SOFTWARE BASED RADIO SIGNAL PROCESSING

Prateek Mohan Dayal

November 16, 2004

Contents

1 Software Radio Application Structure

GNU Radio platform and other software radio platforms in general follow a layered approach to programming. In GNU Radio, the core signal processing blocks are implemented in C++ for speed and robustness and the interfacing is done in python. A software called Simplified Wrapper and Interface Generator (SWIG) makes this possible. SWIG is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages. SWIG is primarily used with common scripting languages such as Perl, Python, Tcl/Tk and Ruby. When provided with a .i interface file and the

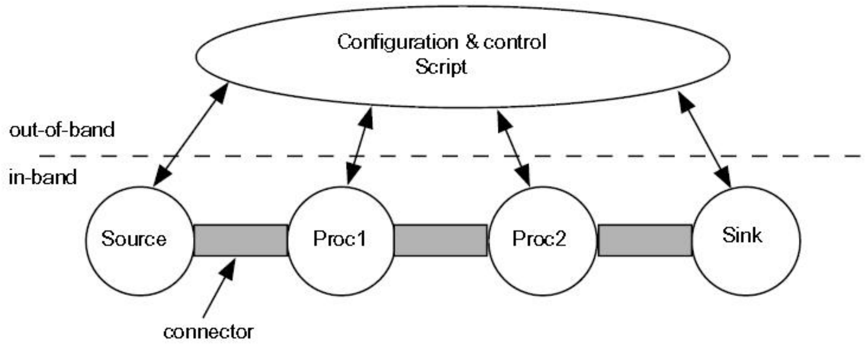


Figure 1: Software Radio Application Architecture

source file, swig can create a TCL/ python module that can be used for calling the functions listed in the interface file from python like any other python functions.

The python code essentially creates a signal flow graph [2], which connects a source, typically a file or the output of a ADC to various signal processing blocks and eventually to a sink, typically a file or sound card. The python script is responsible for creating, running and managing this signal flow graph. Fig. 1 illustrates this approach.

2 Signal Acquisition

Signal Acquisition is one of the most critical and difficult part of the whole application. The complexity arises due to the fact that one channel (Video + Audio) is around 7 Mhz and therefore the signal must be sampled atleast at the rate of 14 Mhz or higher after downconversion to baseband.

The preferred way to acquire signal for further processing in GNU Radio is to use Universal Software Radio Platform (USRP) [3]. The current version of USRP features 4 High-Speed AD Converters (64 MS/s, 12-bit Analog Devices AD9862) which can bandpass-sample signals of up to about 200 MHz, digitizing a band as wide as about 32 MHz, 4 High-Speed DA Converters (128 MS/s, 14-bit) to generate signals up to about 50 MHz. The interface for the USRP kit is high speed USB 2.0 which is fast enough for sustaining 32 MBps to a PC. One of the main features of the USRP is the presence of Altera EP1C12 Q240C8 "Cyclone" FieldProgrammableGateArray for high bandwidth math. This FPGA is also used for "decimation" so that the final output is at a rate that can be transferred to the PC over the USB interface. The usual i/o format is 16-bit I and 16-bit Q data (complex), resulting in 8M complex samples/sec

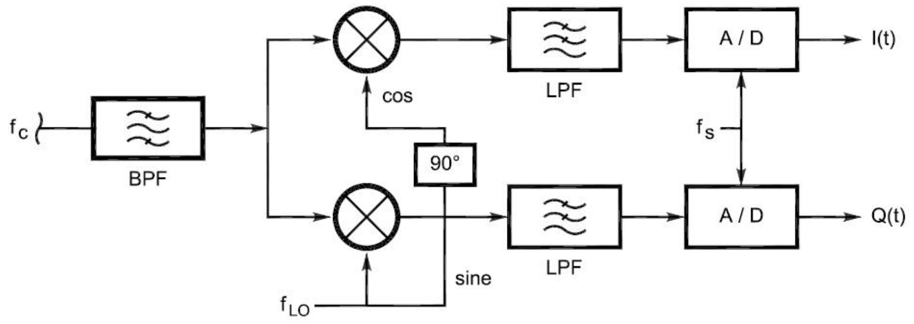


Figure 2: Quadrature Sampler Structure

across the USB.

