CSE3000

Summer Practice I

Report

Student Name: Oğuzhan BÖLÜKBAŞ

Student ID: 150114022

1. About the WIreless Systems, Networks and CyberSecurity (WINS) Lab

Established on the April Fools' Day of 2016, WINS lab strives to conduct research in the field of computer networks, wireless networks, mobile systems and security thereof. We aim at building efficient and dependable solutions for the networks of the future and focus on the design and experimentation of systems and protocols. We support education both at undergraduate and graduate levels.

The specific topics we study are

- 5G and Next Generation Mobile Networks,
- Internet of Things (IoT), Wireless Sensor Networks,
- Software Networks and Software-defined Networked Systems,
- Virtual Networks, Edge and Fog Computing,
- Cybersecurity and Network Security,
- Mobile Computing,
- Ubiquitous and Pervasive Computing, and
- Performance Evaluation of Networks.

Some recent projects are

- 2016-2017 METU-OYP, Software-defined Networked Systems Laboratory, 250.000TL (Proposer)
- 2016-2018 TÜBİTAK 1001, 215E127 Density-adaptive Wireless Networks (DAWN), 427.428 TL (Proposer)
- 2015-2017 TÜBİTAK 2232, 115C064 Internet as the Oracle, 29.000TL (Proposer)
- 2014-2015 BAP-08-11-2014-025 Software Development Platform for Heterogeneous Internet of Things, 30.000 TL (Proposer)

2. Technologies Used in the Training

During the training, we used the following technologies and environments.

a) GNU Radio

GNU Radio is a free software development toolkit that provides signal processing blocks to implement software-defined radios and signal-processing systems. It can be used with external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in hobbyist, academic, and commercial environments to support both wireless communications research and real-world radio systems.

Overview

The GNU Radio software provides the framework and tools to build and run software radio or just general signal-processing applications. The GNU Radio applications themselves are generally known as "flowgraphs", which are a series of signal processing blocks connected together, thus describing a data flow. As with all software-defined radio systems, reconfigurability is a key feature. Instead of using different radios designed for specific but disparate purposes, a single, general-purpose, radio can be used as the radio front-end, and the signal-processing software (here, GNU Radio), handles the processing specific to the radio application.

These flowgraphs can be written in either C++ or the Python programming language. The GNU Radio infrastructure is written entirely in C++, and many of the user tools are written in Python. [1]

b) Software-Defined Radio (SDR)

Software-defined radio (SDR) is a radio communication system where components that have been typically implemented in hardware
(e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system. While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which used to be only theoretically possible.

Overview

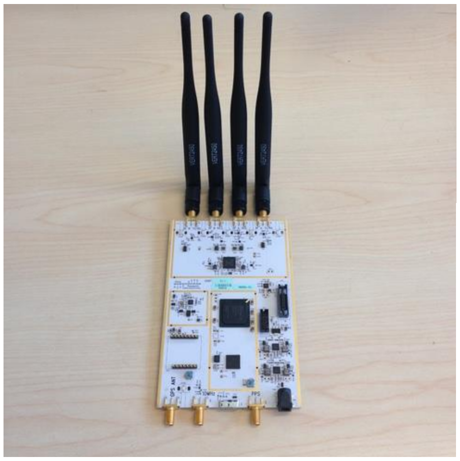
A basic SDR system may consist of a personal computer equipped with a sound card, or other analog-to-digital converter, preceded by some form of RF front end. Significant amounts of signal processing are handed over to the general-purpose processor, rather than being done in special-purpose hardware (electronic circuits). Such a design produces a radio which can receive and transmit widely different radio protocols (sometimes referred to as waveforms) based solely on the software used.

Software radios have significant utility for the military and cell phone services, both of which must serve a wide variety of changing radio protocols in real time.

In the long term, software-defined radios are expected by proponents like the SDRForum (now The Wireless Innovation Forum) to become the dominant technology in radio communications. SDRs, along with software defined antennas are the enablers of the cognitive radio. [2]

c) USRP B210

- ✓ USRP B210 SDR Kit - Dual Channel Transceiver (70 MHz - 6GHz) - Ettus Research
- ✓ Full duplex, MIMO (2 Tx & 2 Rx) operation with up to 56 MHz of real-time bandwidth (61.44MS/s quadrature)
- ✓ GNURadio and OpenBTS support through the open-source USRP Hardware *Driver™ (UHD)* [3]



3. Work Done

In the training, I worked on wireless communication with using USRP B210. Below, I describe the progress of wireless communication and work done in weeks.

a. Week 1 (12 – 16 JUNE 2017)

In this week, I spent most of the time to be adept to radio signals' world.

I installed gnuradio-companion program on my laptop and required packages on Kali Linux 2016.2 release with following steps in https://wiki.gnuradio.org/index.php/InstallingGR

To run the GNU Radio, we should *write "gnuradio-companion"* command on terminal without quotes.

I have an account on laboratory server and connect to server with using SSH (Secure Shell) Protocol from my home with typing the following command on terminal:

$ssh "my_username"@winscenter.ceng.metu.edu.tr –P *"Port_Number"*

with typing correct word and numbers without quotes. After connection, we can reach our files and see them on CLI(Command-Line Interface) screen.

In order to copy file from server account to our computer, we can write the following command on terminal:

$scp –P *"portNumber" "myUsername"@winscenter.ceng.m*etu.edu.tr:filename *"computerPath"*

An example:

$scp –P 1453 fatih@winscenter.ceng.metu.edu.tr:fetih.py /root/Downloads

In order to copy whole folder and its file(s) recursively:

$scp -r –P 1453 fatih@winscenter.ceng.metu.edu.tr:Gaza /root/Downloads

I have also studied both what is wireless communication and software-defined radio.

I have practice to use GNU Radio and making flowgraphs and learn object-oriented programming with Python because the GNU Radio flowgraphs are generate python codes and in order to make change on them, being familiar with python programming language will be helpful.

