Distributed Spectrum Monitoring and Channel Sounding

Khyati Dinesh Patel, Mingkun Sun

Introduction

Khyati Dinesh Patel
Graduate student at
Rutgers University

Mingkun Sun (No photos)

UG student currently studying ECE

About our Project:

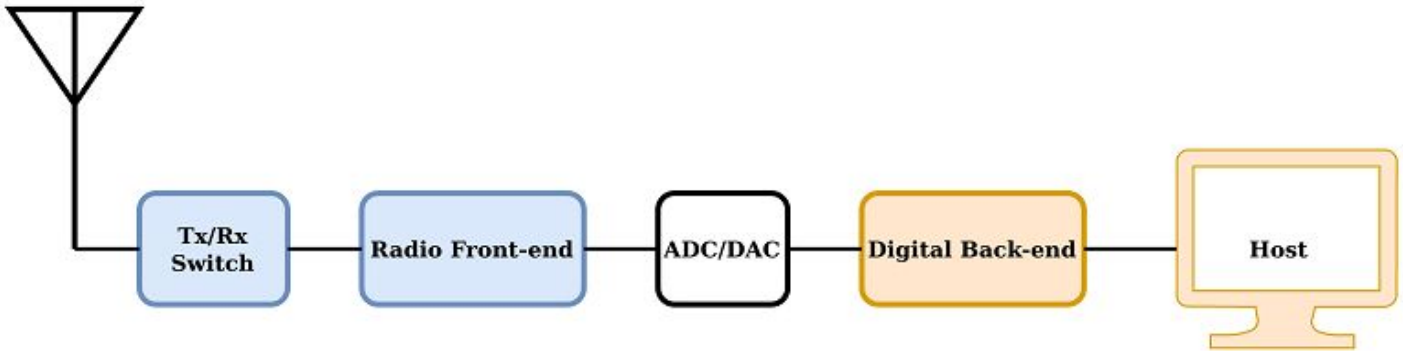
- Use a collection of software defined radios (SDRs) to detect spectrum occupancy and perform channel usage coordination among number of radio systems

The goals for our projects are:

- Using nodes in COSMO/ORBIT testbed to transmit reference signals
- Collect data from the transmission process with the portable SDRs
- Use both traditional and AI/ML analysis for channel characterization (including occupancy)

What is SDR?

- Collection of hardware and software technologies where some or all of the radio's operating functions are implemented through modifiable software or firmware operating on programmable processing technologies



Why Spectrum Monitoring ?

- Interference can significantly degrade the quality of services
- Facilitate the identification and removal of illegal or unlicensed interference signals
- Monitoring frequencies provides the information needed to optimize spectrum for maximum utilization.

USRP and UHD

Universal Software Radio Peripheral (USRP): Software-defined RF architecture used to design, prototype, and deploy wireless systems with custom signal processing

USRP Hardware Driver (UHD): Is a software API (Application Program Interface) that supports application development on all USRP SDR products

Experiment 1:
Working with USRP 2

- Transmit and receive a single frequency over the air to demonstrate the use of Universal software Radio peripheral Hardware Drivers (UHD)
- Consists of two nodes (node1-1 & node1-2) and each node has a USRP2 connection via Ethernet

Results:

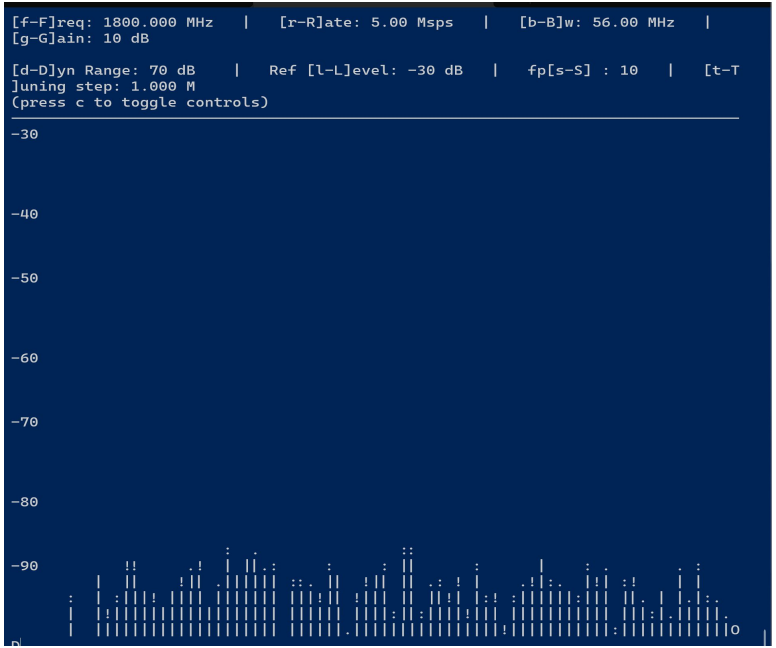
- Generated a SINE wave
- Plotted a frequency spectrum in the terminal

```
Setting TX Rate: 5.000000 Msps ...
[INFO] [B200] Asking for clock rate 40.000000 MHz ...
[INFO] [B200] Actually got clock rate 40.000000 MHz.
Actual TX Rate: 5.000000 Msps ...

Setting TX Freq: 1800.000000 MHz ...
Setting TX LO Offset: 0.000000 MHz ...
Actual TX Freq: 1800.000000 MHz ...

Setting TX Gain: 10.000000 dB ...
Actual TX Gain: 10.000000 dB ...

Setting device timestamp to 0 ...
Checking TX: LO: locked  ...
Press Ctrl + C to stop streaming ...
^C
Done!
```



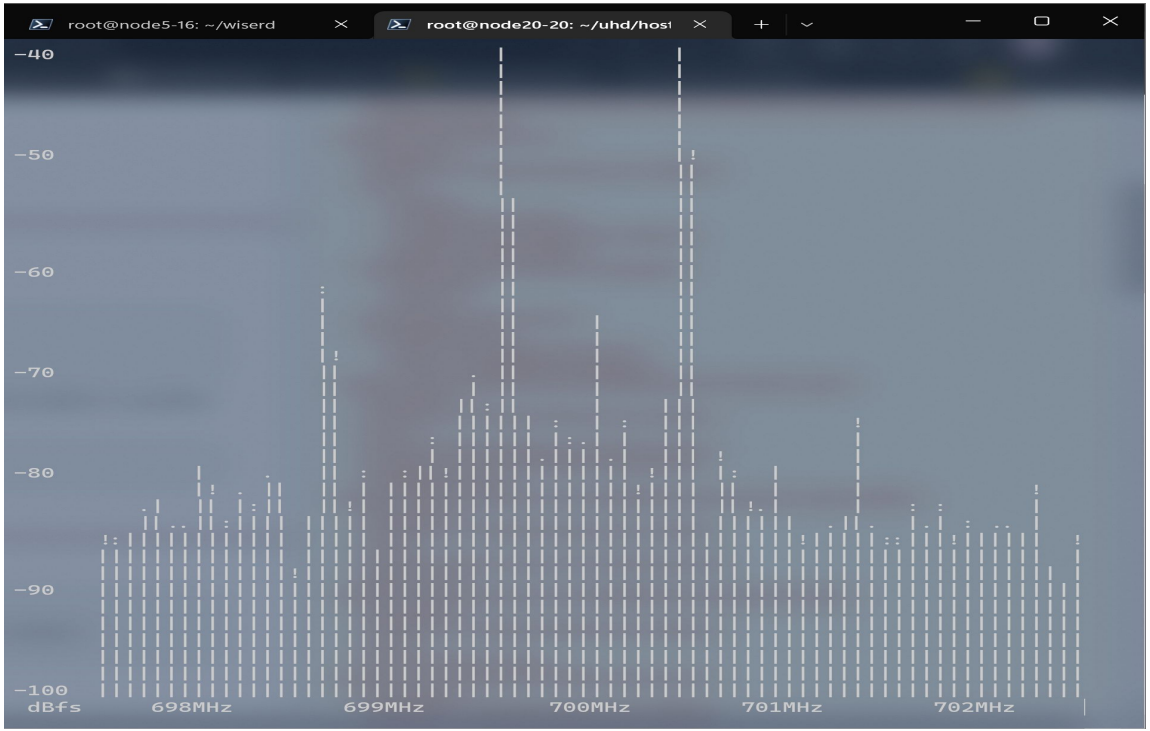
Experiment 2:
Using 2 transmitter and 2 receiver