The RF frontend used between the Antenna and the A/D chip onboard the USRP is a Microtune 4937 cable modem tuner module [4]. The tuner module can downconvert the frequencies ranging from 50 Mhz to 860 Mhz to an I.F of 43.75 Mhz (US) or 36.125 Mhz (Europe). There is a second downconversion step available which outputs to 5.75 Mhz (US) or 7.125 Mhz (Europe). In the setup that was used to acquire the samples for this project, the output was taken from the second stage at a frequency of 5.75 Mhz. The module contains the I.F amplifier so we can directly sample the output. The tuner module can be converted through the parallel port with the control software for linux available at any of the GNU Radio mirrors.

The output of the cable tuner is fed to one of the A/D converters. The I.F signal at 5.75 Mhz is sampled at 64 MSamples per second, 12 bits but samples and then later on downconverted to complex baseband and decimated to 8M samples per second in the USRP itself. The data thus contains real and imaginary 16 bit numbers. The theory of quadrature sampling is used for this purpose, which allows us to sample at much lower sampling rate if use two samplers that result in I and Q channel or the real and imaginary channel. Fig. 2 illustrates this approach.

3 Quadrature Sampling

Quadrature Sampling [5] is the process of translating the spectrum of a signal to the baseband and then sampling it. The concept can be graphically represented as in Fig 3.

The process of quadrature sampling can be easily understood if we remember that sin and cos can be written as sum and difference of exponentials. Also multiplying a signal by $e^{j2\pi f_c t}$ shifts the signal up by f_c in the frequency domain and multiplying a signal by $e^{-j2\pi f_c t}$ shifts the signal down by f_c in the frequency domain. Fig. 4 and Fig. 5 utilize this principle.