My written examples:

a) first.py

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

deneme.py

```python
import sys

def quote(s ):
    return " ’"+s+" ’"

print sys.argv
args = sys.argv [1:]
print args
args.sort()
mapped = map(quote , args)
print mapped
print ’{ ’+"|". join (mapped)+’} ’
```

b) oop1.py

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

oop1.py

```python
class Employee:

    def __init__(self, first, last, pay):

        self.first = first
        self.last = last
        self.pay = pay
        self.email = first + '.' + last + '@company.com'

    def fullName(self):

        return '{} {}'.format(self.first, self.last)

emp_1 = Employee('Ahmet', 'Alaca' , 50000)
emp_2 = Employee('Mehmet', 'Karaca', 60000)

print(emp_1.first)
print(emp_2.fullName())
```

c) oop2.py

```python

class Employee:

    raiseAmount = 1.04
    numberOfEmployees = 0

    def __init__(self, first, last, pay):

        self.first = first
        self.last = last
        self.pay = pay
        self.email = first + '.' + last + '@company.com'

        Employee.numberOfEmployees += 1

    def fullName(self):

        return '{} {}'.format(self.first, self.last)

    def applyIncrease(self):

        self.pay = int(self.pay * 1.04)

print(Employee.numberOfEmployees)
emp_1 = Employee('Ahmet', 'Alaca' , 50000)
print(emp_1.fullName())
print(Employee.numberOfEmployees)
emp_2 = Employee('Mehmet', 'Karaca', 60000)
print(emp_2.fullName())
print(Employee.numberOfEmployees)

print(emp_1.fullName())
print('Before increasing: ' + str(emp_1.pay))

emp_1.applyIncrease()

print('After increasing: ' + str(emp_1.pay))

```

b. Week 2 (19 – 23 JUNE)

I have design a FM (Frequency Modulation) radio receiver with using GNU Radio. Program code in python language and screenshots of the program are below:

Code:

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

fm_receiver.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: FM Receiver
# Generated: Mon Jul 17 14:39:13 2017
##################################################

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from PyQt4 import Qt
from gnuradio import analog
from gnuradio import audio
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio import qtgui
from gnuradio import uhd
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from gnuradio.qtgui import Range, RangeWidget
from optparse import OptionParser
import sip
import sys
import time
from gnuradio import qtgui


class fm_receiver(gr.top_block, Qt.QWidget):

    def __init__(self):
        gr.top_block.__init__(self, "FM Receiver")
        Qt.QWidget.__init__(self)
```

Line 51, Column 43 Tab Size: 4 Python

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

fm_receiver.py

```
    def __init__(self):
        gr.top_block.__init__(self, "FM Receiver")
        Qt.QWidget.__init__(self)
        self.setWindowTitle("FM Receiver")
        qtgui.util.check_set_qss()
        try:
            self.setWindowIcon(Qt.QIcon.fromTheme('gnuradio-grc'))
        except:
            pass
        self.top_scroll_layout = Qt.QVBoxLayout()
        self.setLayout(self.top_scroll_layout)
        self.top_scroll = Qt.QScrollArea()
        self.top_scroll.setFrameStyle(Qt.QFrame.NoFrame)
        self.top_scroll_layout.addWidget(self.top_scroll)
        self.top_scroll.setWidgetResizable(True)
        self.top_widget = Qt.QWidget()
        self.top_scroll.setWidget(self.top_widget)
        self.top_layout = Qt.QVBoxLayout(self.top_widget)
        self.top_grid_layout = Qt.QGridLayout()
        self.top_layout.addLayout(self.top_grid_layout)

        self.settings = Qt.QSettings("GNU Radio", "fm_receiver")
        self.restoreGeometry(self.settings.value("geometry").toByteArray())

        ##################################################
        # Variables
        ##################################################
        self.samp_rate = samp_rate = 32e6
        self.gain = gain = 50
        self.freq = freq = 93e6
        self.audio_rate = audio_rate = 48e3
        self.audio_interp = audio_interp = 4

        ##################################################
        # Blocks
        ##################################################
        self._gain_range = Range(0, 100, 1, 50, 200)
        self._gain_win = RangeWidget(self._gain_range, self.set_gain, 'gain', "c
        self.top_layout.addWidget(self._gain_win)
        self._freq_range = Range(87e5, 108e7, 1e5, 93e6, 200)
        self._freq_win = RangeWidget(self._freq_range, self.set_freq, 'freq', "c
        self.top_layout.addWidget(self._freq_win)
```

Line 90, Column 56 Tab Size: 4 Python

```python
        self.top_layout.addWidget(self._freq_win)
        self.uhd_usrp_source_0 = uhd.usrp_source(
            ",".join(("", "")),
            uhd.stream_args(
                cpu_format="fc32",
                channels=range(1),
            ),
        )
        self.uhd_usrp_source_0.set_samp_rate(samp_rate)
        self.uhd_usrp_source_0.set_center_freq(freq, 0)
        self.uhd_usrp_source_0.set_gain(gain, 0)
        self.uhd_usrp_source_0.set_antenna('TX/RX', 0)
        self.uhd_usrp_source_0.set_bandwidth(samp_rate, 0)
        self.rational_resampler_xxx_0 = filter.rational_resampler_ccc(
                interpolation=int(audio_rate * audio_interp),
                decimation=int(samp_rate),
                taps=None,
                fractional_bw=None,
        )
        self.qtgui_sink_x_0 = qtgui.sink_c(
            1024, #fftsize
            firdes.WIN_BLACKMAN_hARRIS, #wintype
            0, #fc
            samp_rate, #bw
            "QT GUI PLOT", #name
            True, #plotfreq
            True, #plotwaterfall
            True, #plottime
            True, #plotconst
        )
        self.qtgui_sink_x_0.set_update_time(1.0/10)
        self._qtgui_sink_x_0_win = sip.wrapinstance(self.qtgui_sink_x_0.pyqwidge
        self.top_layout.addWidget(self._qtgui_sink_x_0_win)

        self.qtgui_sink_x_0.enable_rf_freq(False)



        self.blocks_wavfile_sink_0 = blocks.wavfile_sink('fm_music', 1, int(48e3
        self.audio_sink_0 = audio.sink(int(audio_rate), '', False)
        self.analog_wfm_rcv_0 = analog.wfm_rcv(
            quad_rate=audio_rate * audio_interp,
```

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

fm_receiver.py

```python
            quad_rate=audio_rate * audio_interp,
            audio_decimation=audio_interp,
        )