- Used grid to perform the experiment
- We used USRP with 2 TX/RX and 2 RX2 antennas such as X310 and B210

Experiment 3:
Spectrum sensing with USRP2 and wiserd (OEDL and OML)

- We used USRP2 to send sine wave from transmitter to receiver
- Loaded multiantennatutorial.ndz image into the nodes
- Plotted a spectrum of sine wave on the receiver end

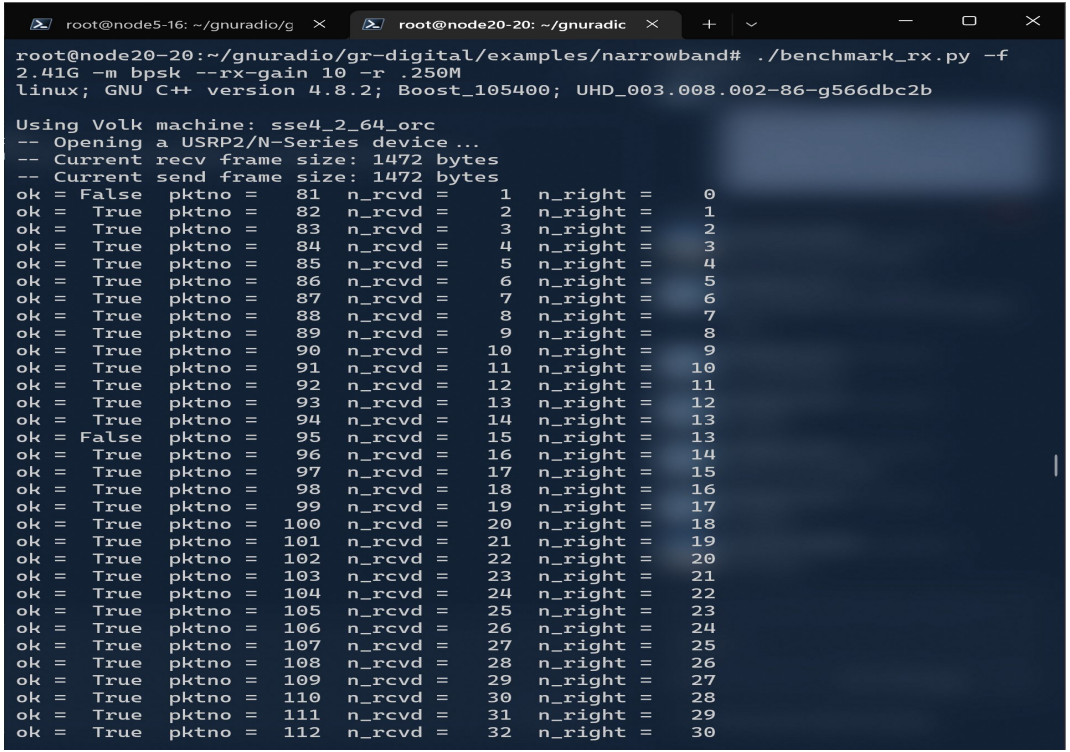
Results:


root@node5-16: ~/wiserd
root@node20-20: ~/uhd/hos
-40
-50
-60
-70
-80
-90
-100
dBfs
698MHz
699MHz
700MHz
701MHz
702MHz

Experiment 4:
Simple radio example with GNURADIO benchmark scripts

- We used USRP2 to show a simple transfer of packets between 2 nodes
- Loaded multiantennatutorial.ndz image into the nodes
- Utilized GNU Radio benchmarks scripts to transfer packets over a radio link between two nodes
- This experiment had a single transmitter and receiver