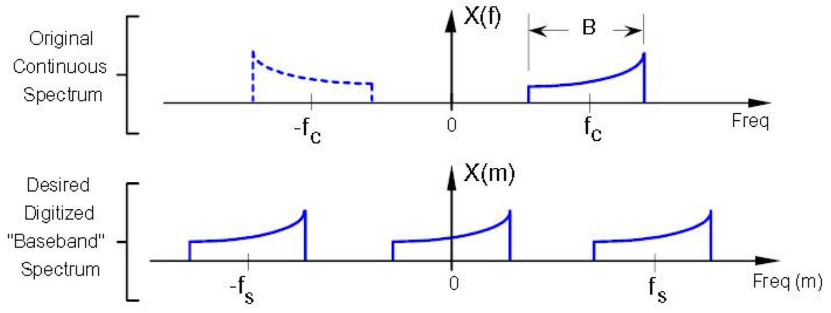


Figure 3: Quadrature Sampling Objective

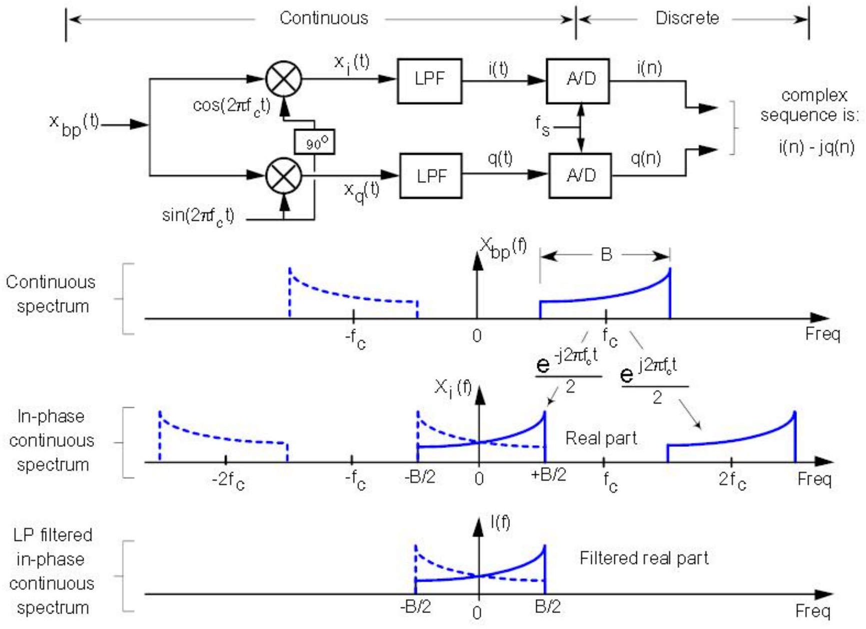


Figure 4: Frequency Spectrum of the I channel

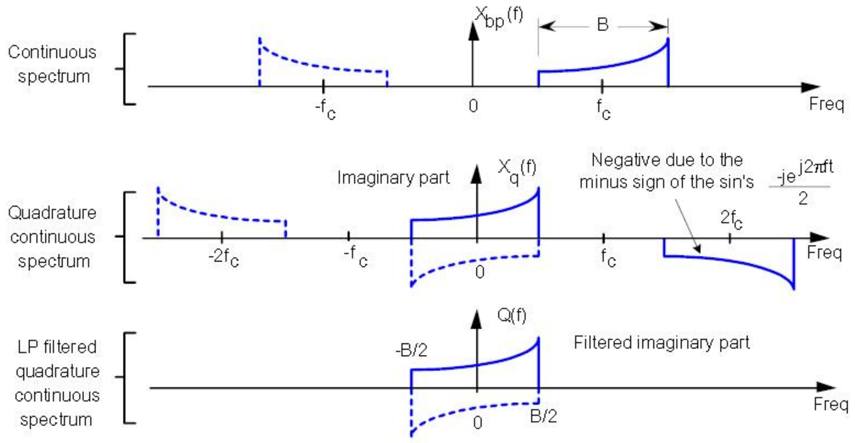

Continuous spectrum
$X_{bp}(f)$
B
$-f_c$
0
f_c
Freq
Imaginary part
$X_q(f)$
Negative due to the minus sign of the sin's
$\frac{-je^{j2\pi ft}}{2}$
Quadrature continuous spectrum
$-2f_c$
$-f_c$
0
f_c
$2f_c$
Freq
Q(f)
LP filtered quadrature continuous spectrum
-B/2
Filtered imaginary part
0
B/2
Freq

Figure 5: Frequency Spectrum of the Q channel

If we add the I and Q channels as I-jQ, we get the desired spectrum. The -j operation in the frequency domain results in turning the spectrum of Q channel by an angle -90 degrees and getting it in the plane of the I channel, so that the spectras can be added. This step is shown graphically in Fig. 6.

3.1 Benefits of Quadrature Sampling

A modulated signal has an amplitude and a phase component. The modulation schemes such as AM modulate the carrier amplitude and hence by evaluating the magnitude of the sampled signal, signal demodulation can be performed. The modulation schemes such as FM modulate the phase of the signal and to demodulate the signal, phase information has to be extracted. If we have the I and Q components of any signal, we can calculate the magnitude and instantaneous phase information. Thus we can demodulate even FM in a much simpler fashion. Quadrature signals can also be used analytically to remove the image frequencies and leave only the desired sideband.

4 Concept of Decimation

The input data rate in GNU Radio is typically 8 MSamples per second where each sample is 4 bytes long. This is so because the USRP kit typically gives the output at this data rate. Also the signal sampled this way is large bandwidth signal, in which the region of interest may be a smaller band. For example in the application that we are developing, the bandwidth is around 7 MHz but if we want to demodulate the audio carrier only, the band is around 200 kHz

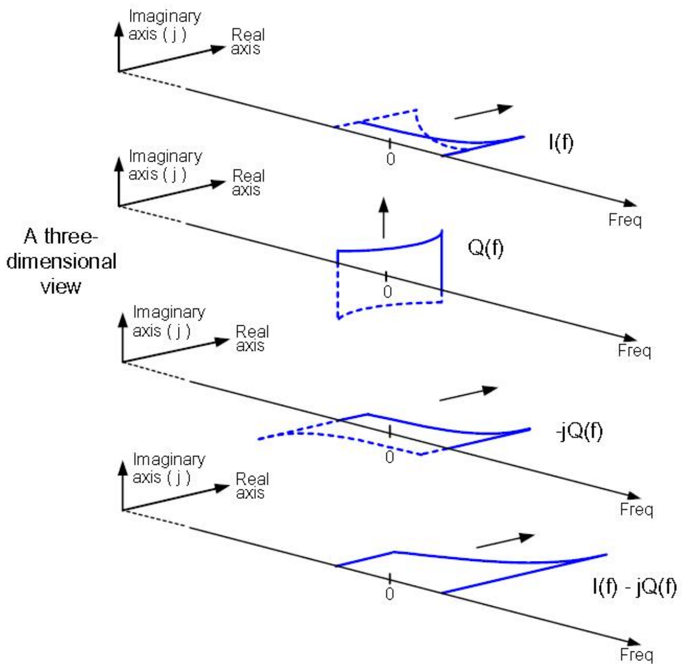


Figure 6: Frequency Spectrum of the Q channel

only. Therefore we can reduce the data rate at different part of the code and speed up the processing. This process is called decimation. However in order to avoid aliasing during decimation, the frequency content of the undecimated signal should be limited to an appropriate range. The process is detailed later in the report.

