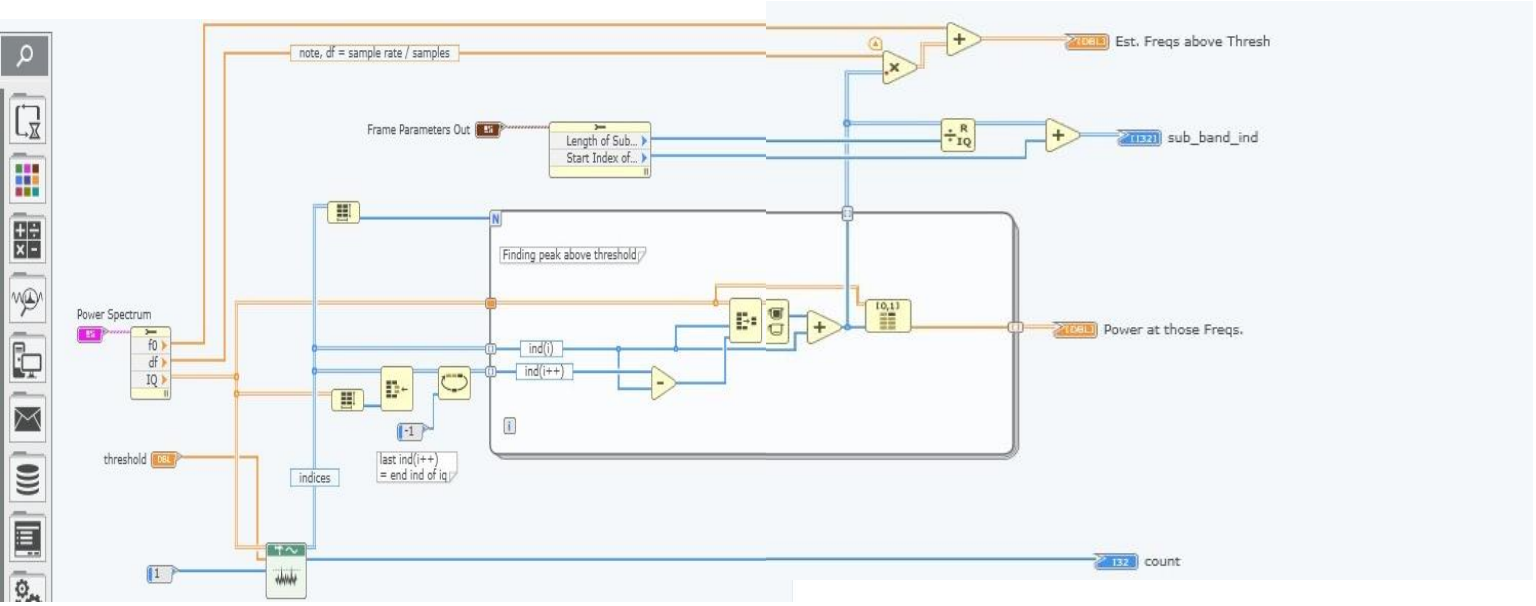
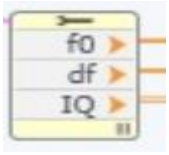
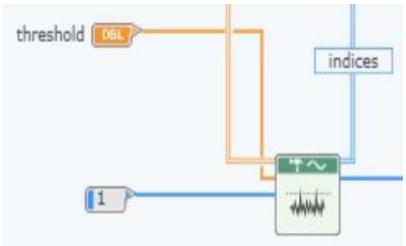


Figure 29 Signal Power Detection code



Here in this sub-VI we get the output from the Plot Spectrum stage which is FFT Power Spectrum and estimate the IQ and f0 and df from it by using Cluster properties VI.

We also use Threshold Detector VI which is used to analyzes the input sequence X which is the IQ of the signal for valid peaks and keeps a count of the number of peaks encountered and a record of Indices, which locates the points that exceed the threshold in a valid peak.



The output of this stage (power at the frequencies of the

band , estimate frequencies above thresh , sub-band index , count) are taken as input to the next stage.

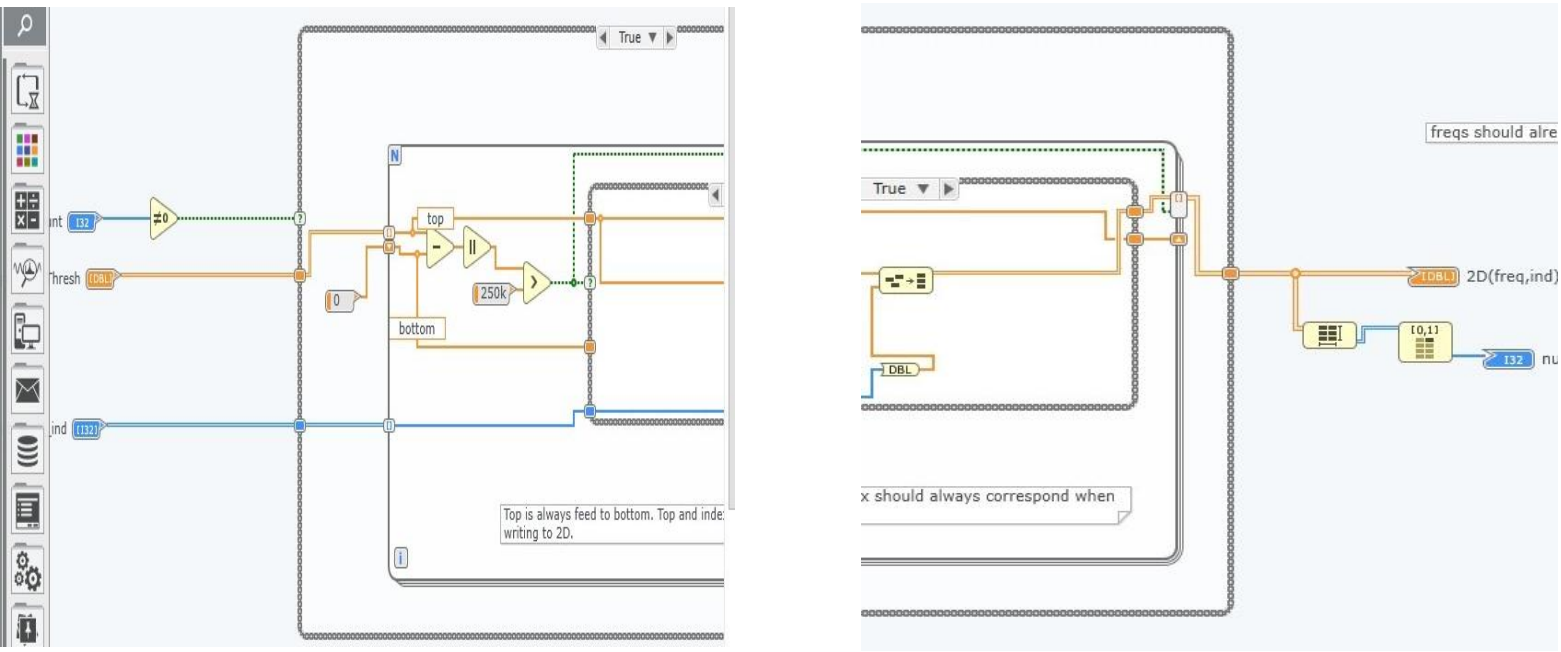


Figure 30 Array Fill code

By using this Sub-VI we can initiate 2D array which contain the frequencies and the indices.

The other thing that we can get from this stage is number of elements in the array

The output of this stage and from signal power detection stage are taken as an input in build cluster VI to get a list which contain the read of all signals power in each channel in the band in this testbed.

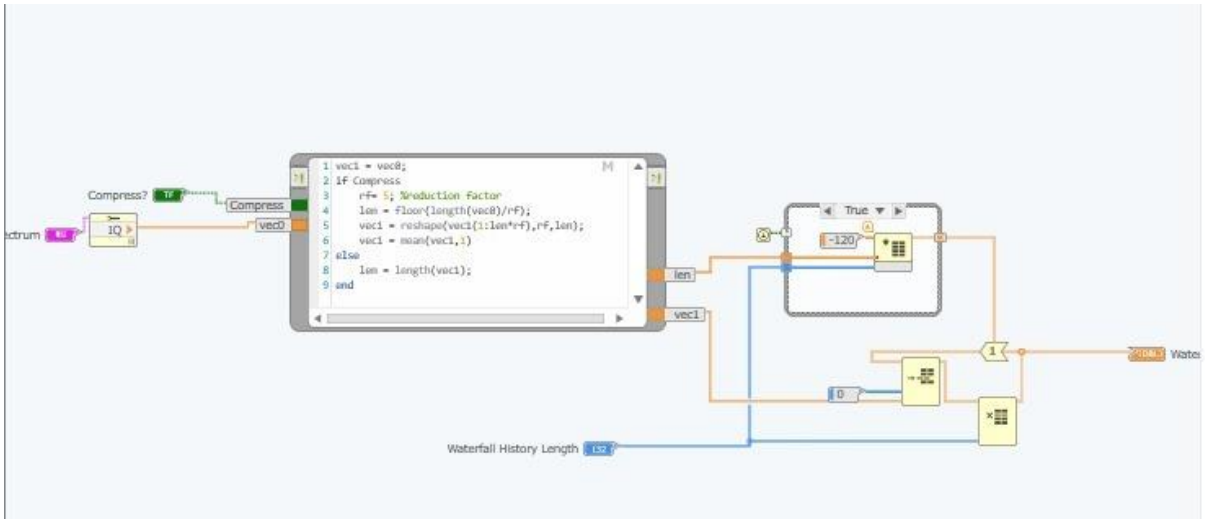


Figure 31: Waterfall Plot Generator code

By using this sub-VI we can show the PU and the Holes in our band like this.

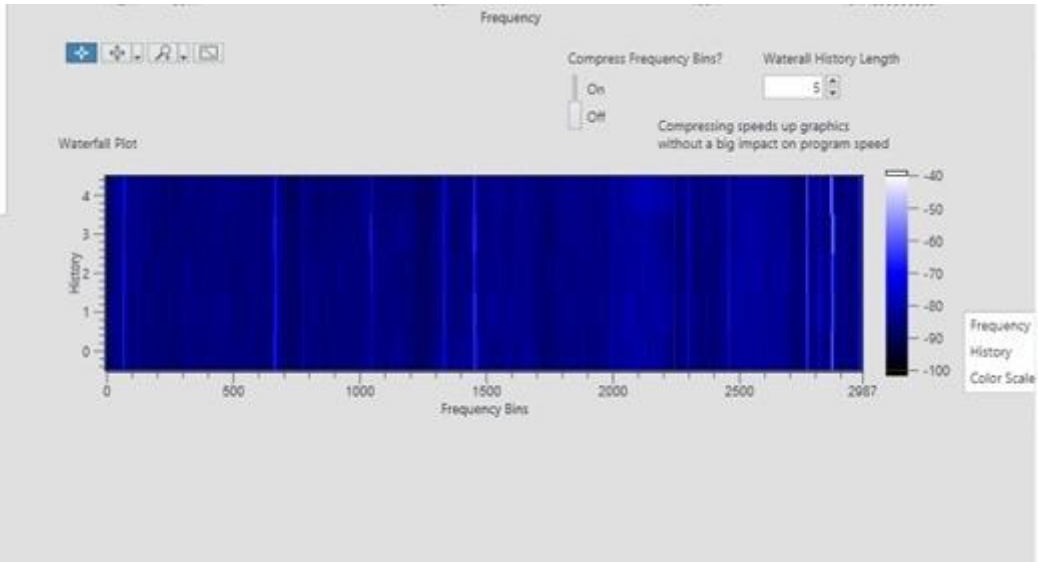
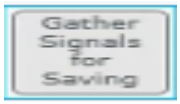


Figure 32 The output graph from waterfall



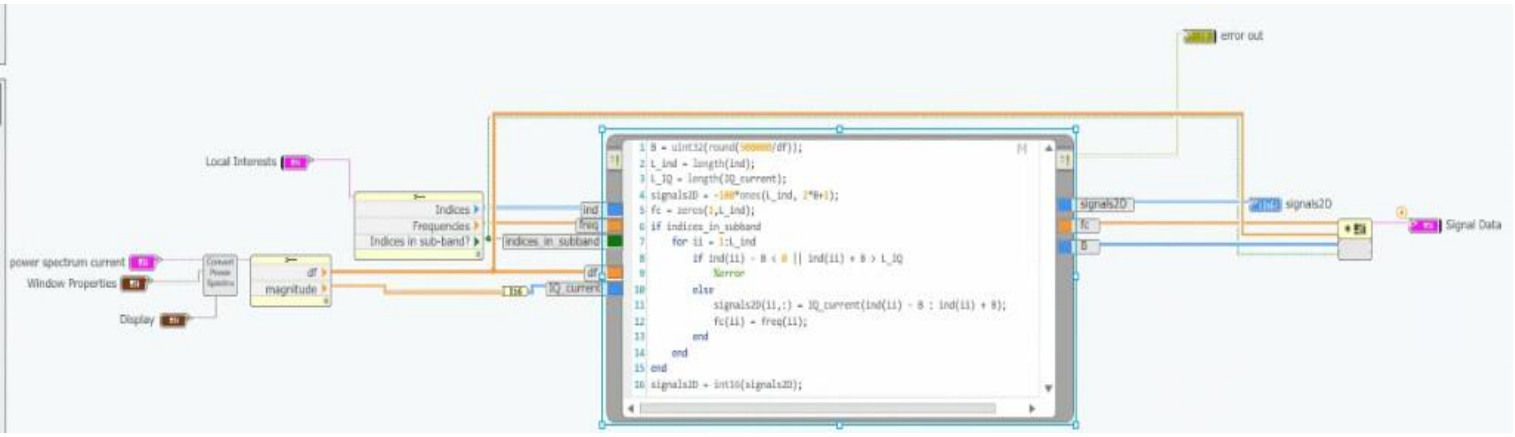
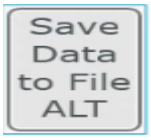