        ##################################################
        # Connections
        ##################################################
        self.connect((self.analog_wfm_rcv_0, 0), (self.audio_sink_0, 0))
        self.connect((self.analog_wfm_rcv_0, 0), (self.blocks_wavfile_sink_0, 0)
        self.connect((self.rational_resampler_xxx_0, 0), (self.analog_wfm_rcv_0,
        self.connect((self.uhd_usrp_source_0, 0), (self.qtgui_sink_x_0, 0))
        self.connect((self.uhd_usrp_source_0, 0), (self.rational_resampler_xxx_0

    def closeEvent(self, event):
        self.settings = Qt.QSettings("GNU Radio", "fm_receiver")
        self.settings.setValue("geometry", self.saveGeometry())
        event.accept()

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.uhd_usrp_source_0.set_samp_rate(self.samp_rate)
        self.uhd_usrp_source_0.set_bandwidth(self.samp_rate, 0)
        self.qtgui_sink_x_0.set_frequency_range(0, self.samp_rate)

    def get_gain(self):
        return self.gain

    def set_gain(self, gain):
        self.gain = gain
        self.uhd_usrp_source_0.set_gain(self.gain, 0)


    def get_freq(self):
        return self.freq

    def set_freq(self, freq):
        self.freq = freq
        self.uhd_usrp_source_0.set_center_freq(self.freq, 0)
```

Line 110, Column 10 Tab Size: 4 Python

File Edit Selection Find View Goto Tools Project Preferences Help Arduino

fm_receiver.py

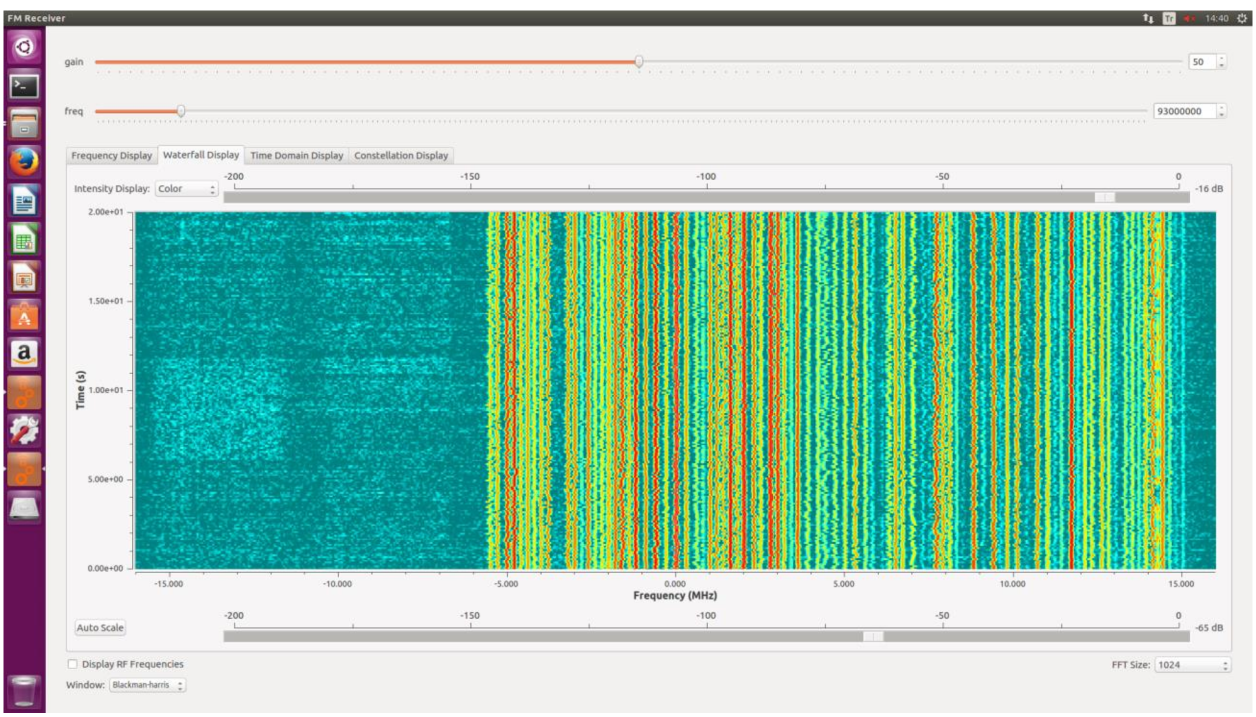
```python

    def get_audio_rate(self):
        return self.audio_rate

    def set_audio_rate(self, audio_rate):
        self.audio_rate = audio_rate

    def get_audio_interp(self):
        return self.audio_interp

    def set_audio_interp(self, audio_interp):
        self.audio_interp = audio_interp


def main(top_block_cls=fm_receiver, options=None):

    from distutils.version import StrictVersion
    if StrictVersion(Qt.qVersion()) >= StrictVersion("4.5.0"):
        style = gr.prefs().get_string('qtgui', 'style', 'raster')
        Qt.QApplication.setGraphicsSystem(style)
    qapp = Qt.QApplication(sys.argv)

    tb = top_block_cls()
    tb.start()
    tb.show()

    def quitting():
        tb.stop()
        tb.wait()
    qapp.connect(qapp, Qt.SIGNAL("aboutToQuit()"), quitting)
    qapp.exec_()