5 Application Example: FM Signal Demodulation

We have been able to demodulate the audio part of the NTSC signal. If the signal has been converted to the baseband, the audio carrier is around 2.75 Mhz and the Video carrier is around -2.75 Mhz. The audio signal is frequency modulated and spans around 75 khz on both side of the carrier. The frequency demodulation involves the steps detailed in the following subsections. Several concepts introduced earlier are used in the application code.

The data is read as complex numbers where the real and imaginary parts are both two bytes long. Complex number mathematics is used all throughout as it is normally faster and steps like Quadrature Demodulation can be implemented very easily with it. The signal flow graph for the demodulator is show in Fig. 8 and explained below.

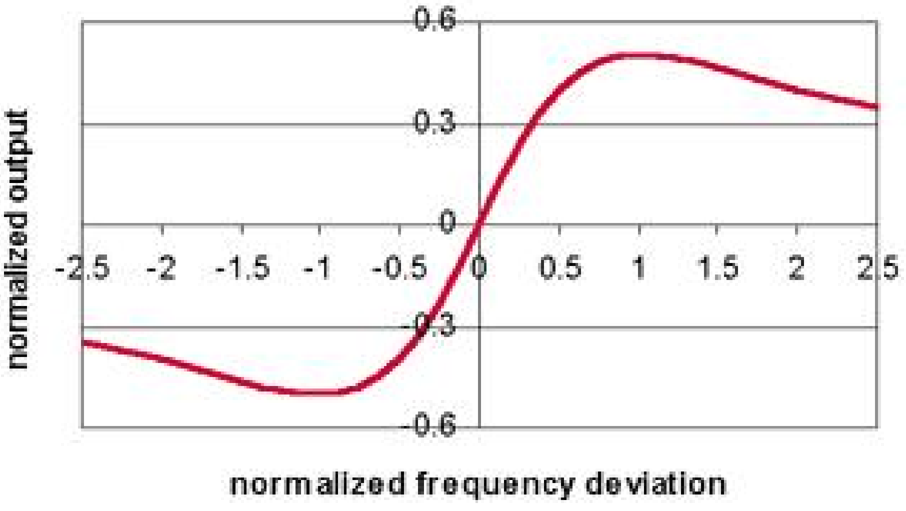


Figure 7: Frequency Response of the Quadrature Detector

5.1 Downconversion and Decimation

The audio carrier is centred around 2.75 Mhz. It is downconverted to baseband by multiplying it with a complex sinusoid. The bandwidth of the audio signal is around 150 khz and therefore it can be decimated to a signal with data rate much lower than the input data rate. We choose a data rate of 320 Kbps. In order to avoid aliasing, a low pass filter is applied before decimation. The cutoff frequency of the filter is chosen to be around 150 kHz and Hanning window is used. GNU Radio library provides a filter design function that returns the FIR filter coefficients. The decimation and downconversion can be performed in a single step given the low pass filter coefficients. This step also determines to a large order, the speed of overall demodulation. Complex multiplication used in this step is the bottleneck when it comes to speed. By increasing the decimation, the speed can be increased, but as we filter out more and more signal information, the voice quality degrades.

5.2 Quadrature Demodulation

The quadrature demodulator is available in GNU Radio as a built in function [8]. The frequency response of the quadrature detector is shown in Fig. 7.

5.3 Sink: Sound Card

The demodulated data is in a format that is incompatible with the soundcard. GNU Radio provides a function to compute the sound card compatible output from the demodulated data. Other sinks that can be used in place of the sound card are File Sinks, FFT Sinks etc. We describe the FFT sink in detail below.