Figure 33: Gather Signals for saving sub-VI and its code



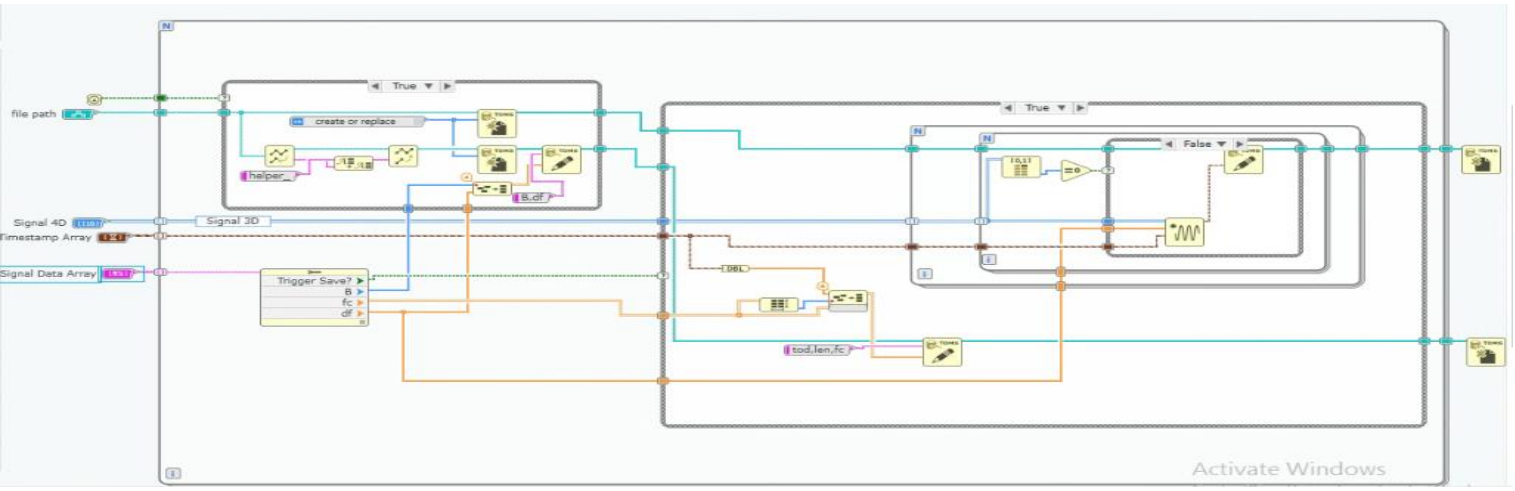
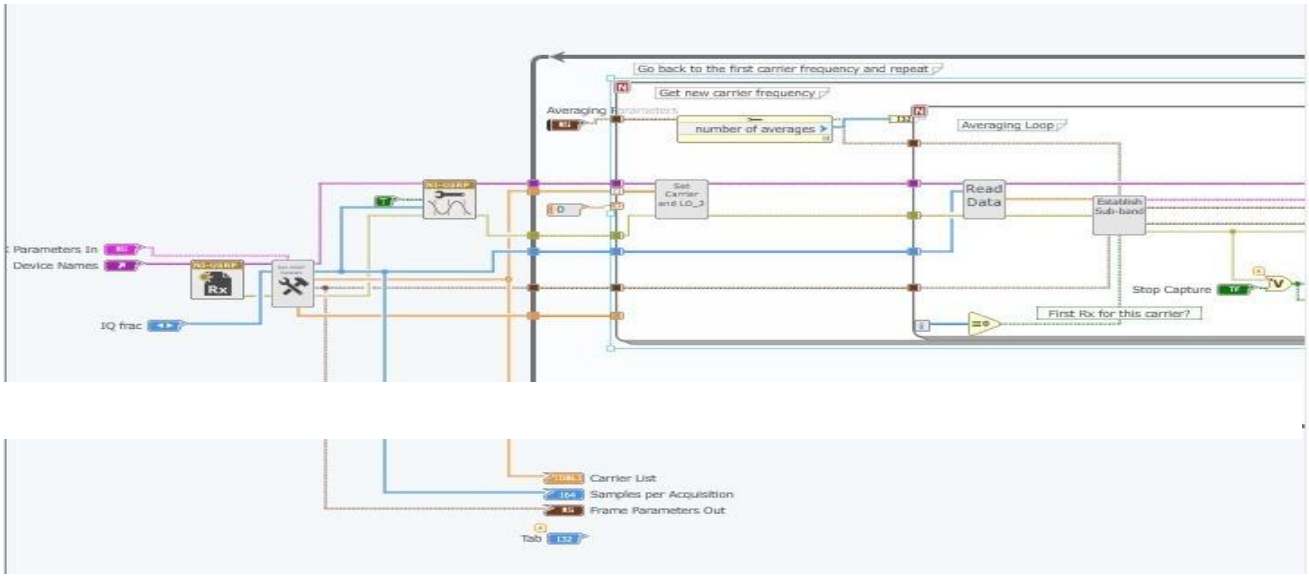


Figure 34 Save Data to file sub-VI and its code

By using these two sub-VI we can gather the data that the USRP is read from the band in n-dimensional array and then save this read into a file in our PC.



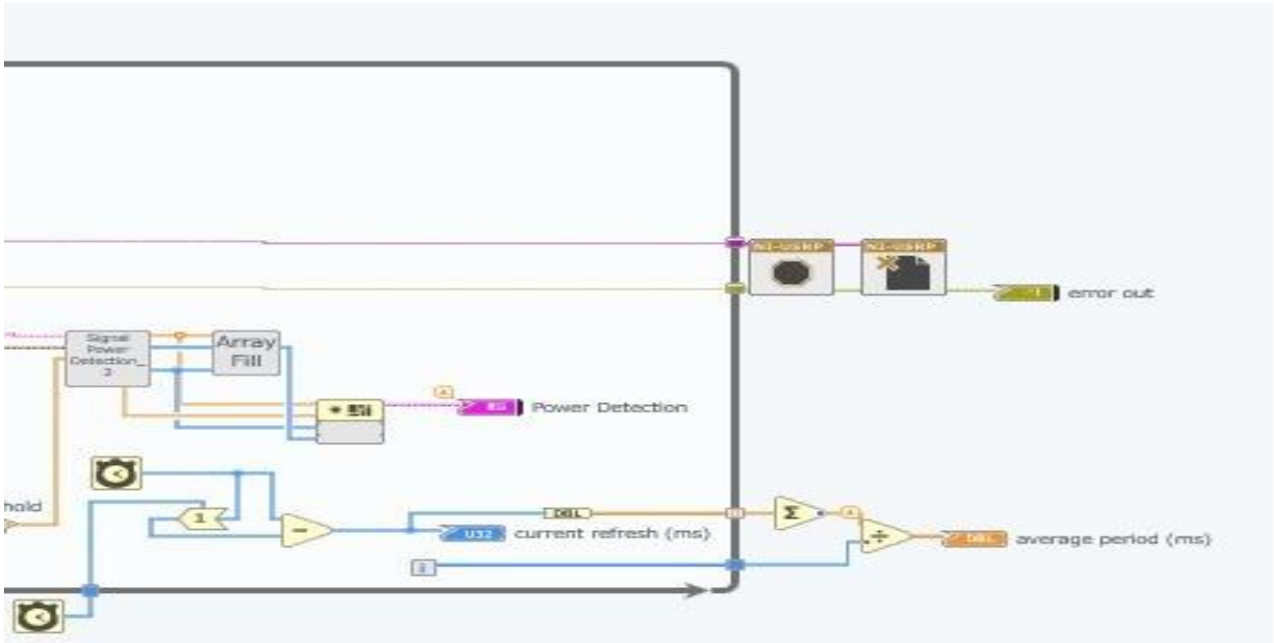
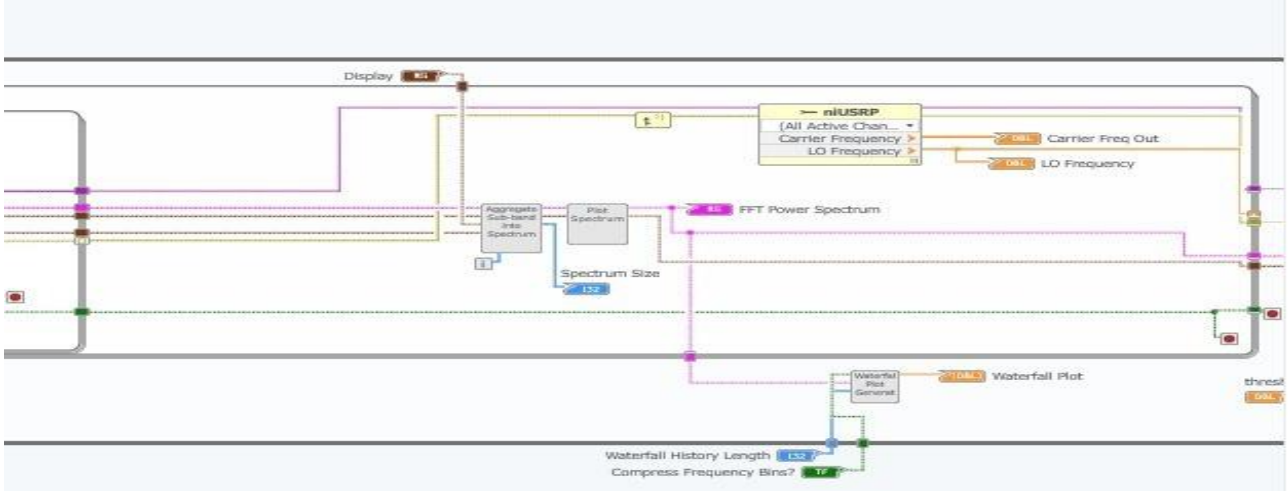
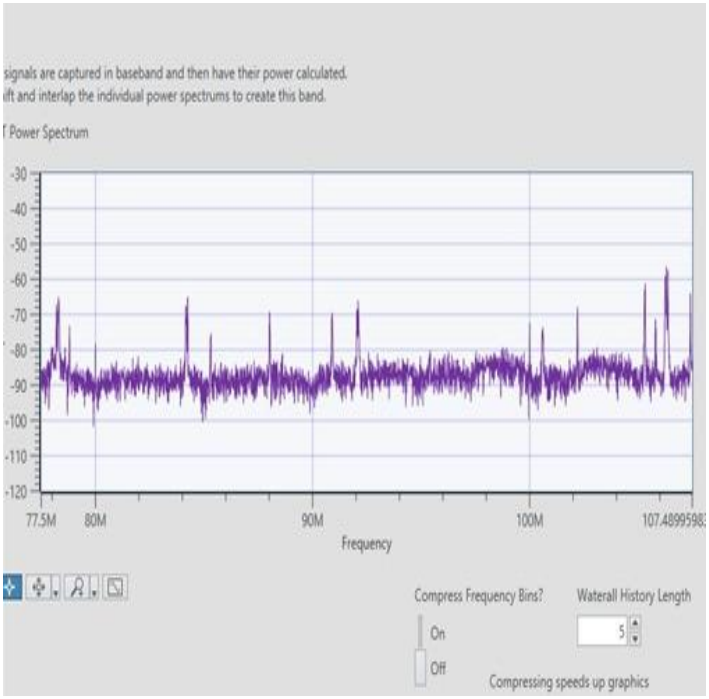


Figure 35 Spectrum Monitoring Full System





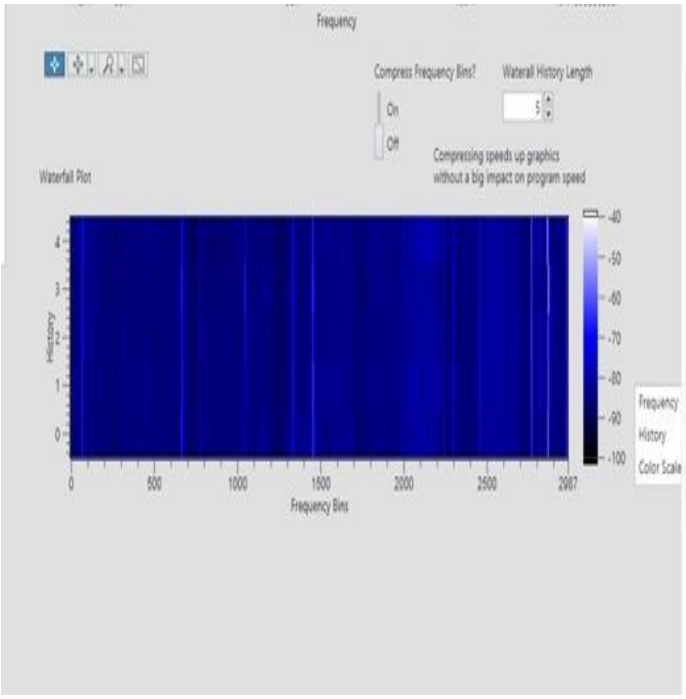
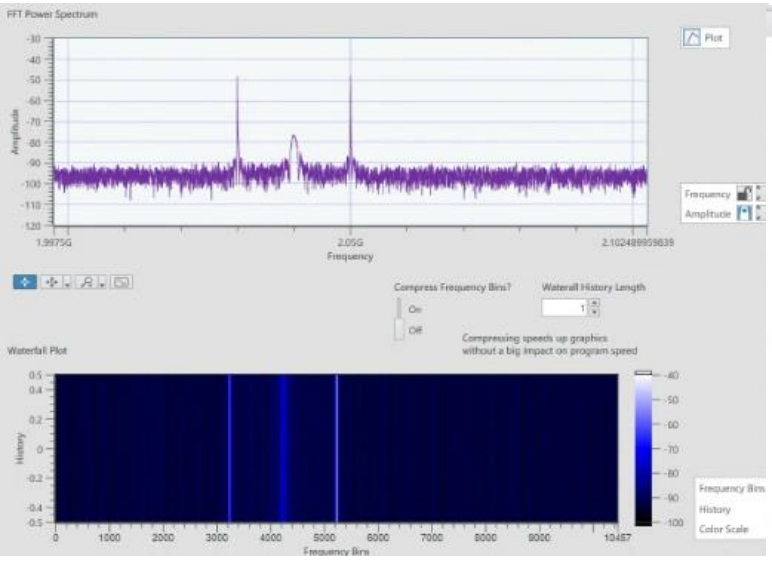
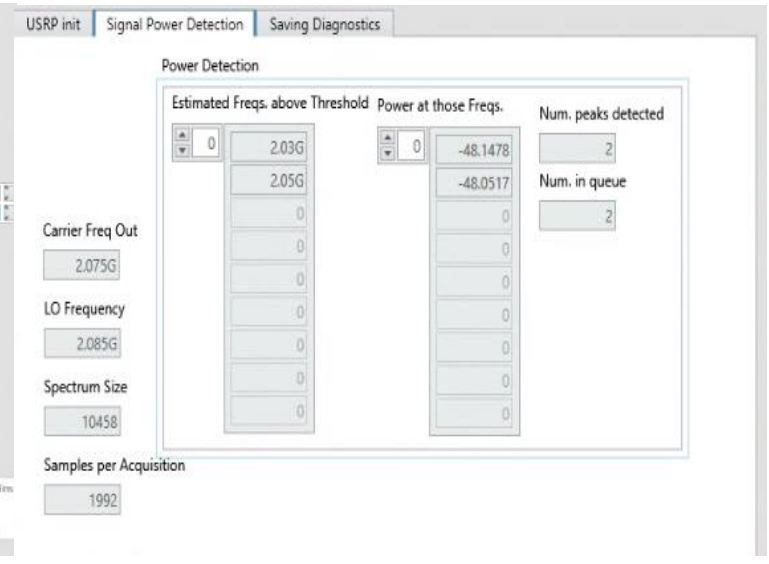