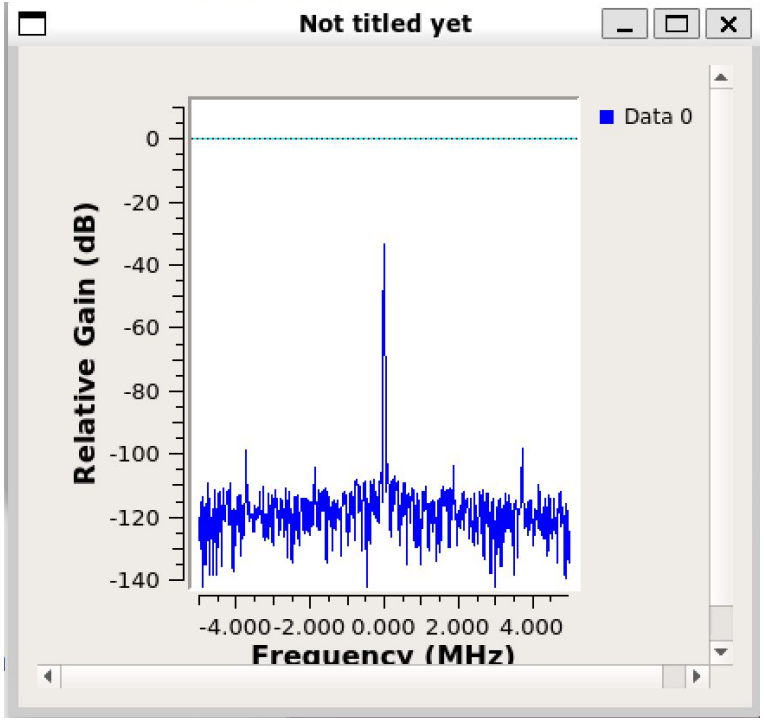
Results:

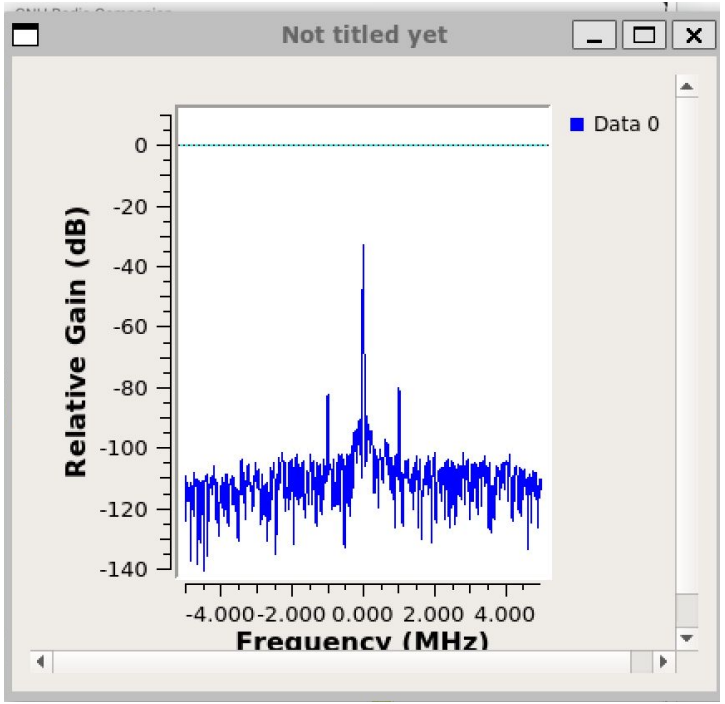

root@node5-16: ~/gnuradio/g
root@node20-20: ~/gnuradic
root@node20-20:~/gnuradio/gr-digital/examples/narrowband# ./benchmark_rx.py -f 2.41G -m bpsk --rx-gain 10 -r .250M
linux; GNU C++ version 4.8.2; Boost_105400; UHD_003.008.002-86-g566dbc2b
Using Volk machine: sse4_2_64_orc
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
ok = False pktno = 81 n_rcvd = 1 n_right = 0
ok = True pktno = 82 n_rcvd = 2 n_right = 1
ok = True pktno = 83 n_rcvd = 3 n_right = 2
ok = True pktno = 84 n_rcvd = 4 n_right = 3
ok = True pktno = 85 n_rcvd = 5 n_right = 4
ok = True pktno = 86 n_rcvd = 6 n_right = 5
ok = True pktno = 87 n_rcvd = 7 n_right = 6
ok = True pktno = 88 n_rcvd = 8 n_right = 7
ok = True pktno = 89 n_rcvd = 9 n_right = 8
ok = True pktno = 90 n_rcvd = 10 n_right = 9
ok = True pktno = 91 n_rcvd = 11 n_right = 10
ok = True pktno = 92 n_rcvd = 12 n_right = 11
ok = True pktno = 93 n_rcvd = 13 n_right = 12
ok = True pktno = 94 n_rcvd = 14 n_right = 13
ok = False pktno = 95 n_rcvd = 15 n_right = 13
ok = True pktno = 96 n_rcvd = 16 n_right = 14
ok = True pktno = 97 n_rcvd = 17 n_right = 15
ok = True pktno = 98 n_rcvd = 18 n_right = 16
ok = True pktno = 99 n_rcvd = 19 n_right = 17
ok = True pktno = 100 n_rcvd = 20 n_right = 18
ok = True pktno = 101 n_rcvd = 21 n_right = 19
ok = True pktno = 102 n_rcvd = 22 n_right = 20
ok = True pktno = 103 n_rcvd = 23 n_right = 21
ok = True pktno = 104 n_rcvd = 24 n_right = 22
ok = True pktno = 105 n_rcvd = 25 n_right = 23
ok = True pktno = 106 n_rcvd = 26 n_right = 24
ok = True pktno = 107 n_rcvd = 27 n_right = 25
ok = True pktno = 108 n_rcvd = 28 n_right = 26
ok = True pktno = 109 n_rcvd = 29 n_right = 27
ok = True pktno = 110 n_rcvd = 30 n_right = 28
ok = True pktno = 111 n_rcvd = 31 n_right = 29
ok = True pktno = 112 n_rcvd = 32 n_right = 30

Experiment 5:
Working with USRP X310

- Used two USRP X310s on ORBIT Sandbox 2 to transmit and receive a single frequency over the air
- This experiment demonstrates the use of the **U**SRP **H**ardware **D**rivers (UHD) and GNU Radio with the USRP X310
- Used GNU companion to plot the spectrum of the received signal

Results:

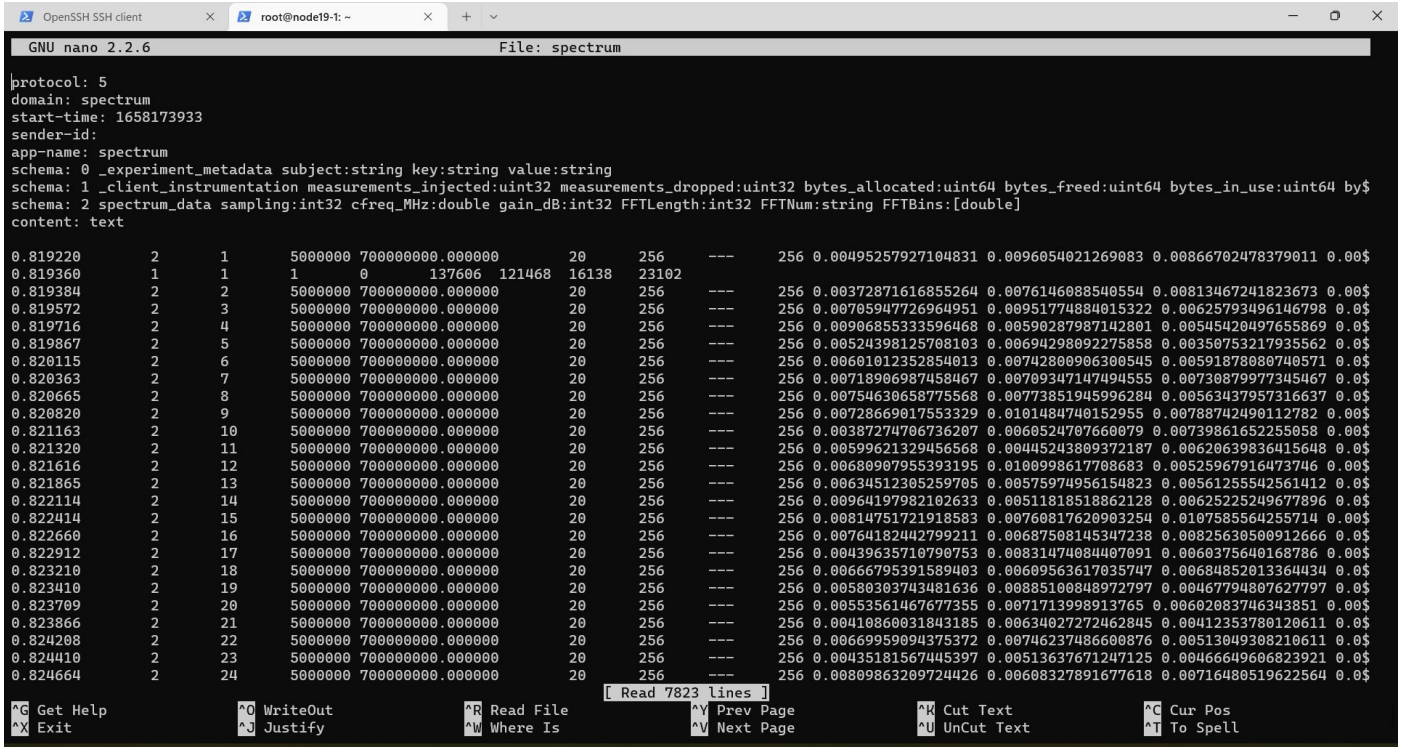

Not titled yet
Data 0
Relative Gain (dB)
0
-20
-40
-60
-80
-100
-120
-140
-4.000 -2.000 0.000 2.000 4.000
Frequency (MHz)


Not titled yet
Data 0
Relative Gain (dB)
0
-20
-40
-60
-80
-100
-120
-140
-4.000 -2.000 0.000 2.000 4.000
Frequency (MHz)

Data Collection: For spectrum sensing using USPR2 and WISERD (OEDL and OML)

- Used WISERD to record this spectrum data
- Wiserd recorded the spectrum data to file for 2000ms

Sample Data:


OpenSSH SSH client
root@node19-1: ~
GNU nano 2.2.6 File: spectrum
protocol: 5
domain: spectrum
start-time: 1658173933
sender-id:
app-name: spectrum
schema: 0 _experiment_metadata subject:string key:string value:string
schema: 1 _client_instrumentation measurements_injected:uint32 measurements_dropped:uint32 bytes_allocated:uint64 bytes_freed:uint64 bytes_in_use:uint64 by$
schema: 2 spectrum_data sampling:int32 cfreq_MHz:double gain_dB:int32 FFTLength:int32 FFTNum:string FFTBins:[double]
content: text
0.819220 2 1 5000000 700000000.000000 20 256 --- 256 0.00495257927104831 0.0096054021269083 0.00866702478379011 0.00$
0.819360 1 1 1 0 137606 121468 16138 23102
0.819384 2 2 5000000 700000000.000000 20 256 --- 256 0.00372871616855264 0.00761460885540554 0.00813467241823673 0.00$
0.819572 2 3 5000000 700000000.000000 20 256 --- 256 0.00705947726964951 0.00951774884015322 0.00625793496146798 0.0$
0.819716 2 4 5000000 700000000.000000 20 256 --- 256 0.00906855333596468 0.00590287987142801 0.00545420497655869 0.0$
0.819867 2 5 5000000 700000000.000000 20 256 --- 256 0.00524398125708103 0.00694298092275858 0.00350753217935562 0.0$
0.820115 2 6 5000000 700000000.000000 20 256 --- 256 0.00601012352854013 0.00742800906300545 0.00591878080740571 0.0$
0.820363 2 7 5000000 700000000.000000 20 256 --- 256 0.00718906987458467 0.00709347147494555 0.00730879977345467 0.0$
0.820665 2 8 5000000 700000000.000000 20 256 --- 256 0.00754630658775568 0.00773851945996284 0.00563437957316637 0.0$
0.820820 2 9 5000000 700000000.000000 20 256 --- 256 0.00728669017553329 0.0101484740152955 0.00788742490112782 0.00$
0.821163 2 10 5000000 700000000.000000 20 256 --- 256 0.00387274706736207 0.0060524707660079 0.00739861652255058 0.00$
0.821320 2 11 5000000 700000000.000000 20 256 --- 256 0.00599621329456568 0.00445243809372187 0.00620639836415648 0.0$
0.821616 2 12 5000000 700000000.000000 20 256 --- 256 0.00680907955393195 0.0100998617708683 0.00525967916473746 0.00$
0.821865 2 13 5000000 700000000.000000 20 256 --- 256 0.00634512305259705 0.00575974956154823 0.00561255542561412 0.0$
0.822114 2 14 5000000 700000000.000000 20 256 --- 256 0.00964197982102633 0.00511818518862128 0.00625225249677896 0.0$
0.822414 2 15 5000000 700000000.000000 20 256 --- 256 0.00814751721918583 0.00760817620903254 0.0107585564255714 0.00$
0.822660 2 16 5000000 700000000.000000 20 256 --- 256 0.00764182442799211 0.00687508145347238 0.00825630500912666 0.0$
0.822912 2 17 5000000 700000000.000000 20 256 --- 256 0.00439635710790753 0.00831474084407091 0.0060375640168786 0.00$
0.823210 2 18 5000000 700000000.000000 20 256 --- 256 0.00666795391589403 0.00609563617035747 0.00684852013364434 0.0$
0.823410 2 19 5000000 700000000.000000 20 256 --- 256 0.00580303743481636 0.00885100848972797 0.00467794807627797 0.0$
0.823709 2 20 5000000 700000000.000000 20 256 --- 256 0.00553561467677355 0.0071713998913765 0.00602083746343851 0.00$
0.823866 2 21 5000000 700000000.000000 20 256 --- 256 0.00410860031843185 0.00634027272462845 0.00412353780120611 0.0$
0.824208 2 22 5000000 700000000.000000 20 256 --- 256 0.00669959094375372 0.00746237486600876 0.00513049308210611 0.0$
0.824410 2 23 5000000 700000000.000000 20 256 --- 256 0.00435181567445397 0.00513637671247125 0.00466649606823921 0.0$
0.824664 2 24 5000000 700000000.000000 20 256 --- 256 0.00809863209724426 0.00608327891677618 0.00716480519622564 0.0$
[Read 7823 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Data Collection: Using Spectrum sensing with USRP2 and wiserd

- Again in this experiment we used WISERD to collect the data
- It created a data file in the root
- Used OCTAVE plot received data file
- .bin file that WISERD created and we use this file to plot received data file

Sample Data:


OpenSSH SSH client
root@node19-1: ~
GNU nano 2.2.6
File: rx_signal_freq700000000_rate5000000_gain20.bin
Modified
[Read 3173 lines]
^G Get Help
^O WriteOut
^R Read File
^Y Prev Page
^K Cut Text
^C Cur Pos
^X Exit
^J Justify
^W Where Is
^V Next Page
^U UnCut Text
^T To Spell

Future Work

- With the collected data, use AI and ML approaches to build intelligent networks, in order to adapt their spectral behaviour

- Based on the behavior of different wireless user, use algorithm to assign limit spectrum capacity in a more efficient way

Questions

Thank you