Exercise 1: Receiving AM Signals from Data Files

The USRP along with the Basic RX or LFRX daughterboard can be used to receive strong AM broadcast stations. Since neither of these daughterboards has a pre-amplifier it is important to use a good antenna or to add a pre-amp. The antenna is connected to the RX-A input on the daughterboard.

In this exercise you will be using GNU Radio to receive AM signals that have already been saved in a data file. This is to allow you to experiment with GNU Radio without a USRP.

Part 1. Listening to an AM Signal

Create a directory under your home directory named *exercises*. Copy the following two files into this *exercises* directory.

am_rcv4.py
am_usrp710.dat
sw_usrp.dat

Run this program by typing:

./am_rcv4.py 710

If you have trouble running this script, refer to "Creating and Running Python Scripts" under the Instructional Materials. You should hear about 15 seconds of an AM Broadcast station at 710 KHz. Observe the display. You should notice a peak on the display at 710 KHz. This is the station that you are listening to. When the recording has stopped the display will freeze. Move your cursor over the other peaks that are displayed and note the frequencies that are shown. These are other adjacent AM stations and you will be modifying the receiver to listen to those later in this exercise. When the recording has stopped, close the window.

This station was originally received using the USRP and saved to the data file *am_usrp710.dat*. Open the *am_rcv4.py* script in a text editor and examine the script. Note that the lines beginning with # are comments. Go down to the comment line:

#usrp is data source.

The 15 lines below this are used to set up the USRP and set its frequency and gain. These lines were originally used when the station you heard was received, but are commented out for this exercise.

Just below this you will see a commented section that begins:

#defines a file to collect usrp output.

This section creates a sink, *am_usrp_sink* that was used to collect this 15 seconds of broadcast and place it into the *am_usrp710.dat* file. The section below this creates a "source" that uses this data as input to the receiver, allowing us to test the receiver code without a USRP.

Part 2. Modifying the Receiver Frequency

Originally when this data was captured from the USRP, the receive frequency was set to 710 KHz, the frequency of the station that you have been listening to. As you observed, there are other stations in the spectrum that has been saved. In this section you will modify the GNU radio code to receive one of these other stations. Note the line in the receiver script:

offset = 0

This offset variable is used in a line below it as an argument in the channel filter (chan_filter). This is the filter that selects the station to be demodulated.

Earlier you should have observed a station at 790 KHz. If you did not, re-run part 1. Since this station is 80 KHz above the original frequency (710KHz) we need to shift the spectrum down by 80 KHz. To do this the offset should be set to -80000. Modify the script to replace the original offset line to:

offset = -80000

Save the file under the name *am_rcv5.py* and run it using:

./am_rcv5.py 710

You should now hear a different station. Note that the command line still has the original frequency of 710. This argument ONLY changes the center of the display and not the frequency being received. We leave it set to the receive frequency used when we recorded the signal so that the display is accurate.

Part 3. Setting the Offset from the Command Line

Open the script in the editor. The following line is used to set the frequency of the receiver.

station = parseargs(argv[1:]

argv[] is a python function for capturing arguments on the command line. In our case *argv[1:]* takes the first argument (and the only in our example), and passes it to *parseargs*. *parseargs* is a function defined near the bottom of the python script. Examine this function and you will see that it takes the value typed on the command line (710), multiplies it by 1000 and assigns it to the variable *station*.

Rather than modifying the script as you did in part 2 to receive a different frequency, we would like to be able to specify that frequency in the command line. This can be done as follows. Modify the offset line in the script to be:

offset = -(station + actual_shift_freq)

Note that *actual_shift_freq* is set to -710000 earlier in the script. Thus if we enter a frequency of 790 on the command line, *parseargs* will capture that value and multiply it by 1000 (790000) and pass it to the variable station. Then offset will be calculated to be:

offset = -(790000 – 710000) = -80000

as you entered previously. Make this single line modification, save as am_rcv6.py and run with:

./am_rcv6.py 790

You should hear the station at 790 KHz. Re-run the program with frequencies of 740, 640, and 670. Note that these are the strongest stations that you observed on the display.

Part 4. Varying the Volume

In the script you will see a line:

volumecontrol = gr.multiply_const_ff(.04)

This defines a constant multiplier block that multiplies its input by the specified constant. Modify the script to change the .04 to .06 and re-run. You should notice that the station is louder, but there may be some distortion. Try varying this constant up and down to see how it affects the volume.

Part 5. Listening to a Different File

A second file was captured under the name *sw_usrp.dat* . In order to listen to this file, make the following modifications on your script and save it as *am_rcv7.py*.

Change actual_shift_freq = -710000 to actual_shift_freq = -6000000
Change the file name in the am_usrp_source line from “am_usrp710.dat” to “sw_usrp.dat”.

Note that 6000000 is the frequency that the USRP was set to when the recording was made. Run this program with:

./am_rcv7.py 5950

You should hear Radio Havana Cuba. Look at the frequency peaks and try to receive some of the other stations. They will be weak but you should be able to hear some of them. Use headphones if the volume is too weak.

Challenges: Can you:

1. Vary the program so that the volume can be passed to it from the command line?
2. Vary the program so that the file name can be passed to it from the command line?