Figure 36 Output graph for FM band testbed.





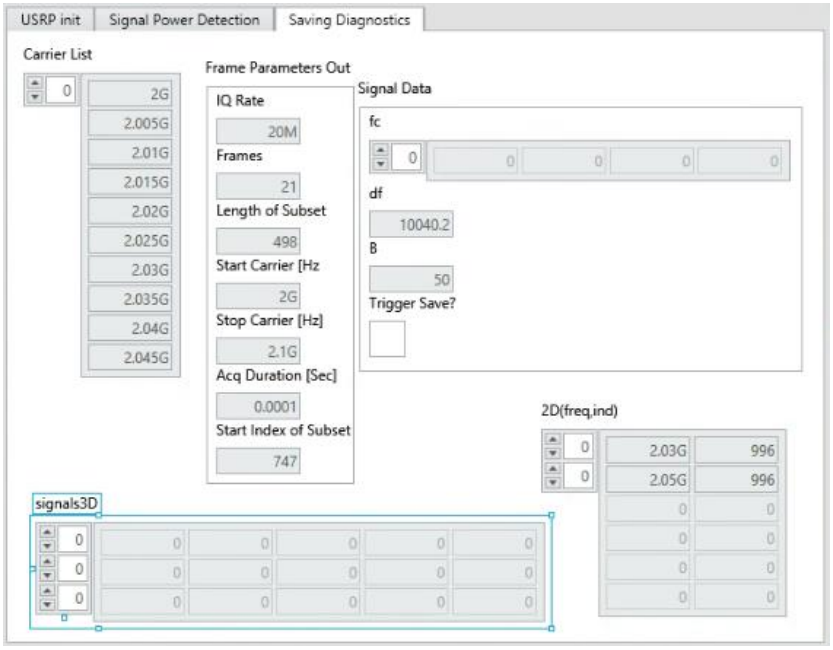


Figure 37 output graph for monitoring (2-2.1) G band

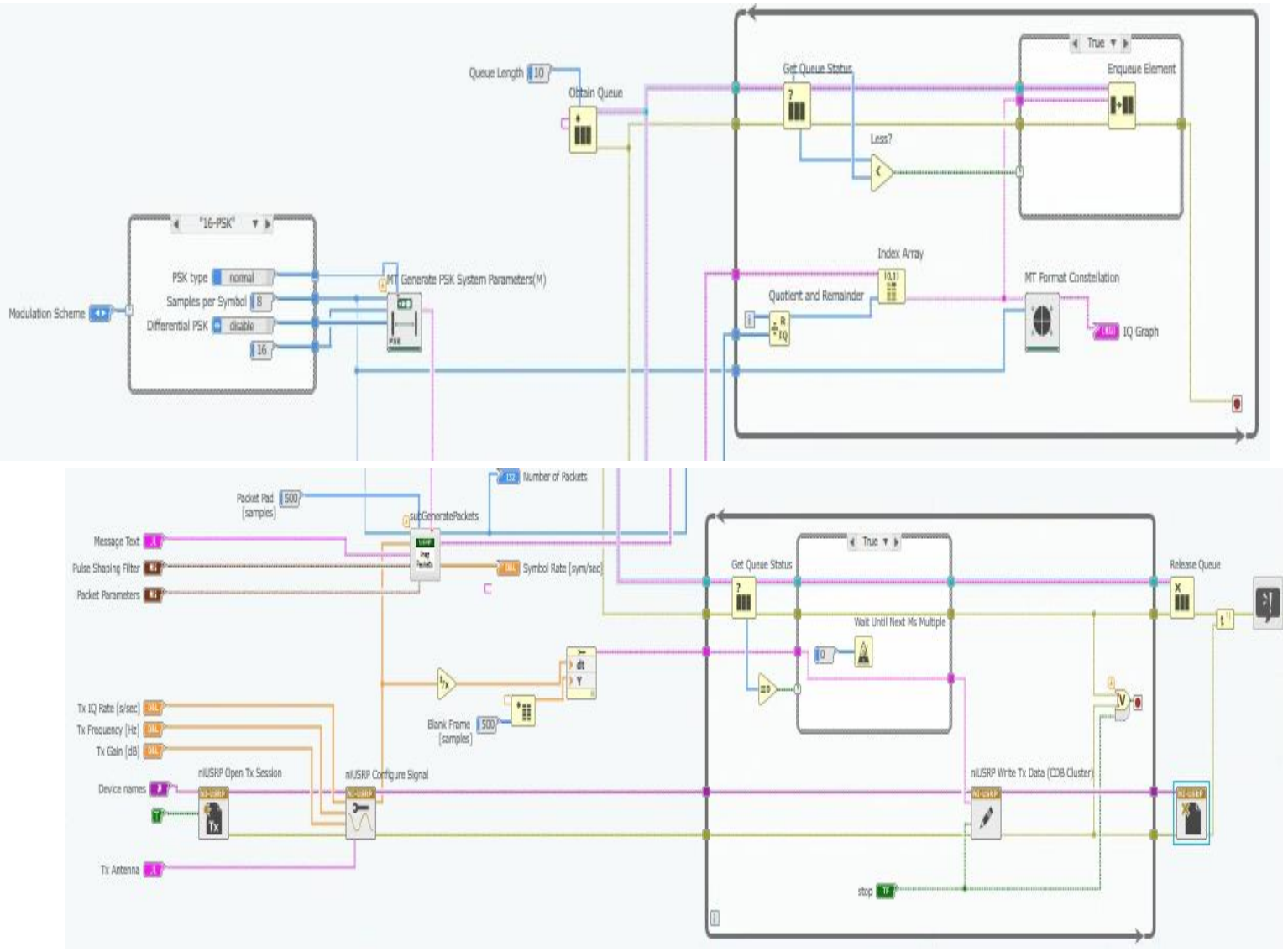
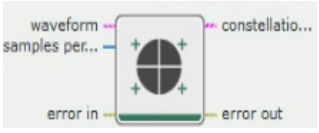


Figure 38 PSK Packet transmitter code

In this system we try to transmit packets of any message by using BPSK or QPSK or OPSK or DQPSK or 16-PSK or 8-PSK Modulation and schemes and using Raised Cosine or Root Raised Cosine or Gaussian Filter.

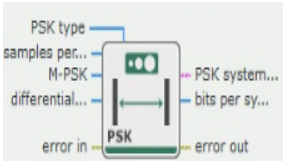
The important VIs and Sub-VIs in this code is:

1. **MT Format Constellation**: Prepares a signal for presentation on a graph that shows the detected symbol locations and the transitions between those symbols. The node specifies a complex-valued waveform and displays a constellation plot of the waveform contents. This node assumes that the input

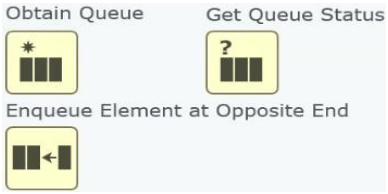


waveform is a digitally modulated complex baseband signal containing samples that fall on symbol boundaries with a sample rate that is an integer multiple of the samples per symbol value.

2. **MT Generate PSK System Parameters (M):** generate parameters such as samples per symbol, symbol map, and differential psk.



3. **Some VI That help me to store and send the word of the message:**



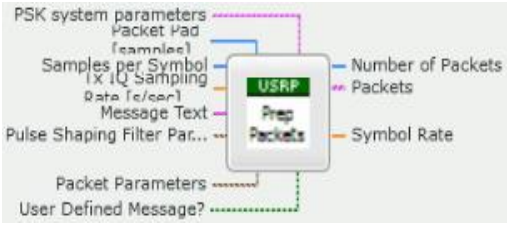
Queues typically use a first-in-first-out flow of data. In rare situations, you might want to interrupt this normal flow of data by adding an element to the front of the queue. After you add an element to the front of a queue, the next Dequeue Element node you call removes the element you added to the front.

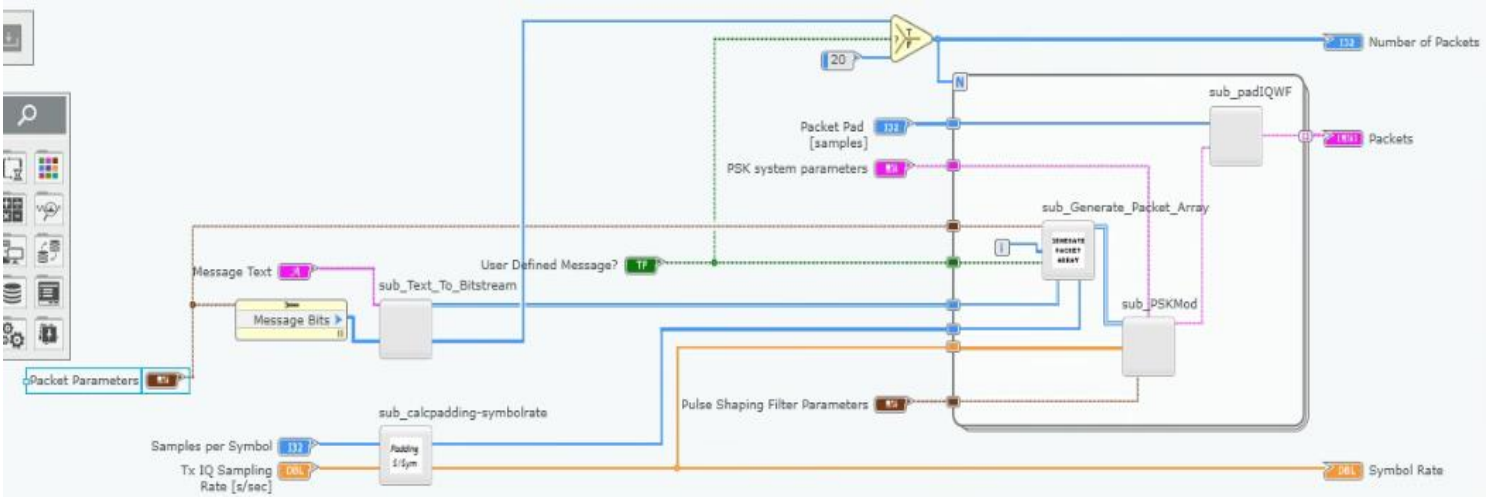
Obtain Queue: Returns a reference to a queue. Use this reference when calling other nodes that perform queue operations.

Get Queue Status: Returns information about the current state of a queue, such as the number of elements currently in the queue.

Enqueue Element at Opposite End: Adds an element to the front of a queue.

4. **subGeneratePackets sub-VI:** in this sub-VI we take the PSK system parameters that out from MT Generate PSK System Parameters (M) and the number of samples and the message that the user want to write and the IQ sampling rate that's coming out niUSRP Configure Signal and pulse shaping filter which is cluster used to help us to change the filter type and the filter length and the alpha as we need and the last input is the packet parameters which is a cluster that help us to control the number of guard, sync, message, and PN sequence order for sync bits.





subGeneratePackets and its code39 Figure

Figure 40 Group of sub-Vis that used in sub Generate Packets code

sub_Text_To_Bitstream sub-sub-VI: used to convert the message into a stream of bits.

sub_Generate_Packet_Array sub-sub-VI: used to generate the packets (the bits fill into 1-D array to make the packet).

sub_PSKMod sub-sub VI: by this sub VI we can modulate the output packets that's out from sub_Generate_Packet_Array.

sub_padIQWF sub-sub VI: in this sub-VI we get the output of sub_PSKMod sub-VI as an input and by the code into this sub-VI we can represent the output complex waveform.