if __name__ == '__main__':
    main()

```

Line 149, Column 24 Tab Size: 4 Python

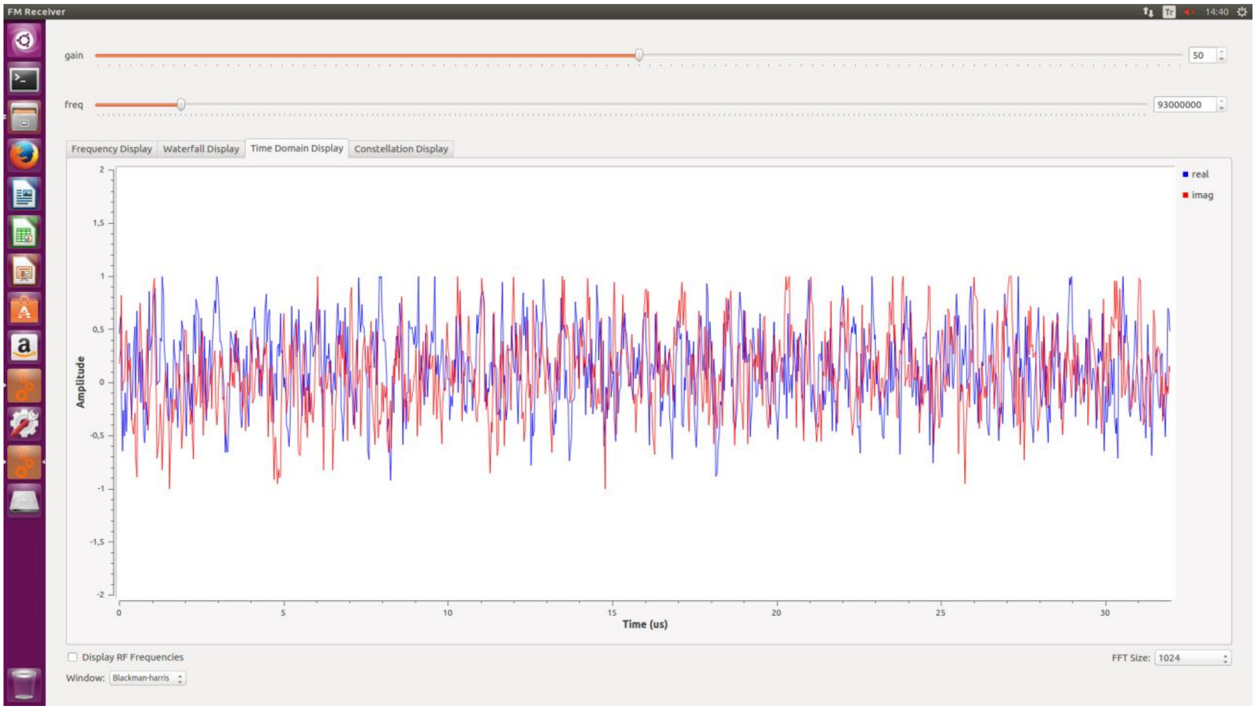
Screenshots:

a) Frequency Display

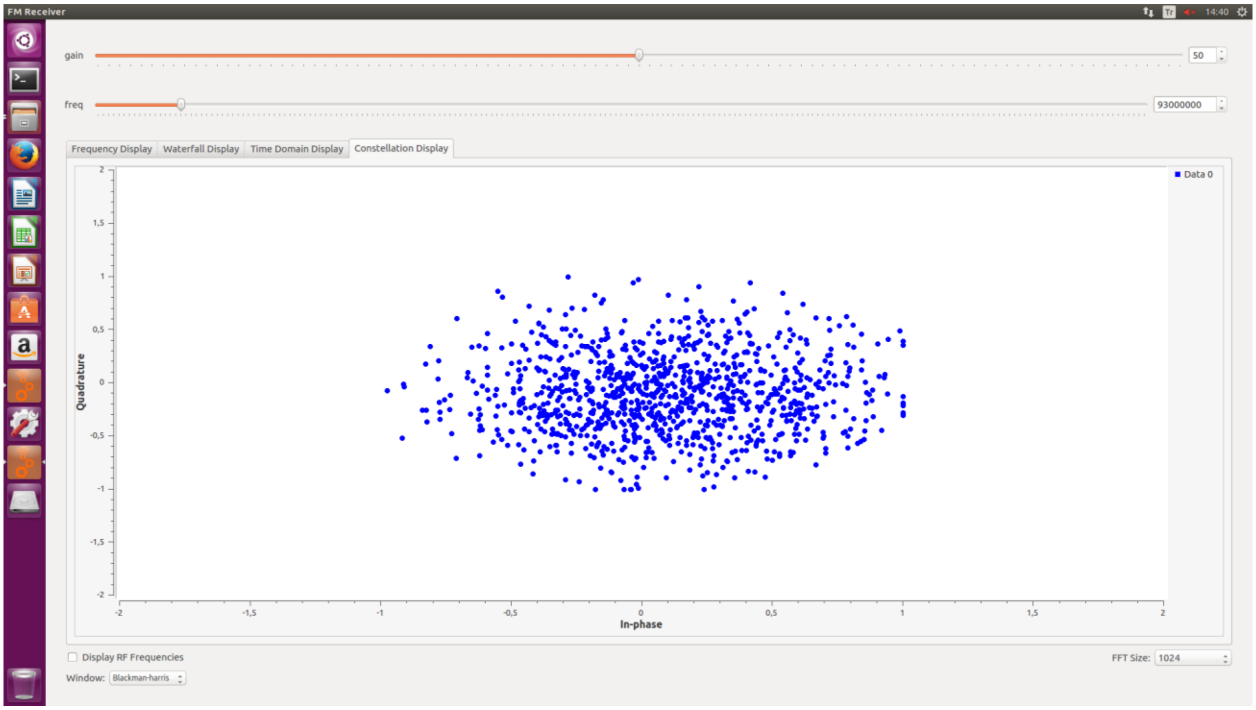
b) Waterfall Display



c) Time Domain Display

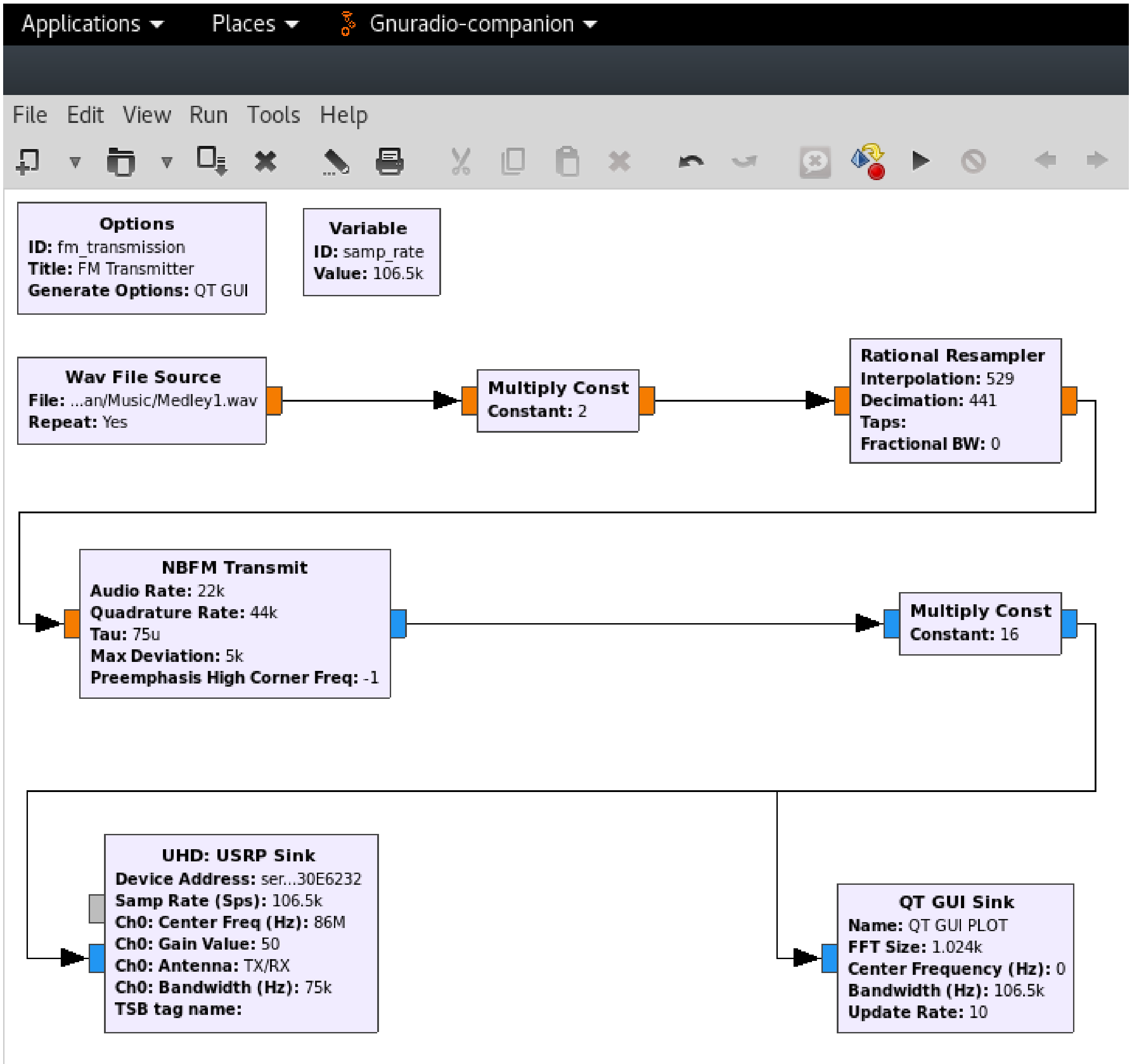


d) Constellation Display



c. Week 3 (28 – 30 JUNE)

Unlike last week, this week I made a FM radio transmitter program. This program reads music (in waveform audio file format(wave/wav)) from computer via serial port and sends to USRP B210 and it generates radio signal. I check the program with previous week program which is FM radio receiver and both of them works.



d. Week 4 (3 – 7 JULY)