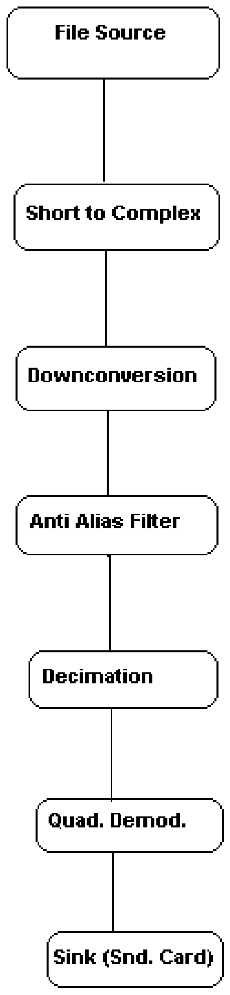
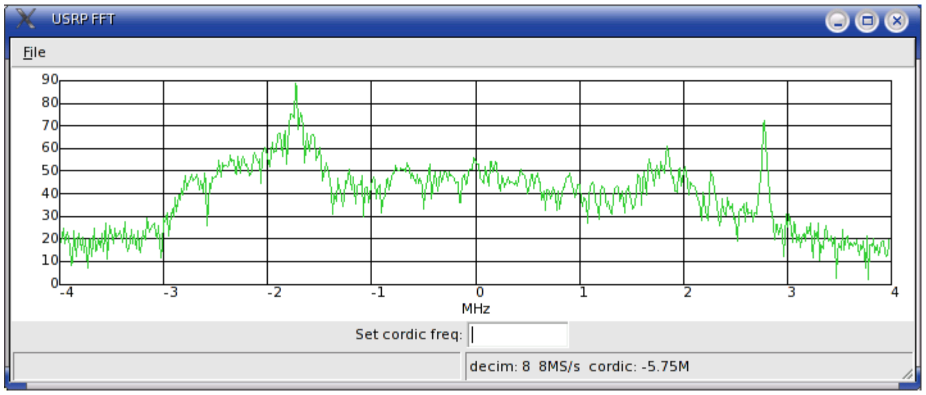


Figure 8: Signal Flow Graph for FM demodulation



decim: 8 8MS/s cordic: -5.75M

Figure 9: FFT of the NTSC data

5.4 Sink: FFT Sink

The FFT sink is capable of displaying the FFT of any signal in real time. The FFT is computed using the award winning open source fft3w library. For building the GUI, wxPython is used. Also Numeric and Numarray library for python are used, which can provide C like array handling capabilities which provide robust and fast data manipulation capability to python. The number of points for which FFT have to be computed, the input data rate etc can be controlled by the user. Fig. 9 shows the FFT of the input data which clearly shows the two carriers, sound and video.

6 Conclusions

This report presents the basic structure of a typical Software Radio application and through FM demodulation, the concepts are exemplified. The working real time FM demodulation is a proof of concept that Software Radio applications can be built on a general purpose computing environment. Future work will involve the demodulation of the video part of the signal, possibly in real time.

7 Acknowledgments

I am very thankful to the enthusiastic people associated with the GNU Radio project, especially Eric Blossom, who provided me with the TV signal samples and a lot of guidance during the course of projects. I am also thankful to many others who resolved my infinite compilation issues during the course of this work. Cheers to the Open Source Community !!!

References

[1] Joseph Mitola, “Software Radio - Cognitive Radio,” available at *www.ourworld.compuserve.com/homepages/jmitola/*

[2] “Vanu Inc website,” *www.vanu.com/*

[3] “Universal Software Radio Platform Wiki,” available at *com-sec.com/wiki?UniversalSoftwareRadioPeripheral*

[4] “Microtune 49xx reference.” available at *http://www.microtune.com/products/49xx_Series.html*

[5] Richard Lyons, “Quadrature Signals: Complex, But Not Complicated,” available at *http://dspguru.com/info/tutor/QuadSignals.pdf*

[6] “SPEAKeasy, the Military Software Radio website,” available at *http://www.rl.af.mil:8001/SPEAKEASY*

[7] “RF Design Online Article,” available at *http://rfdesign.com/mag/radio_field_trials_allsoftware/*

[8] “GNU Radio Project website, ”available at *www.gnu.org/software/gnuradio/*

[9] “Measurement Computing PCI DAS 4020 website, ”avaibale at *http://www.measurementcomputing.com*

[10] Steve Somers, “NTSC Decoding Basics (Part 1-4),” available at *http://www.extron.com/technology/archive.asp?id=ntscdb1*