✓ **The outputs and The parameters configuration.**

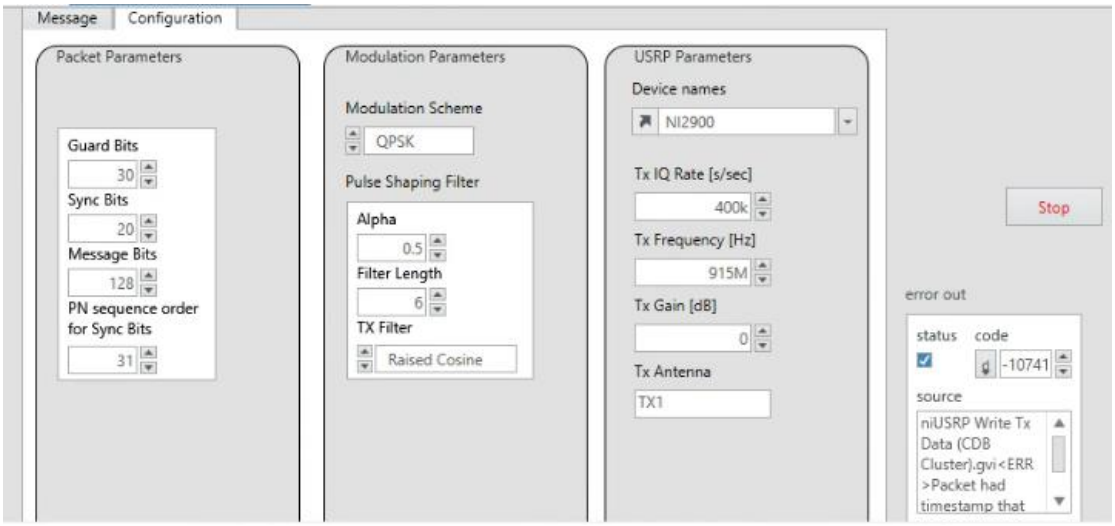


Figure 41 The Tx parameters configuration in front panel

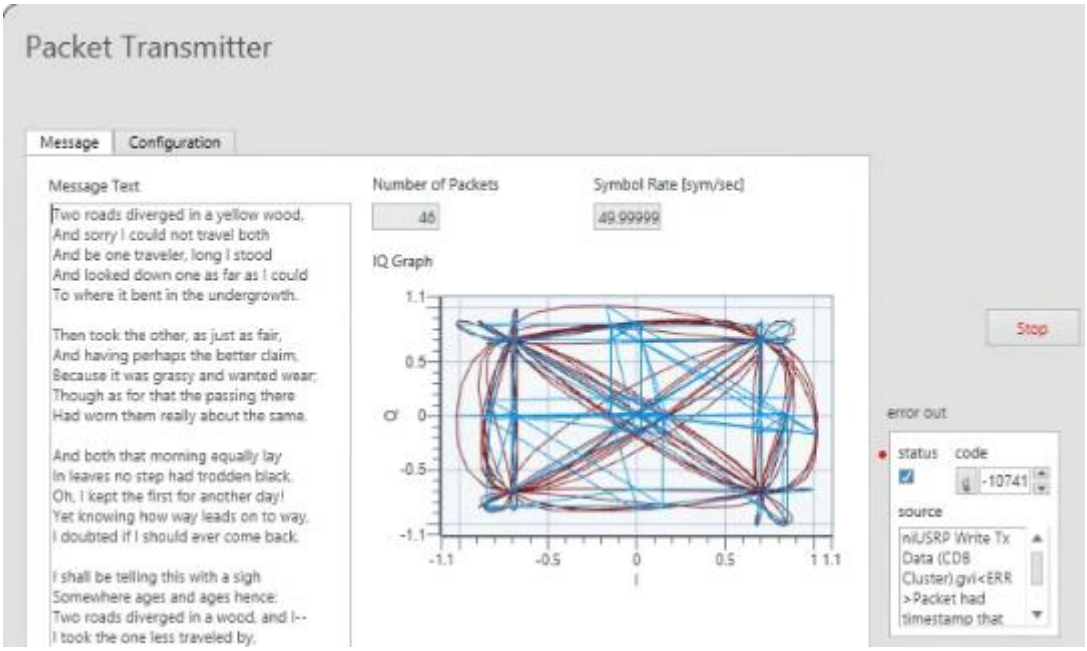
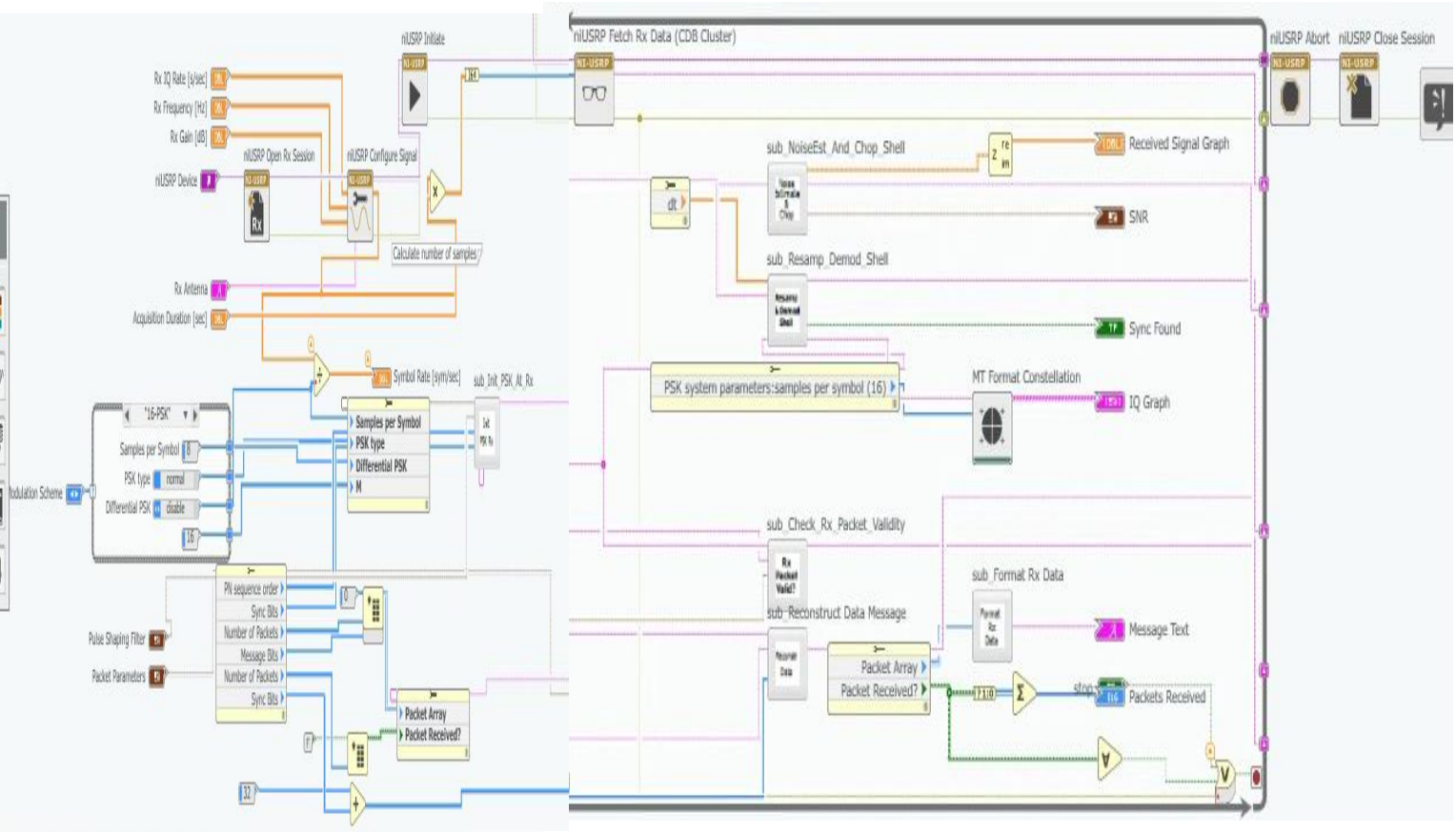


Figure 42 The output graph for QPSK Mod and Raised Cosine filter testbed

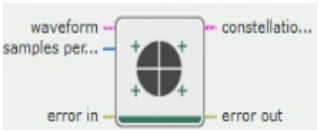


PSK Packet Receiver code 43 Figure

In this system we try to receive packets of any message by using BPSK or QPSK or OPSK or DQPSK or 16-PSK or 8-PSK Modulation and schemes and using Raised Cosine or Root Raised Cosine or Gaussian Filter.

The important VIs and Sub-VIs in this code is:

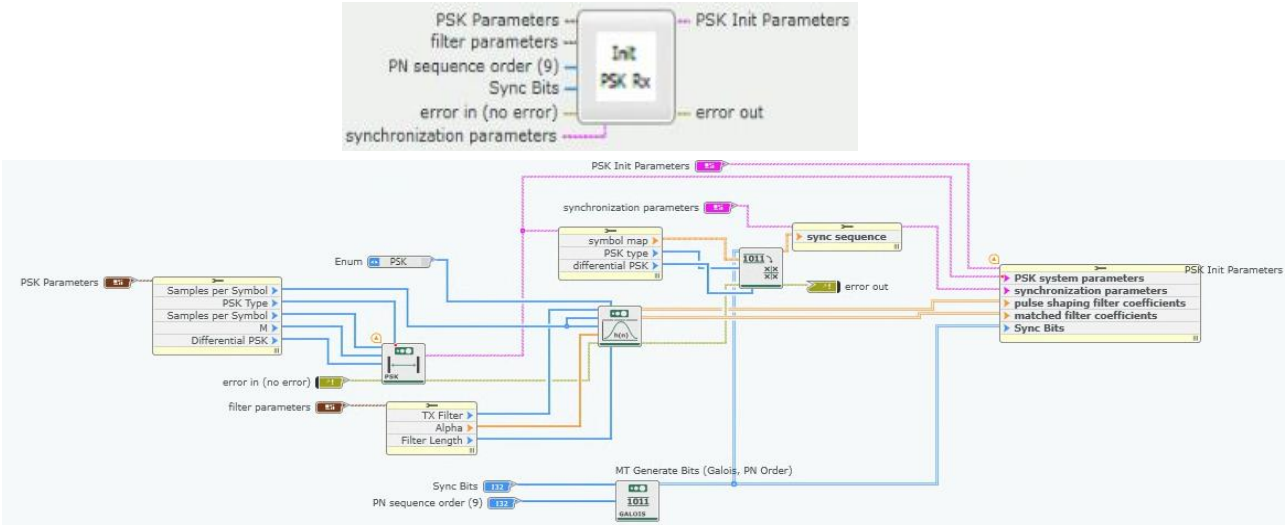
1. **MT Format Constellation**: Prepares a signal for presentation on a graph that shows the detected symbol locations and the transitions between those symbols. The node specifies a complex-valued waveform and displays a constellation plot of the waveform contents. This node assumes that the input waveform is a digitally modulated complex baseband signal containing samples that fall on symbol boundaries with a sample rate that is an integer multiple of the samples per symbol value.



2. **Some sub-VIs that help me to receive the transmitted packets:**

Figure 44 group of sub-VI that used in Rx

sub_Init_PSK_At_Rx sub-VI:



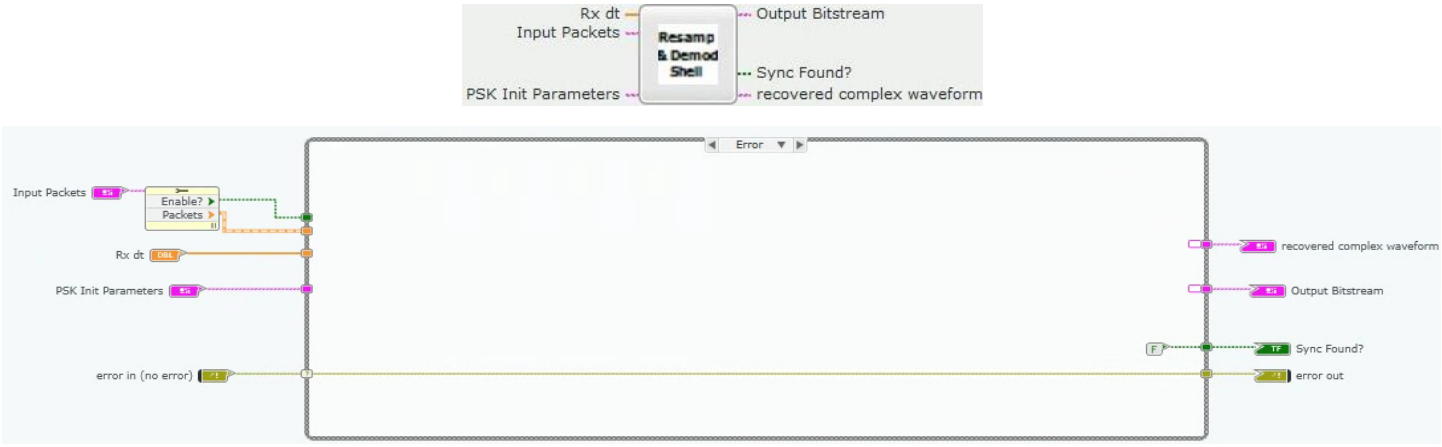
45 Figure sub_Init_PSK_At_Rx sub-VI code

We here take the PSK parameters such as samples per symbol, PSK type, M, Differential PSK and take the filter parameters such as Tx filter, Alpha, filter

length and take PN sequence order and sync bits and synchronization parameters cluster as an input to this sub-VI to get the number of samples per symbol, PSK system parameters, pulse shaping filter coefficient, matched filter coefficient, synchronization parameters, and synchronization bits.

We take all these as an input to sub_Resamp_Demod_Shell sub-VI and take the number of samples per symbol only as an input to MT Format Constellation to plot the IQ graph.

- **sub_Resamp_Demod_Shell sub-VI:**



46 Figure sub_Resamp_Demod_Shell sub-VI and its code

This sub-VI is used as symbol detector/demodulator combination which also contain matched filter so here in this sub-VI we take the differential over time of the signal (Rx dt) and the input packets and PSK Init Parameters cluster as an input to determine the symbol that most matches the signal received, and at this stage, the timing recovery is very important.