This week, I have worked on how to communicate two USRP B210 and how to send and receive data packages with radio signals between two USRP B210. I have generated radio signal using GNU Radio on 2.4GHz signal band via TX/RX antenna on USRP B210 and using another GNU Radio program(flowgraph), I have seen change signal graph on 2.4GHz signal band.

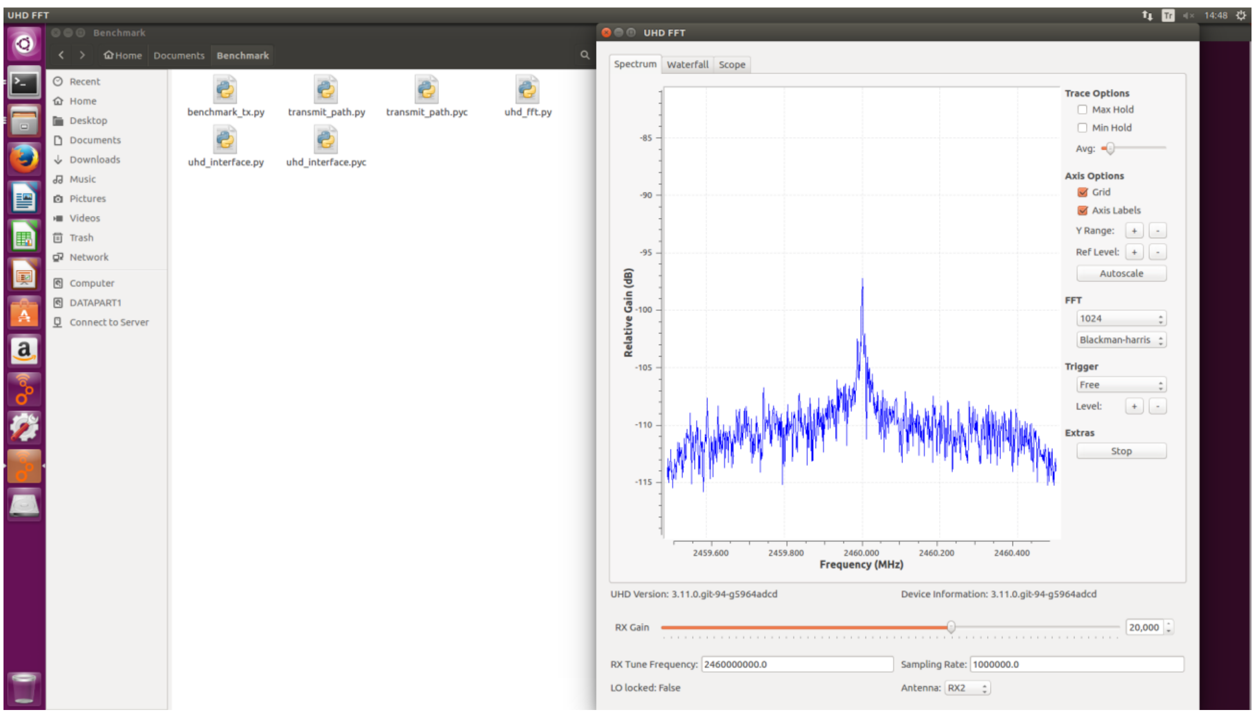
Firstly, I worked on written examples by GNU Radio developers in order to see signal transfer on frequency displays.

To see radio signal at specified frequency band, I open UHD_FFT program.

$cd /usr/local/bin/

$./uhd_fft –f 2.435G

Now, we can listen radio signals at appraoximately 2.435 GHz frequency band.



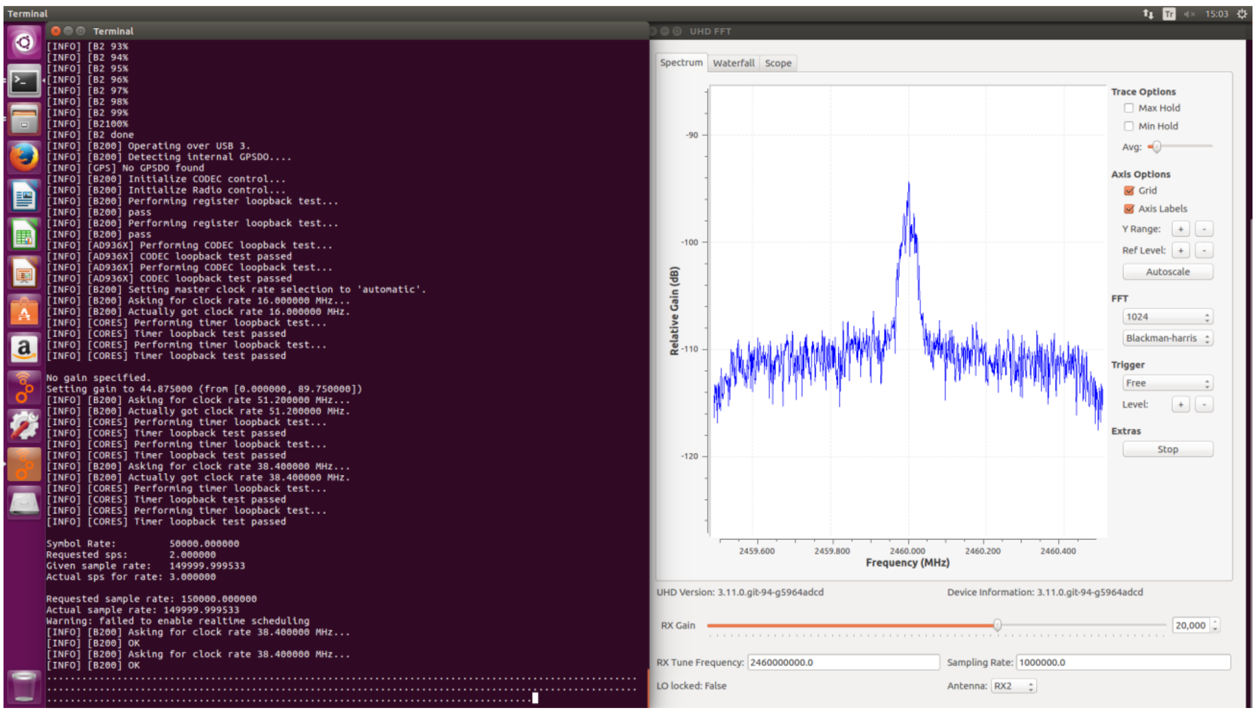
Now, we can generate radio signals from data packages. I order to do this, we can use Benchmark TX program. To run it:

$cd /usr/local/share/gnuradio/examples/digital/narrowband/

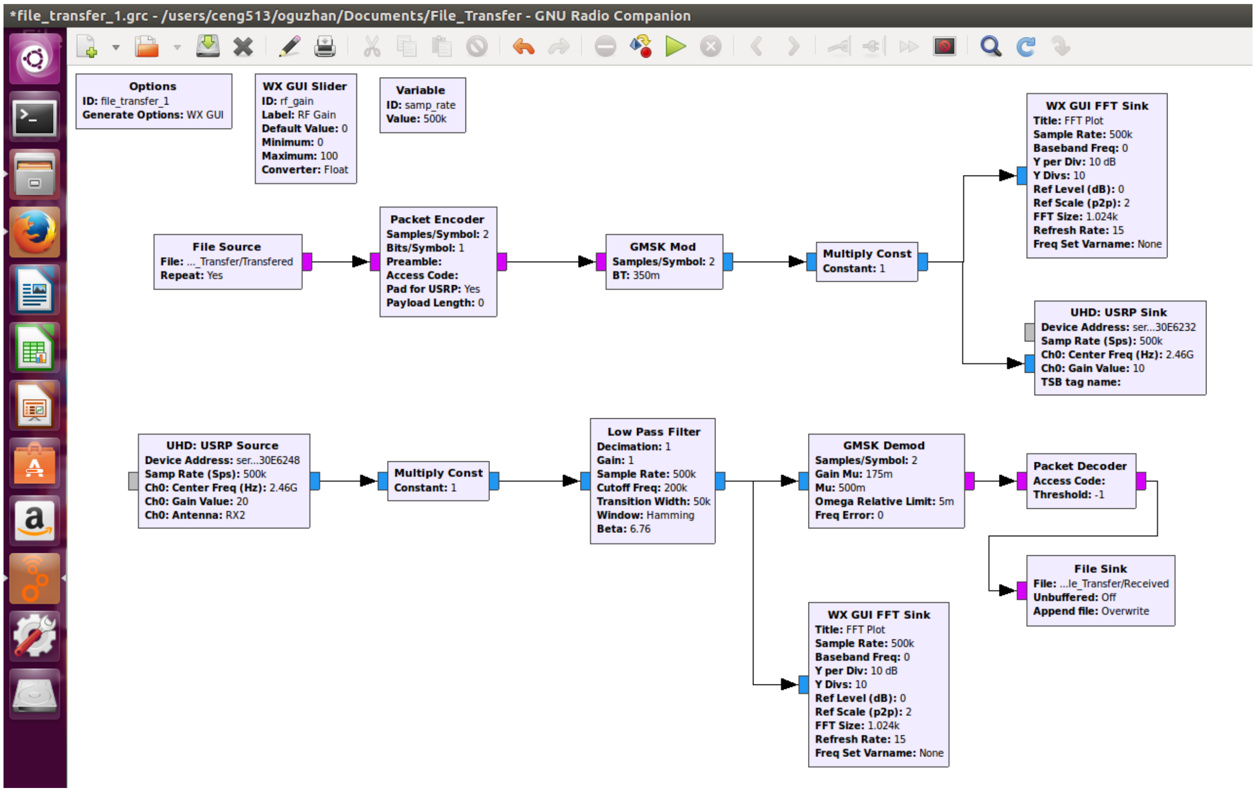
$./benchmark_tx.py -m qpsk –M 2 --discontinuous -f 2.435G

When we write this command, we use QPSK modulation, generate 2.435GHz frequency signals with 2 MB data packages. If we want to generate signal more time, we can increase megabytes size like –M 50 etc.

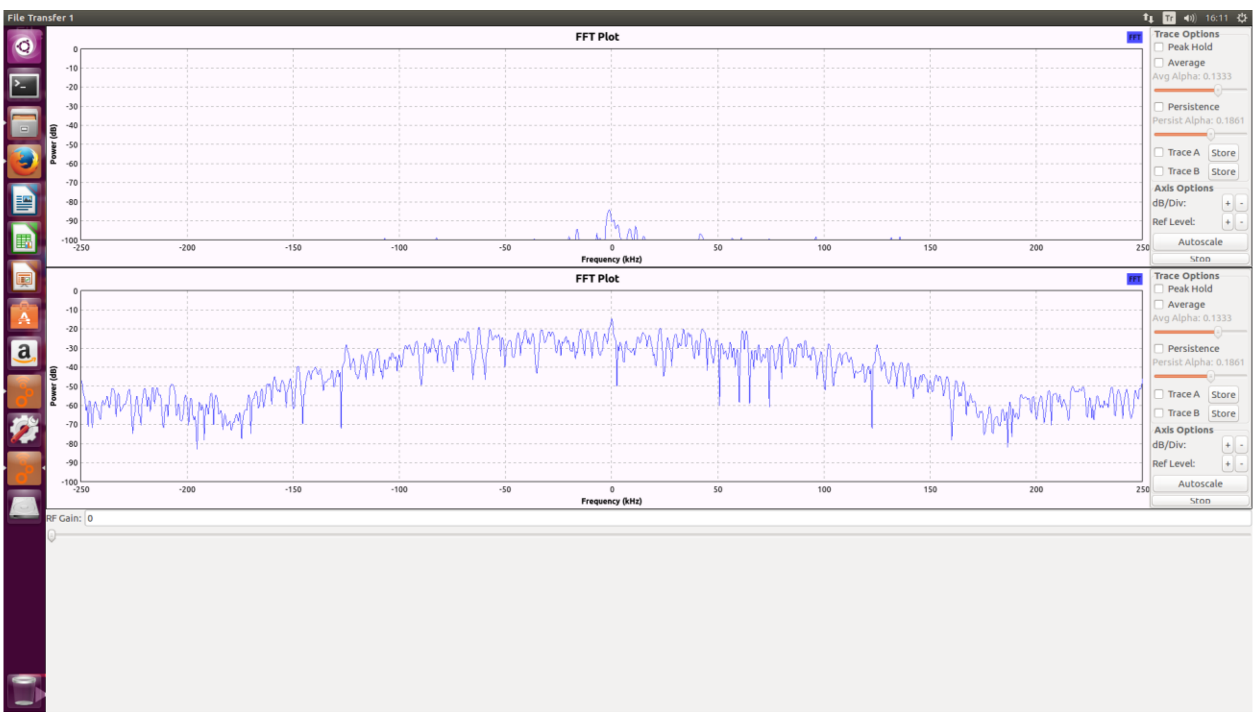
If we do this correctly, window below will appear.



Now, we can look at file transfer part. This program consist of two main part. First part reads text file byte by byte and converts it to radio signals via USRP Source Sink block. The second part takes this signal and converts it to text file. Flowgraph of the program:

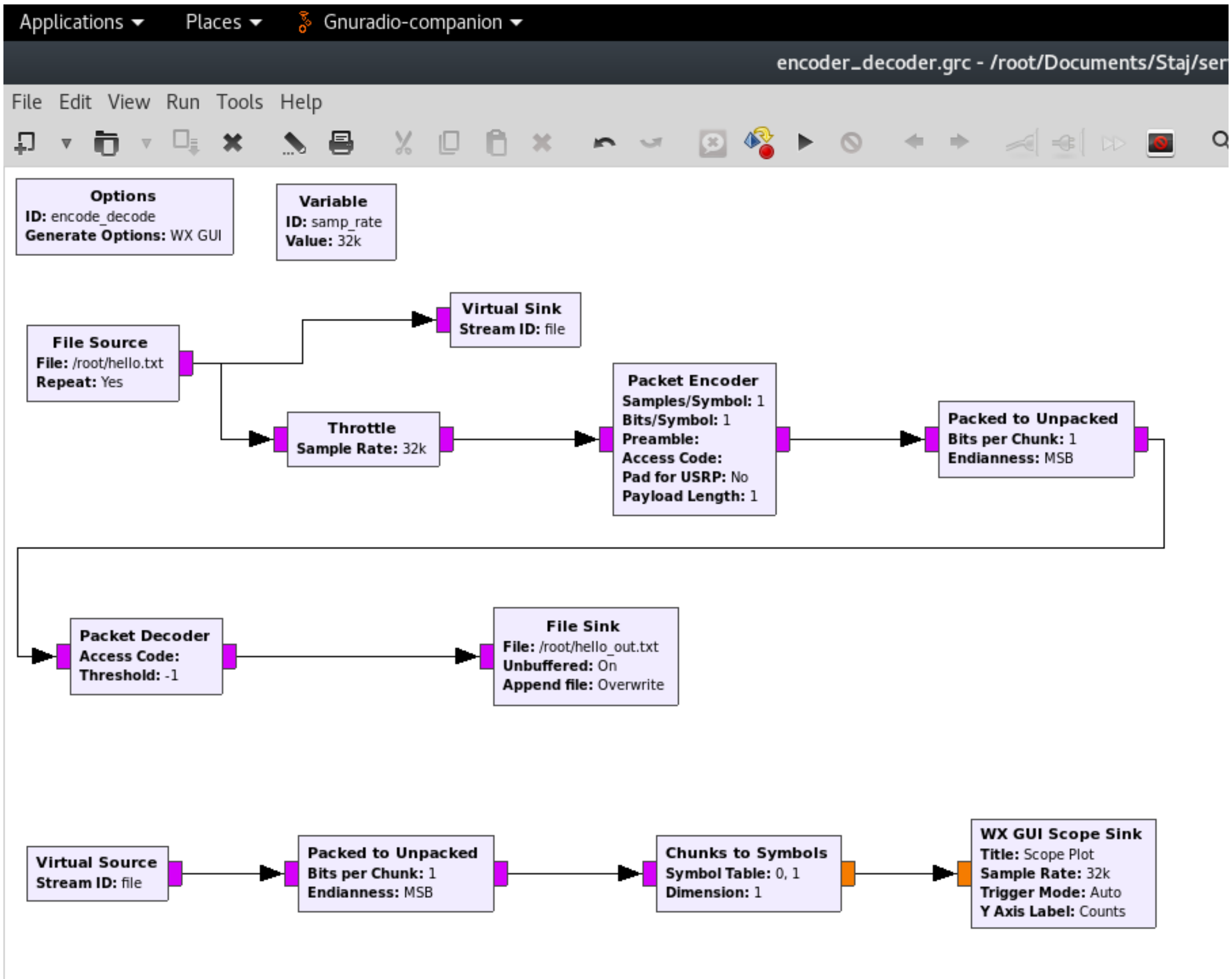