We get from this sub-VI the Bit Stream and the recovered complex wave form that is taken as an input with samples per symbol from the cluster out from the previous sub-VI to MT Format Constellation to plot the IQ graph.

- **sub_Check_Rx_Packet_Validity sub-VI:**

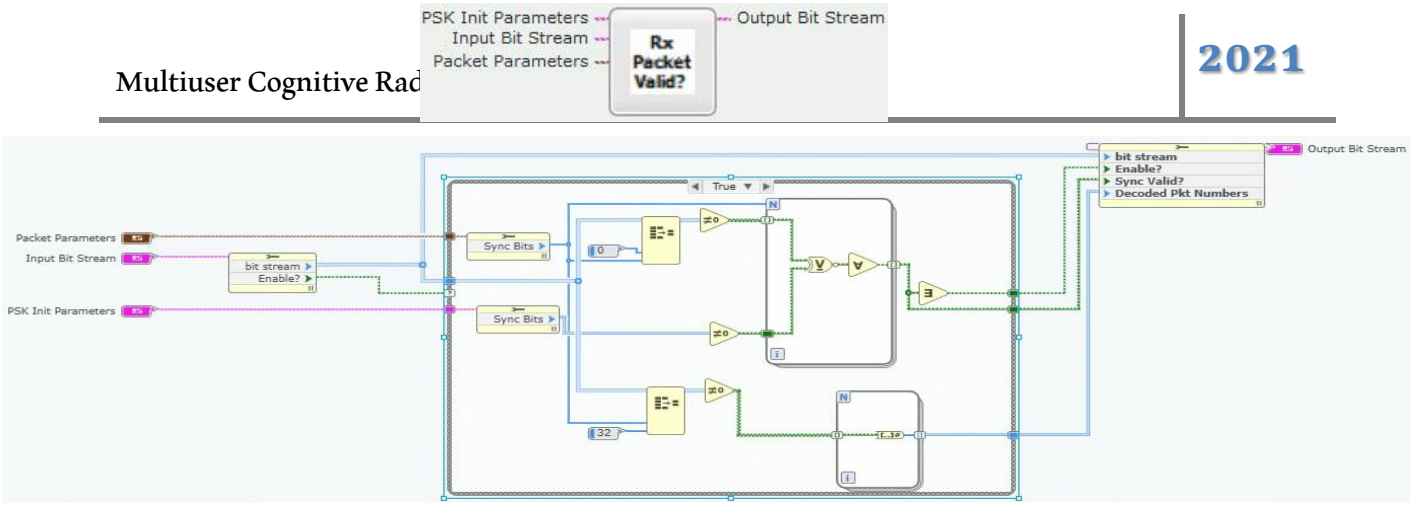


Figure 47 sub_Check_Rx_Packet_Validity sub-VI and its code

Here in this sub-VI we check the validity of symbols we take the PSK Init Parameters and the Input Bit Stream and Packet Parameters and doing process on them to get some outputs those outputs take as an input to cluster properties which its inputs are the bit stream, sync valid, enables, and the decoded packet numbers.

We get from this cluster Out Bit Stream which is the output of the sub_Check_Rx_Packet_Validity sub-VI.

sub_NoiseEst_And_Chop_Shell sub-VI:

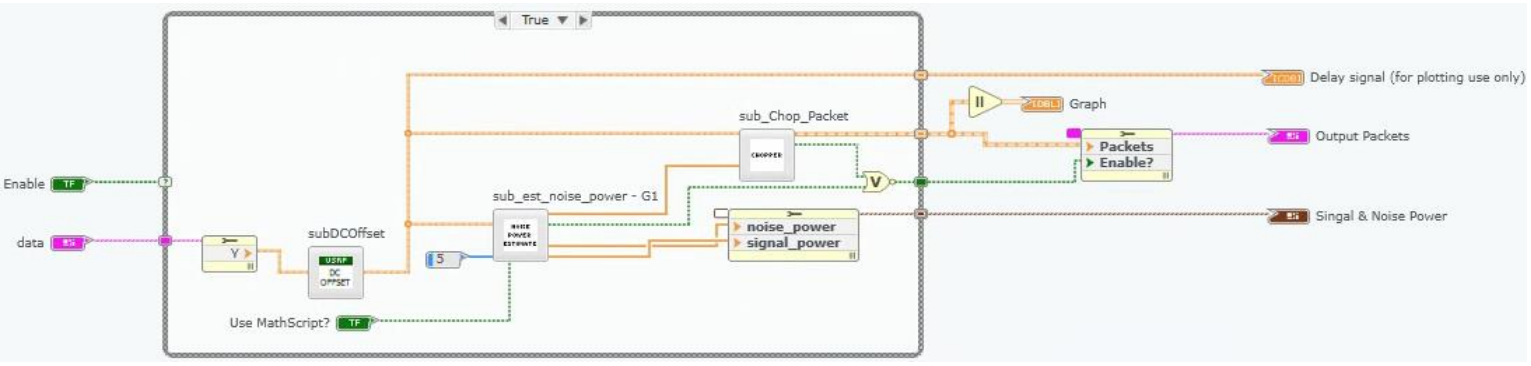


Figure 48 sub_NoiseEst_And_Chop_Shell sub-VI and its code

This sub-VI contain another 3 sub-VIs into its code each one of them has a function to do

1. subDCOffset

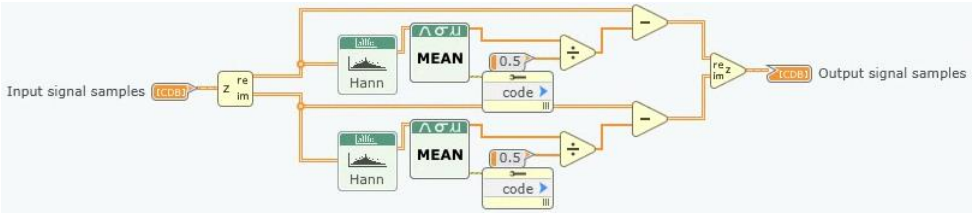
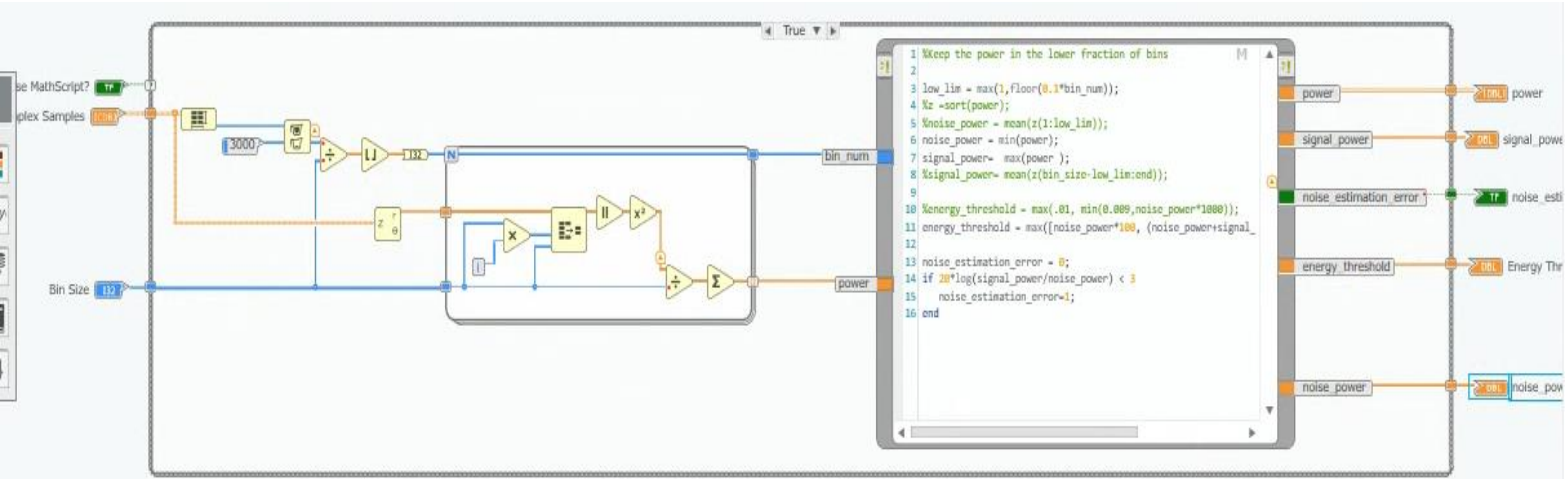


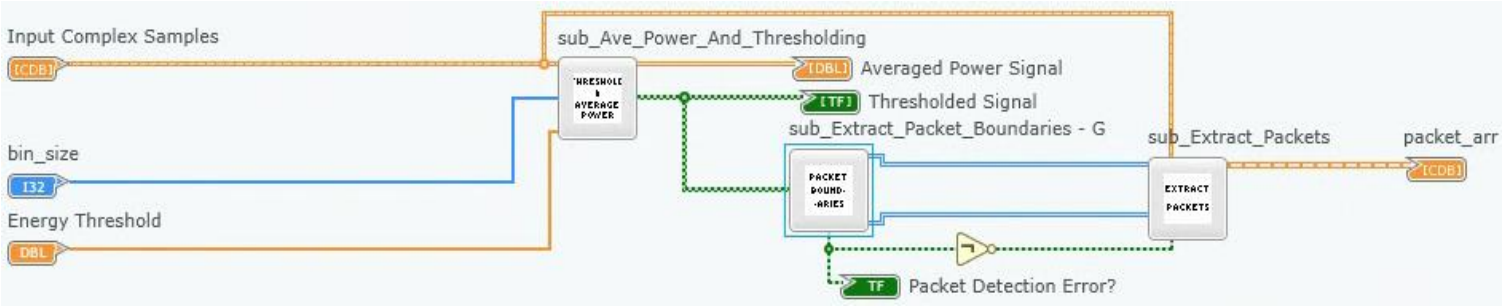
Figure 49 subDCOffset sub-VI code

2. sub_est_noise_power - G1



50 Figure sub_est_noise_power-G1 sub-VI code

3. sub_Chop_Packet



51 Figure

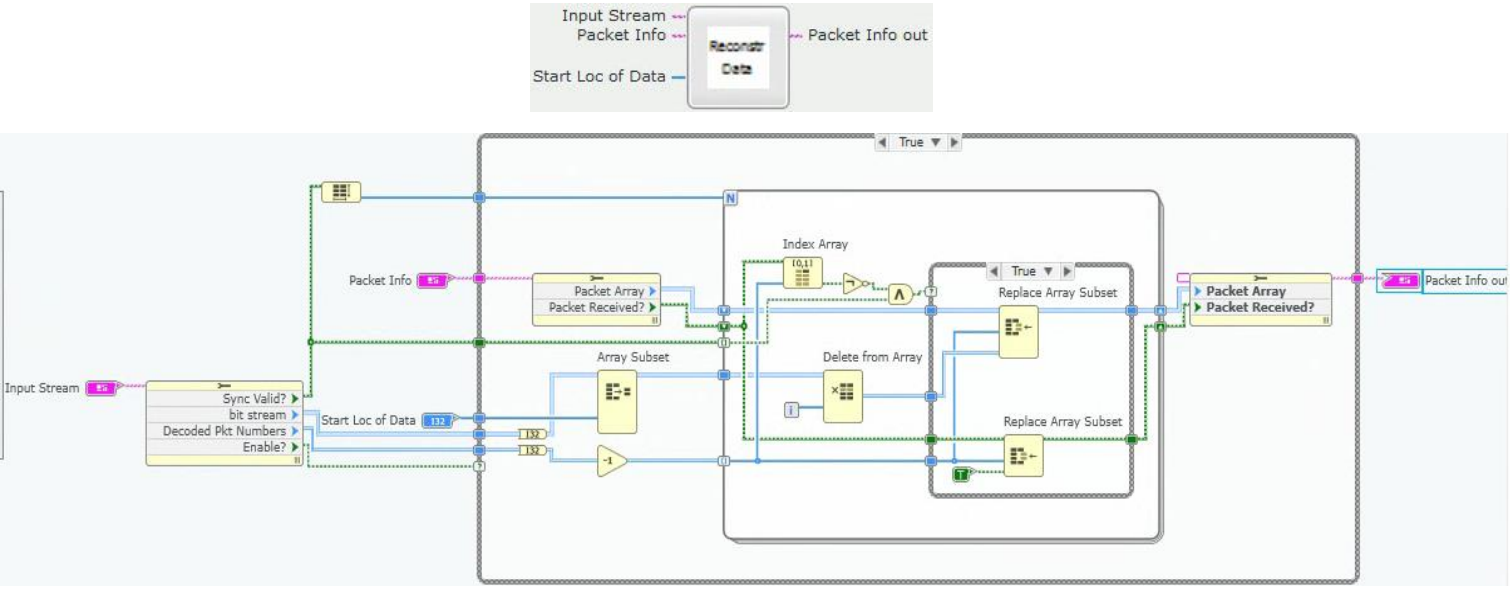
We can get inside sub_NoiseEst_And_Chop_Shell sub-VI the Threshold and by threshold we can check if there is error in any received packet and

we can get array that contain the packets and the average power signal and the power of signal and noise.

By using all of these parameters we finally can get from inside sub_NoiseEst_And_Chop_Shell sub-VI (the output):

SNR(signal & noise power), output packets, and the received signal graph.

- **sub_Reconstruct Data Message sub-VI:**



52 Figure : sub_Reconstruct Data Message sub-VI and its code

in this sub-VI we take the Input Stream cluster which contain (sync valid?), bit stream, Decoded packet Numbers, and (Enable?) and we take the packet Info cluster which contain (Packet Array), and (Packet Received?) and we take the Start Location of Data as an input to this sub-VI to get the Packet Info out which is taken to cluster properties to get the Packet Array (use as input to sub_Format Rx Data) and Packet Received? (Used to stop the loop when the Rx receives all the packets).

sub_Format Rx Data sub-VI:

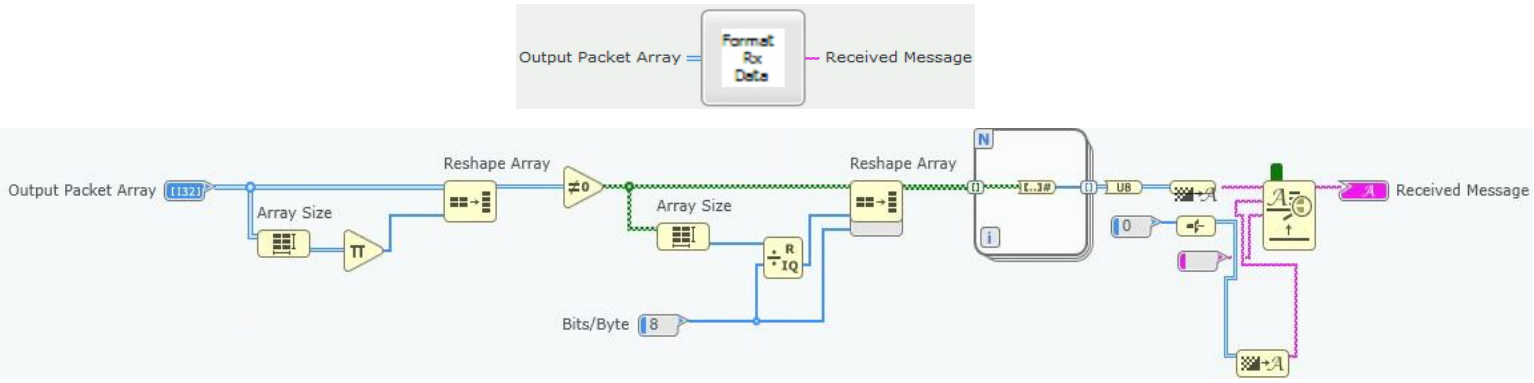


Figure 53 sub_Format RX Data sub-VI and its code

By using this sub-VI we can read the received message.

✓ **The outputs and the parameters configuration**.

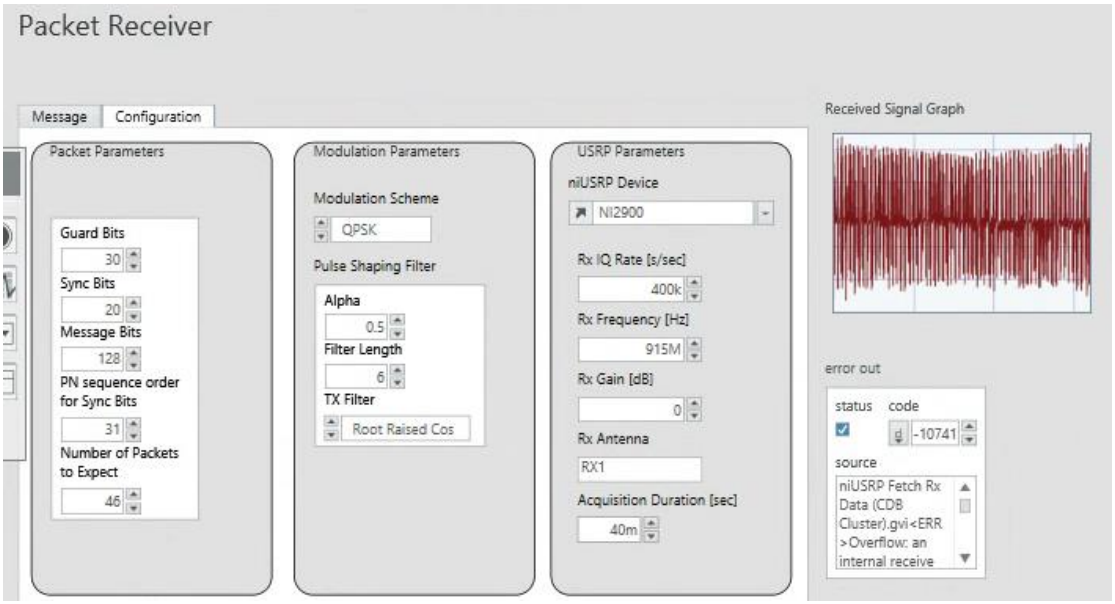


Figure 54 The Rx parameters configuration & the signal graph in front panel

9 GNU Radio.

GNU Radio is an open-source-based on Python and C++ programming language architecture that provides a free software toolkit for learning, building, and deploying software radio. GNU radio is particularly well suited for Linux operating systems. GNU Radio includes a library of signal processing blocks like modulators, demodulators, and filters which are used to construct a radio. The incorporation of the GNU Radio software package in the cognitive radio allows for a highly flexible and scalable system.

9.1 System Implementation

9.2 Design of PU based on OFDM Modulation:

We can't find an environment that helps us implement a CR system because there isn't a principal user, but we simulate the process by constructing a PU in our system and using OFDM Modulation

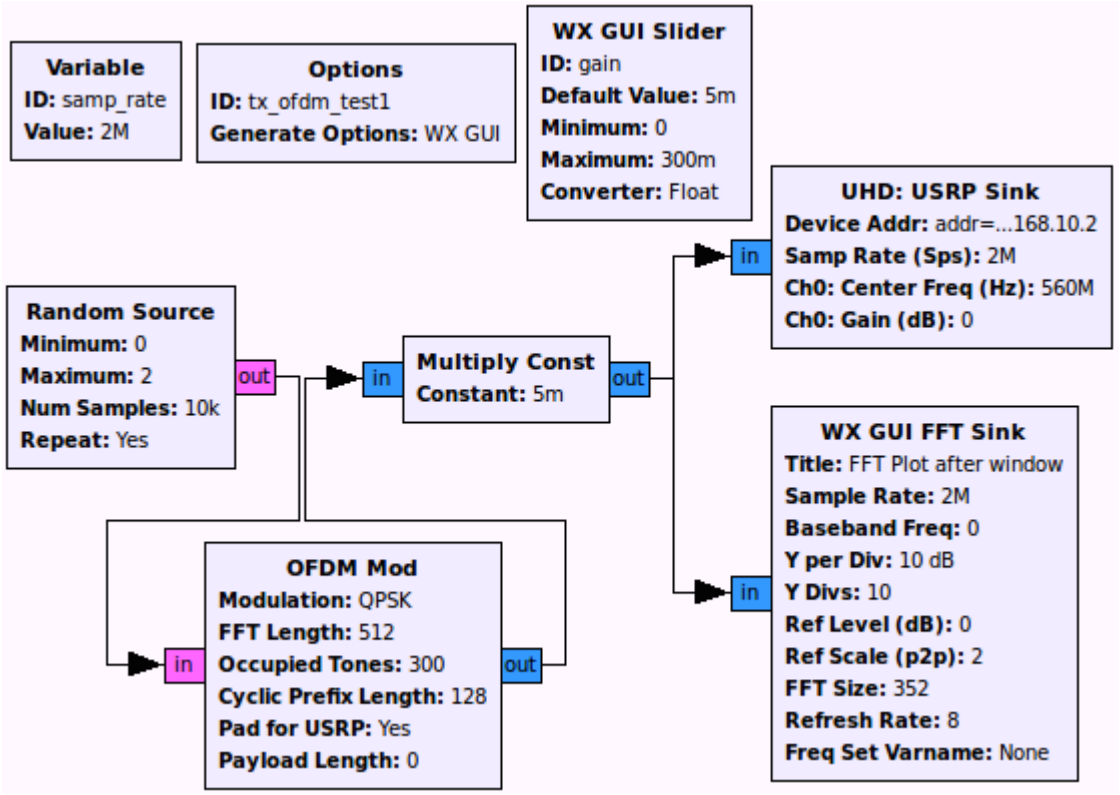


Figure 55 PU using OFDM

9.3 Spectrum sensing

Design for one channel

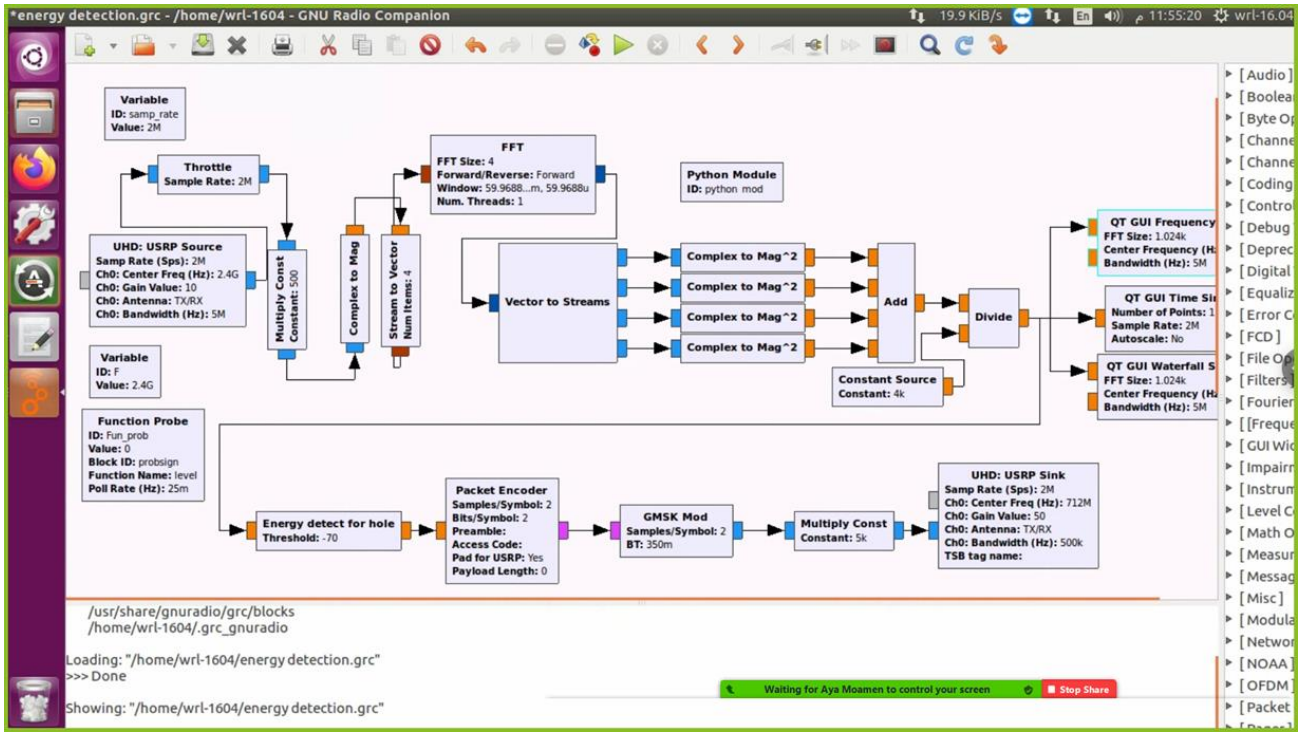
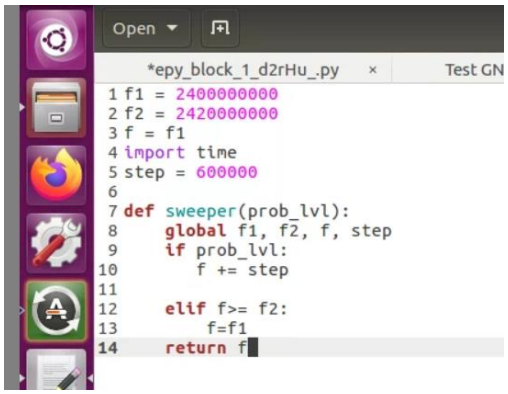


Figure 56 design of spectrum sensing

This design allows us to check if the channel used by the PU or not so we used Python module to change a channel by function we create that returned a result in Frequency variable.



And when the design check in channels we determine we make an array that contain a channel don't use or a hole like a figure 17.