After running program:



e. Week 5 (10 – 14 JULY)

In this week, I tried to send text file and also .png file between two USRP B210. The flowgraph below is generated. It sends .png file bute by byte and shows file transmission in WX GUI SCOPE SINK bitwise.

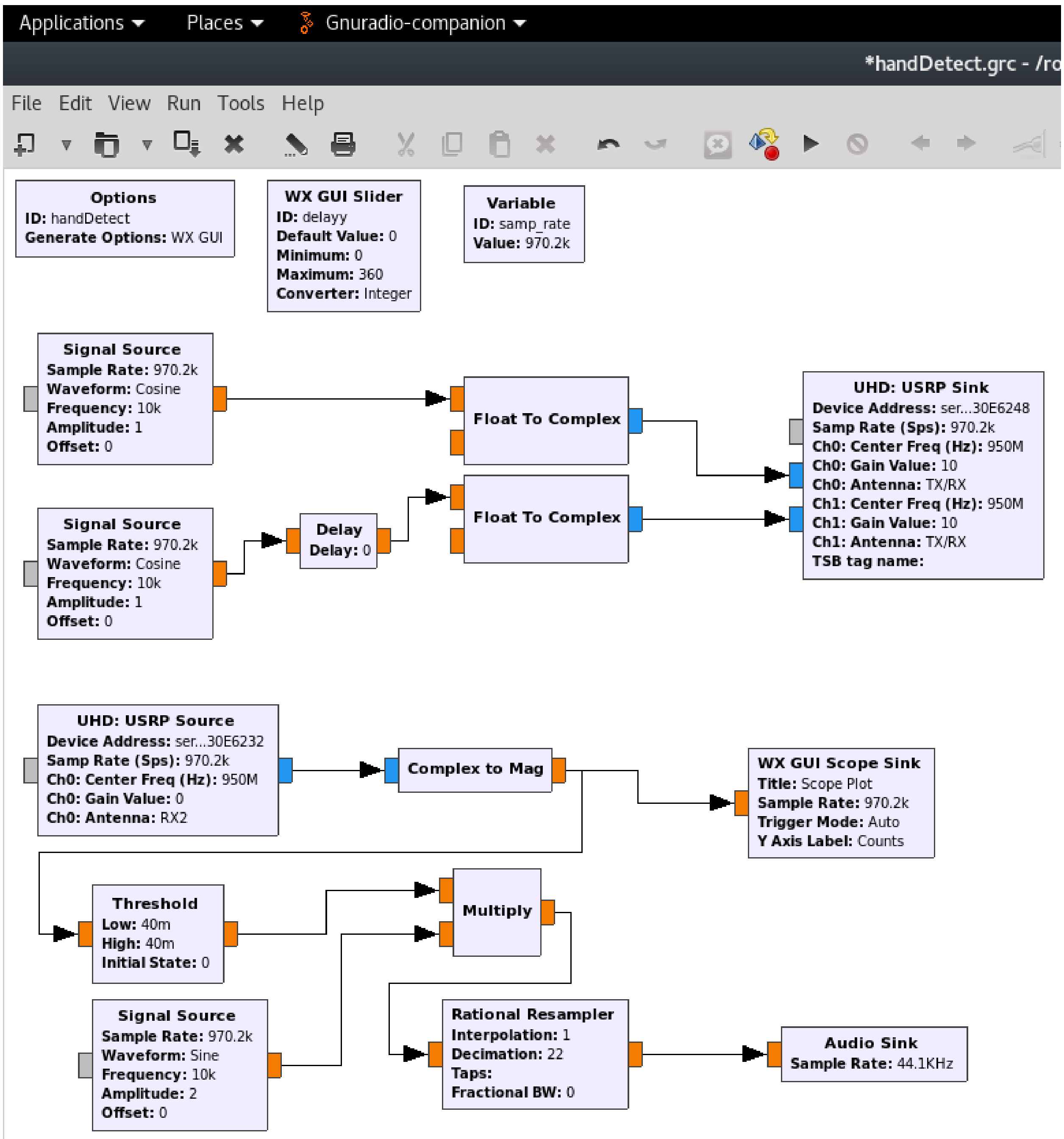


f. Week 6 (17 – 21 JULY)

In this week, I work on how to detect communication blocking objects and warn people.

Another program(flowgraph) is generated. In this program, first USRP B210 sends signal on its two transmission antenna and the second USRP B210 listens coming signal on its RX antenna. When any obstacle is appear between them, the program make a sinusoidal sound, a noise in order to warn people/developer/user etc.

Flowgraphs of the program:



g. Week 7 (24 – 25 JULY)

In this 2 days, I finished my intern report I started to writing before.

4. Conclusion

At the beginning, working about wireless communication systems and software-defined radio systems was difficult to understand and reason because I am not familiar this world, very new topic for me.

After some practice and writing simple programs, I amazed wireless world. I recognize that we can do a lot of things with radio signals. Unfortunately, I can do only few things because of lack of time.

To conclude, working on new topics and projects improves people and brings new viewpoint to the world.

5. References

[1]: https://en.wikipedia.org/wiki/GNU_Radio

[2]: https://en.wikipedia.org/wiki/Software-defined_radio

[3]: https://www.ettus.com/product/details/UB210-KIT