PU found

PU not found

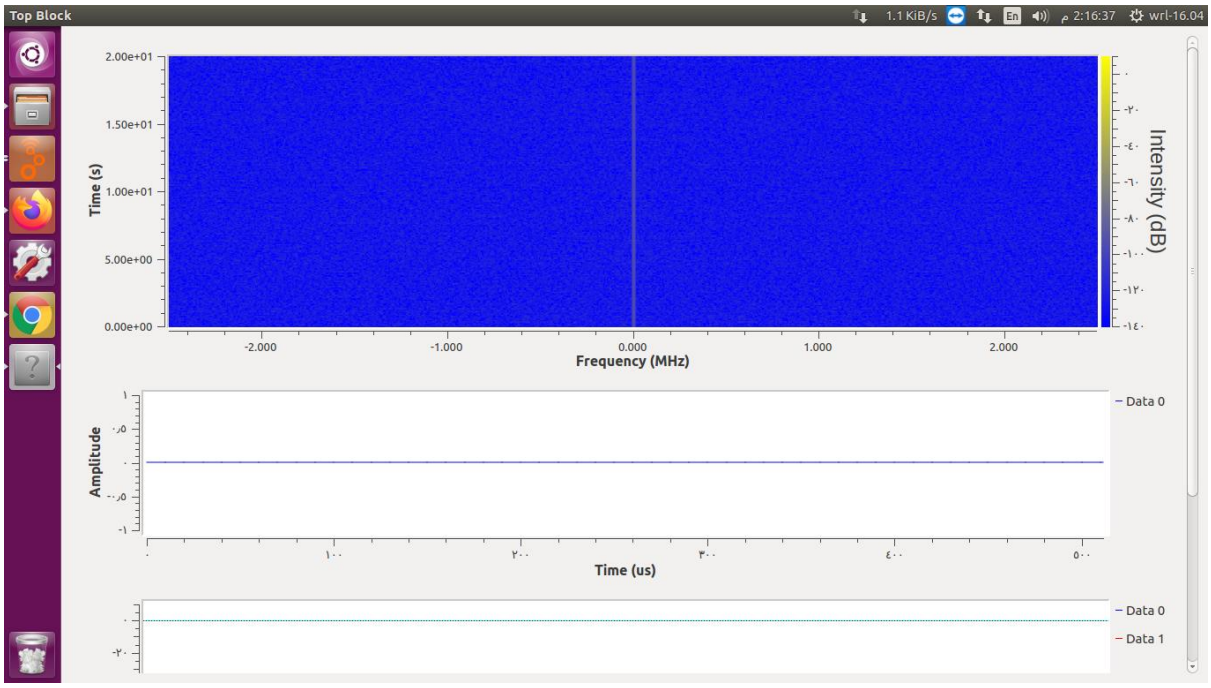


Figure 57 No User

But if we found an user we didn't take this channel like that graph

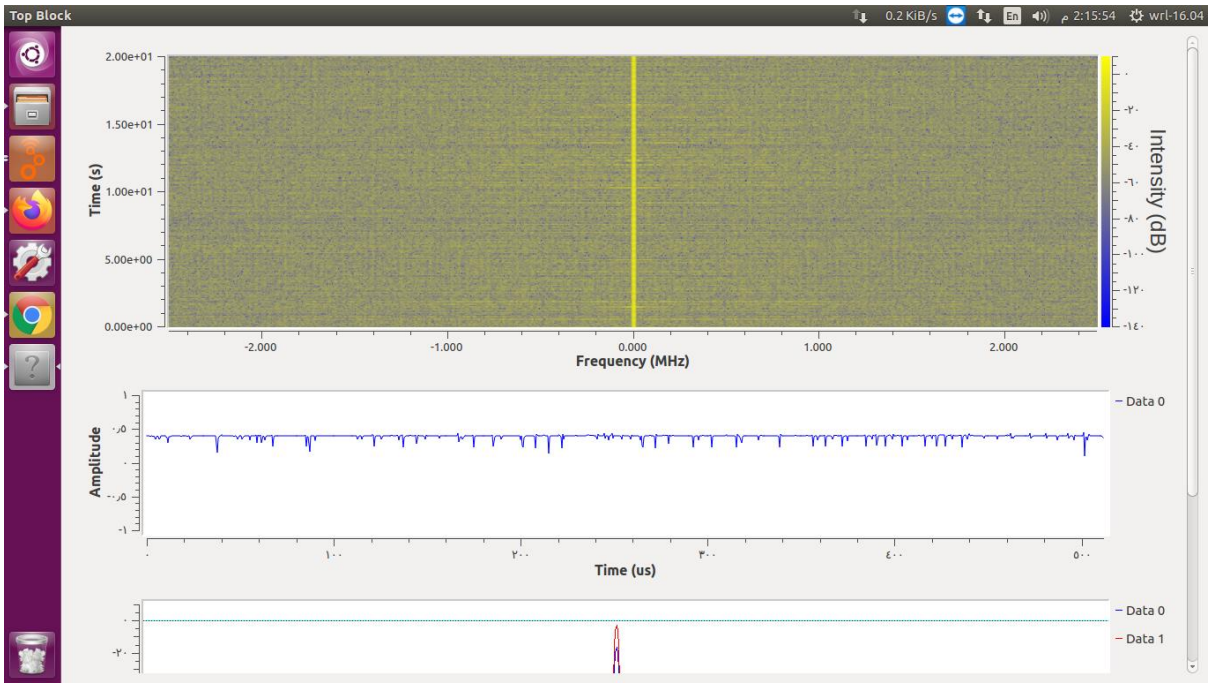


Figure 58: PU found

And in special cases we found an holes or PU start or stop sending data

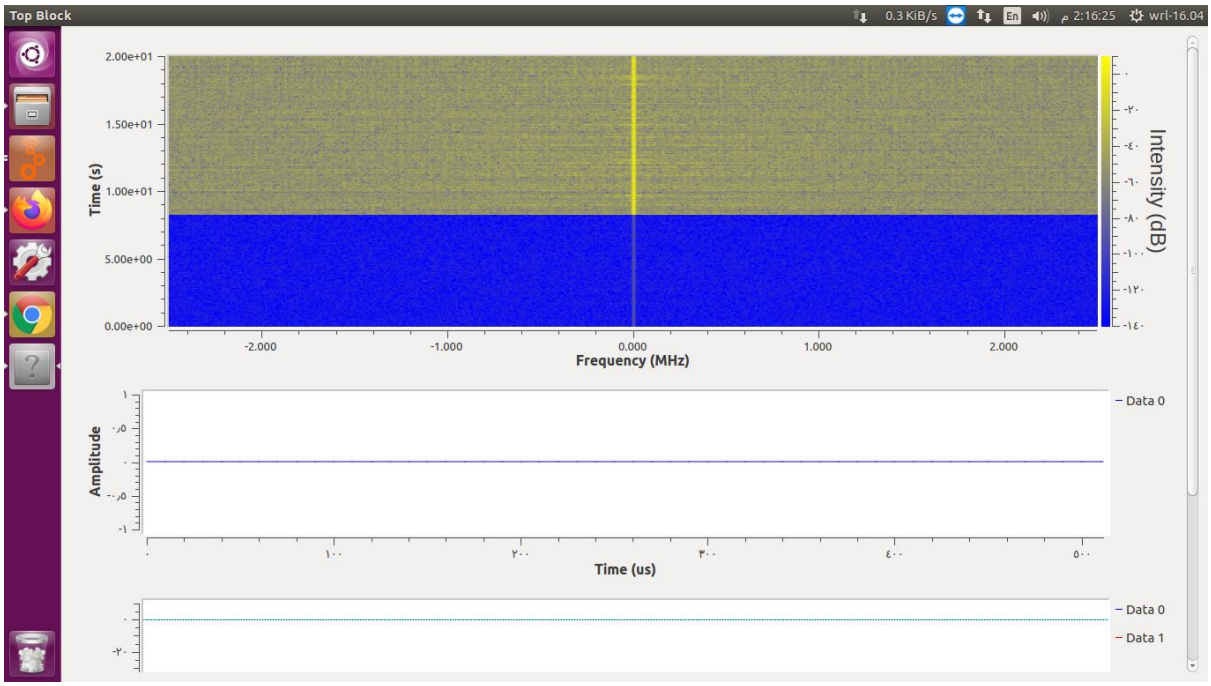


Figure 59 Pu stop send a data

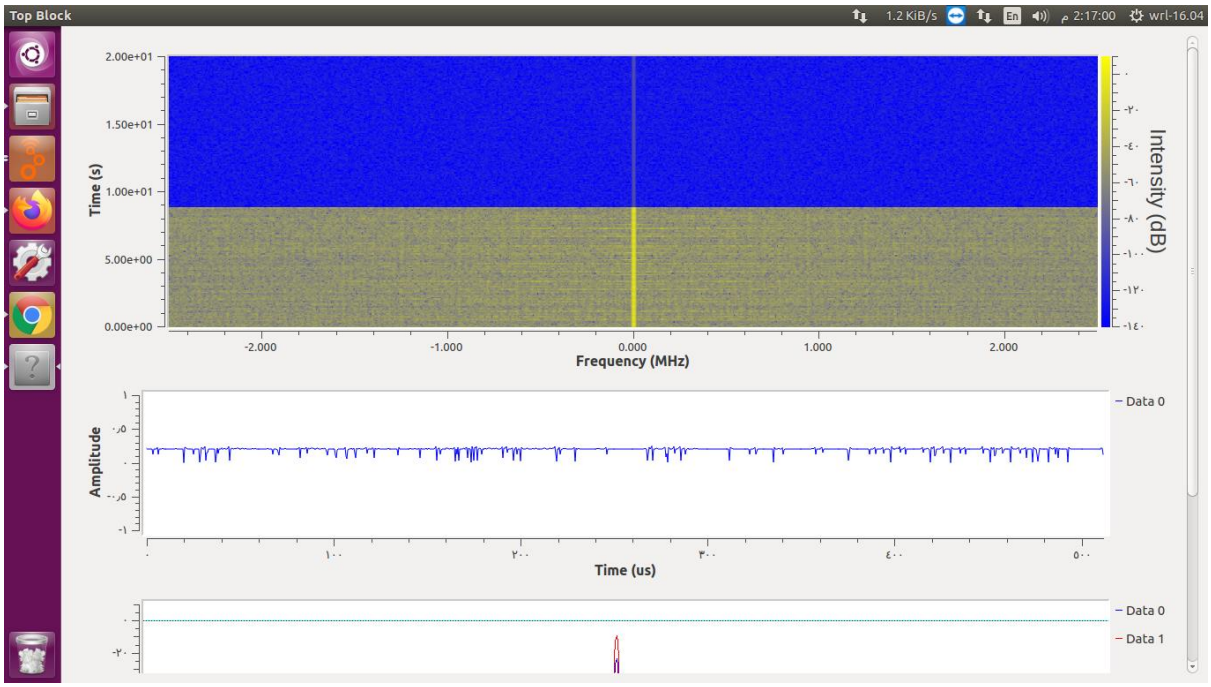


Figure 60 PU start send a data

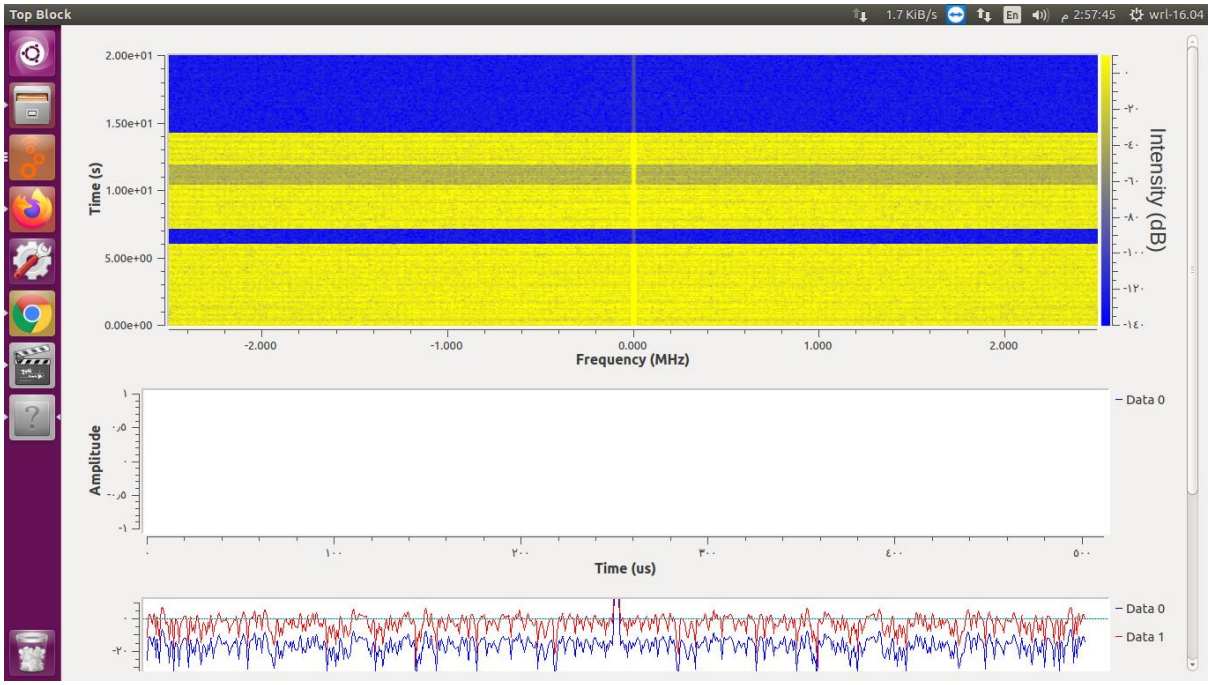


Figure 61 hole in channel

Then we transmit the array to common channel using GMSK modulation like a design

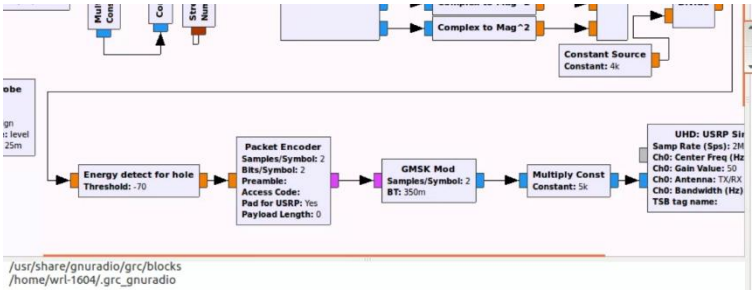


Figure 62 transmit the array to common channel

```
*epy_block_1_d2rHu_.py   Test GNU (1).txt   epy_block_1_yrVdtg.py
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, threshold=1.0):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Energy detect for hole',   # will show up in GRC
            in_sig=[np.float32],
            out_sig=[np.float32]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).
        self.threshold = threshold

    def work(self, input_items, output_items):
        for i in range (0,len(input_items[0])):
                if input_items[0][i] > threshold:
                    return False
                else:
                  input_items[0][i] <= threshold
                  return True

```

Then we transmit the array that contain channels that we found it Free by python code

to a common channel

9.4 Common channel:

The rule of common channel is get all arrays of the CR's users and compare it to find a common holes and broadcast the new array to CR's users

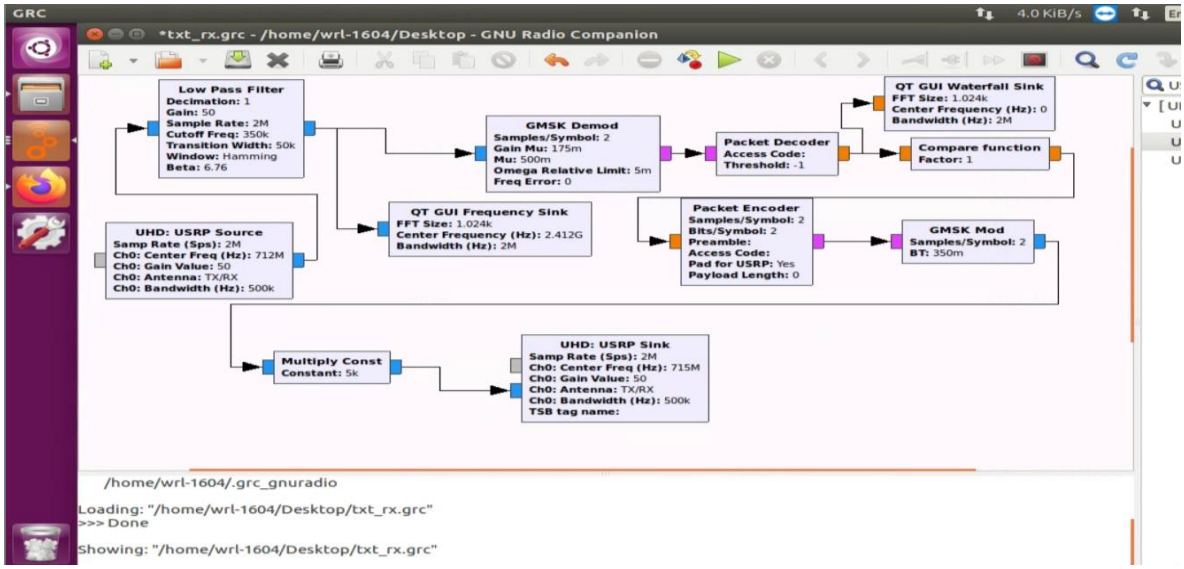


Figure 63 Common channel

Then we receive common holes in Transceiver and start prepare to send or receive a data

And that is the code of common channel and output illustrate the channel that

```
import numpy as np
input1 = [True, False, True,True,False,True,False,True,True,False,True, False, True,True,False,True,False,True,True,False]
input2 = [False, True, True,True,False,False, True, True,True,False,False, True, True,True,False,False, True, True,True,False]
output = np.logical_and(input1,input2)
import numpy as np
f1 = 2400000000
f2 = 2420000000
step = 600000
freqArray = np.arange(f1, f2, step).tolist()
print(freqArray,len(freqArray))

freqInput = freqArray
freqOutput = []
for index, i in enumerate(output):
    if i == True:
        freqOutput.append(freqInput[index])
        print(freqInput[index])

print(output)
print(freqOutput)
```

```
[2400000000, 2400600000, 2401200000, 2401800000, 2402400000, 2403000000, 2403600000, 2404200000, 2404800000, 2405400000, 2406000000, 2406600000, 2407200000, 2407800000, 2408400000, 2409000000, 2409600000, 2
2401200000
2401800000
2404200000
2404800000
2407200000
2407800000
2410200000
2410800000
[False False  True  True False False False  True  True False False False
  True  True False False False  True  True False]
[2401200000, 2401800000, 2404200000, 2404800000, 2407200000, 2407800000, 2410200000, 2410800000]
```

Snipping Tool | 0s completed at 2:12 PM

has empty from PU

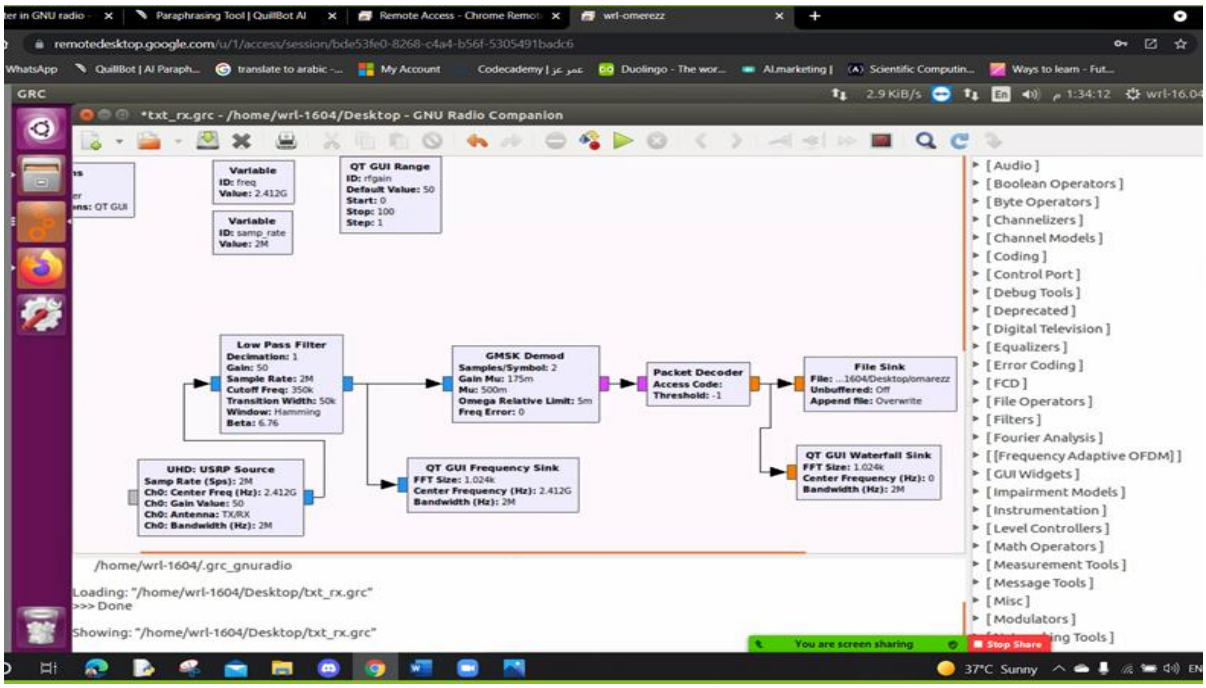


Figure 64 receive common holes

9.5 Transceiver:

1. which gain array that collect a common hole and select a random hole to prepare a transceiver

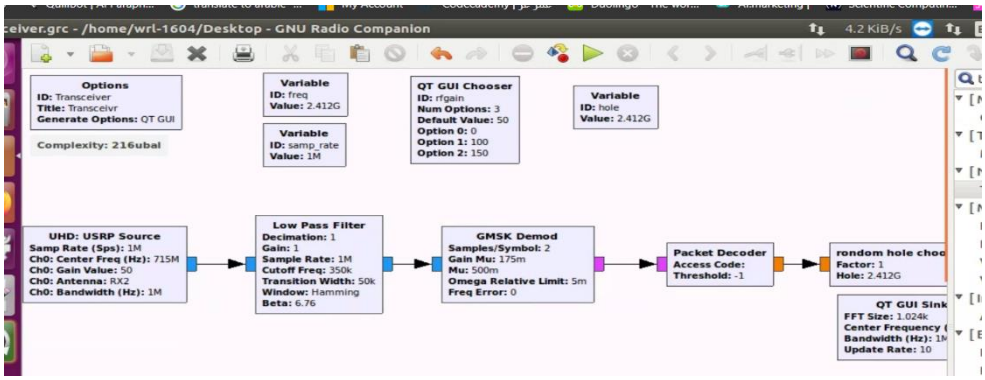
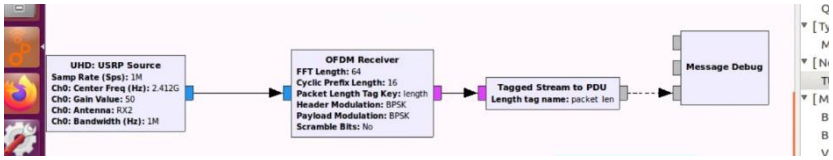


Figure 64: design of common holes and select random hole

2. sensing a common hole and receive a message if any one sent it in common hole to me



:

receive from common hole if anyone sent message to me 66 Figure

3. Check if user want to sent and select a random hole to send a message

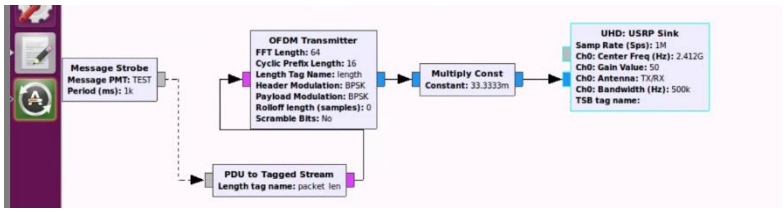


Figure66 sent a message a random hole 67 Figure

10 NI USRP.

Universal Software Radio Peripheral (USRP) is a hardware that has real-time data transmission ability. USRP devices are used in various communication studies. USRP devices ease the learning, teaching and research processes of communication systems. Due to suitability of the USRP-2901 device to LabVIEW & USRP B200 device to GNU, implementation of SDR systems is quite easy. USRP device operates in 2.4 GHz-2.5 GHz and 4.9 GHz-5.9 GHz frequency bands. Moreover, it has half-duplex characteristic and has good hardware features, can support instantaneous bandwidth up to 20 MHz and I/Q sampling rate up to 25 MS/s.

❖ **Transmitter**:

⇒ Frequency range 70 MHz to 6 GHz

⇒ Frequency step < 1 KHz**.**

⇒ Maximum output power (Pout) 20 dBm.

⇒ Gain range 89.75 dB.

⇒ Gain step 0.25 dB.

⇒ Frequency accuracy 2.5 ppm.

⇒ Maximum instantaneous real-time bandwidth 56 MHz

⇒ Maximum I/Q rate: Streaming 15 MS/s.

⇒ Burst: (One channel 61.44 MS/s) & (Two channels 30.72 MS/s).

⇒ Digital-to-analog converter (DAC) 12 bits.

❖ **Receiver**:

⇒ Frequency range 70 MHz to 6 GHz

⇒ Frequency step < 1 KHz**.**

⇒ Gain range 76 dB.

⇒ Gain step 1dB.

⇒ Maximum input power (Pin) -15 dBm.

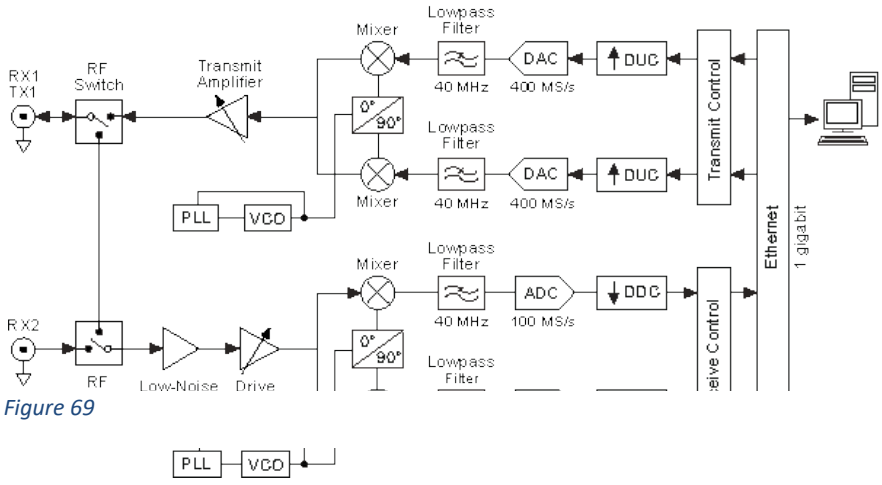


Figure 69

Figure 68 USRP

11 Summary and Conclusions.

Cognitive Radio is a very intelligent wireless that help us to solve the problem of spectrum scarcity by using optimistic spectrum sharing techniques.

CR is able to sense the dynamically changing environment by using spectrum sensing techniques (Cooperative & Transmitter detection & Interference temperature detection).

Here in our project, we are trying to implement the cognitive radio (CR) and we are trying to overcome a problem that is one of the most serious problems that can face the cognitive radio, which is the loss of the path, due to which the receiver owned by the other user I want to call cannot detect the signal sent to him by my transmitter due to shadowing or fading or attenuation.

To solve this problem we discovered two efficient solutions which are:

- Use a common eNodeB.
- Use a common channel which is some sub-carriers that's assigned to CR users by ISP or Providers like Vodafone or Orange or Etisalat.

We discovered the first solution is very complex and expensive but more efficient and the second solution is less complex and cheaper than the first solution so we Therefore, we here into this project try to implement the second solution which is "common channel".

Cognitive Radio can be done by using Software Defined Radio (SDR) where it can be applied, implemented and programmed using GNU or LabVIEW. We use in programming some software languages such as C, C++, and Python. So in this project, we are trying to implement cognitive radio in two directions (LabVIEW and GNU) where a logical approach is taken to implement our CR in both directions.

12 References.

- Danijela Cabric, Shridhar Mubaraq Mishra, Robert W. Brodersen, "Implementation Issues in Spectrum Sensing for Cognitive Radios," *Asilomar Conference on Signals, Systems, and Computers*, 2004. November 2004. Page(s): 772 - 776 Vol.1.

- I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next Generation Dynamic Spectrum Access Cognitive Radio Wireless Networks: A Survey," *The International Journal of Computer and Telecommunications Networking*. 2006. vol. 50, pp. 2127 – 2159.

- International Telecommunication Union (ITU). *Handbook Frequencyadaptive communication systems and networks in the MF/HF bands*. Edition 2002.

- J. Mitola III, "Cognitive Radio: An Integrated Agent Architecture for

Software Defined Radio," Ph.D. thesis, KTH- Royal Institute of Technology, Stockholm, Sweden, 2000.

- Gianmarco Baldini et. Al, "Reconfigurable Radio Systems for Public Safety Based on Low-Cost Platforms," *EuroISI 2008*, LNCS 5376, pp. 237–247, 2008.